ABSTRACT

TANG, WEN. Optimized Structure Learning in Inference Problems. (Under the direction of Hamid Krim.)

The essence of a human to understand the chaotic world around, following the sensing the mapped image of the real world, classification is also a basic, critical and essential inference task in Machine Learning and Artificial Intelligence areas to make a machine understand and learn the world after using the imaged and collected data. The classification tasks in Machine Learning are divided into two taxonomies based on whether test objects are seen or unseen during the training, which are respectively named as Classification and Zero-shot Classification.

For the basic Classification tasks, an integrated/end-to-end method for jointly learning the classifier and selecting features is the most common and preferred method in Supervised Learning. Since during human classification, defining criteria for different equivalence classes favors distinguishibility, the discrimination of a classifier also becomes crucial and critical to classification inference tasks. As the different structures of the integrated method/end-to-end model determine different discriminative abilities of a classifier, for enhanced classification performance, we thus primarily explore, in this thesis, different optimized structures for different inference problems.

In Chapter 2, we first try to use the integrated method to jointly learn an analysis dictionary, a structure of the union of subspaces (UoS) and a universal classifier to explore the intrinsic structure of the invariant low-dimensional spaces. This results in an enhanced capability of discrimination and to ensure a more complete consistent classification.

In recent years, Zero-shot Classification has also increasingly received much interest in the Machine Learning area. Since Zero-Shot is to transfer the known knowledge of seen objects to unseen ones, the underlying invariant space for different “concepts” is the key to solving such Zero-Shot Classification task.

In Chapter 3, we hence focus on optimizing the learning structure for the Zero-Shot Classification task. We propose a novel inductive ZSL model, which is based on projecting both visual and semantic features into a common distinct latent space with class-specific knowledge, and on reconstructing both visual and semantic features by such a distinct common space thus narrowing the domain shift gap.

With developments of hardware capability, the amounts of data are also dramatically grown. To accommodate such a tremendous dataset, the structures of such an integrated method/end-to-end model for Classification require a critical and dominant factor of classifier discrimination, which in turn, focuses on the ability of selecting the various distinct class features.

In Chapter 4, to exploit the deep structure of data, specifically, Convolutional Neural Networks (CNNs), we address the Classification task of very large datasets by principally seeking an optimized. While the Dictionary Learning (DL) method is used in early feature selection stages, a novel differentiable programming method based on Deep DL (DDL) is proposed in Chapter 4, to improve the
discrimination and significantly enhance the efficiency of the original DDL. The newly proposed and developed "capsule" can also be flexibly integrated into any neural network architectures to achieve a better performance than the generic CNNs.
Optimized Structure Learning in Inference Problems

by

Wen Tang

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Computer Engineering

Raleigh, North Carolina

2020

APPROVED BY:

Patrick Combettes

Edgar Lobaton

Tianfu Wu

Hamid Krim
Chair of Advisory Committee
BIOGRAPHY

Wen Tang received her one B.S. degree in Information and Computing Science from East China University of Science and Technology, Shanghai, China, in 2012. In the same year, she also received her another B.S. degree in Computer Science from Fayetteville State University, Fayetteville, NC, USA. In August 2014, she started her PhD in the Department of Electrical and Computer Engineering in North Carolina State University and joined Vision, Information and Statistical Signal Theories and Applications (VISSTA) group under the direction of Dr. Hamid Krim. Further, she has been a visiting student at the Center for Visual Computing at CentraleSupelec, France. Her current research interests include machine learning and its applications for computer vision. More specifically, she is working on developing conventional/deep dictionary learning methods, especially, the analysis dictionary learning methods for different kinds of image classification problems, such as conventional classification and zero-shot classification.
ACKNOWLEDGEMENTS

First and foremost, I express my special gratitude and appreciation to my advisor, Dr. Hamid Krim, who helped me, supported me and enlightened me during my research. Without him, it would not have been possible for me to earn my Ph.D..

I also thank the rest of my advisory committee: Dr. Patrick Combettes, Dr. Edgar Lobaton, Dr. Tianfu Wu. I am truly thankful for your suggestions, comments, and questions. Besides, I am also very grateful to Dr. Liyi Dai and Dr. Jean-Christophe Pesquet for their time, patience and advise on my research.

In addition, I really appreciate having interacted and worked with Dr. Ashkan Panahi and Dr. Han Wang, the postdoctoral researchers in the VISSTA Laboratory, and for their guidance during my PhD. Last but not least, I greatly valued all the discussions and lengthy debates within the VISSTA Laboratory. Shahin Mahdizadehaghdam, Hui Guan, Yuming Huang, Bo Jiang, Kenneth Tran, Siddharth Roheda and Sally Ghanem have been great officemates and friends.

Finally, I want to express my gratitude to my parents, and many others who could not be mentioned here, but have helped by supporting me during this challenging journey.
# TABLE OF CONTENTS

LIST OF TABLES ........................................................................................................ vi

LIST OF FIGURES ....................................................................................................... vii

Chapter 1 INTRODUCTION ......................................................................................... 1

Chapter 2 ANALYSIS DICTIONARY LEARNING BASED CLASSIFICATION: STRUCTURE FOR ROBUSTNESS ................................................................. 5
  2.1 Introduction ......................................................................................................... 5
  2.2 Structured Analysis Dictionary Learning ............................................................ 9
    2.2.1 Notation .................................................................................................. 9
    2.2.2 Structured ADL Method .................................................................... 9
  2.3 Algorithmic Solution ........................................................................................ 12
    2.3.1 SADL Algorithm ............................................................................. 12
    2.3.2 Distributed SADL Algorithm .......................................................... 15
  2.4 Classification Procedure .................................................................................. 15
  2.5 Convergence ..................................................................................................... 17
  2.6 Experiments and Results .................................................................................. 18
    2.6.1 Parameter Settings ........................................................................... 18
    2.6.2 State-of-the-art Methods .................................................................. 18
    2.6.3 Extended YaleB ................................................................................. 19
    2.6.4 AR Face .............................................................................................. 20
    2.6.5 Caltech 101 ....................................................................................... 21
    2.6.6 Caltech 256 ....................................................................................... 24
    2.6.7 Scene 15 ............................................................................................. 26
    2.6.8 Comparative Evaluation .................................................................... 27
  2.7 Conclusion ......................................................................................................... 27

Chapter 3 JOINT CONCEPT MATCHING BASED LEARNING FOR ZERO-SHOT RECOGNITION ........................................................................ 29
  3.1 Introduction ....................................................................................................... 29
  3.2 Related Work .................................................................................................... 32
    3.2.1 Class-specific Information .................................................................. 32
    3.2.2 Projection Learning .......................................................................... 32
    3.2.3 Projection Domain Shift ................................................................... 33
  3.3 Methodology ..................................................................................................... 33
    3.3.1 Notation .............................................................................................. 33
    3.3.2 Problem Statement ............................................................................ 34
    3.3.3 Model Formulation .......................................................................... 34
    3.3.4 Algorithmic Solution ....................................................................... 34
    3.3.5 Zero-shot Recognition .................................................................... 36
  3.4 Experiments ....................................................................................................... 37
    3.4.1 Datasets and Settings ....................................................................... 37
    3.4.2 Experiment Results .......................................................................... 40
    3.4.3 Further Evaluations .......................................................................... 41
  3.5 Conclusion ......................................................................................................... 46
Chapter 4  DEEP TRANSFORM AND METRIC LEARNING NETWORK: WEDDING DEEP DICTIONARY LEARNING AND NEURAL NETWORKS  47
4.1 Introduction ......................................................... 47
4.2 Preliminaries ....................................................... 50
  4.2.1 Notation ....................................................... 50
  4.2.2 Dictionary Learning for Classification ....................... 50
  4.2.3 Deep Dictionary Learning ................................... 51
4.3 Joint Deep Metric and Transform Learning ......................... 52
  4.3.1 Proximal interpretation .................................... 52
  4.3.2 Multilayer representation .................................. 53
4.4 Q-Metric Learning ............................................... 54
  4.4.1 Prox computation ........................................... 54
  4.4.2 RNN implementation ....................................... 55
  4.4.3 Training procedure ......................................... 56
4.5 Experiments and Results ......................................... 56
  4.5.1 Architectures ............................................... 56
  4.5.2 Datasets and Training Settings .......................... 57
  4.5.3 Results .................................................... 58
4.6 Conclusion ....................................................... 60

Chapter 5  CONCLUSION AND FUTURE WORK  62
5.1 Conclusion ....................................................... 62
5.2 Future Work .................................................... 63

BIBLIOGRAPHY .......................................................... 65

APPENDICES ............................................................ 75
Appendix A  Proof of the Algorithm1 in Chapter 2 .................. 76
Appendix B  Proof of the Algorithm 3 in Chapter 3 ................ 84
Appendix C  Alternative derivation of Algorithm 4 in Chapter 4 .... 87
Appendix D  Illustration of DeTraMe-Net Architectures in Chapter 4 89
  D.1 DeTraMe-PlainNet ............................................ 89
  D.2 DeTraMe-ResNet ............................................... 89
**LIST OF TABLES**

| Table 2.1 | Classification Results on Extended YaleB Dataset | 20 |
| Table 2.2 | Classification Results on AR Dataset | 22 |
| Table 2.3 | Classification Results on Caltech 101 Dataset | 22 |
| Table 2.4 | Training Time and # Training Samples on Caltech 101 Dataset | 23 |
| Table 2.5 | Classification Results on Caltech 256 Dataset | 25 |
| Table 2.6 | Classification Results on Scene 15 Dataset | 26 |
| Table 2.7 | Classification Results of Different Constraints Comparison | 27 |

| Table 3.1 | The details of four evaluated datasets. Notation: 'SS'- the semantic space; 'A'- the attribute, 'W'- the word vector; 'SS-D'- the dimension of the semantic space. | 38 |
| Table 3.2 | The standard ZSL classification accuracy (%). For ImNet, hit @5 is reported. For visual Features: G - GoogleNet [Sze15]; V - VGG19 [Sim14]; R - ResNet101 [He16b]. | 39 |
| Table 3.3 | The comparative results(%) of generalized ZSL as setting in [Cha16b]. All the results are tests on GoogleNet visual features. | 41 |
| Table 3.4 | The evaluation of the contributions of JCMSPL and the improtance of both reconstruction constraint and class-specific information. | 42 |
| Table 3.5 | The parameters $\lambda_1, \ldots, \lambda_4$ settings are used for four different datasets. | 44 |
| Table 4.1 | Model Description of PlainNet | 58 |
| Table 4.2 | ResNet Model [He16a] | 58 |
| Table 4.3 | DeTraMe-Net vs. DDL: Both DeTraMe-Net and DDL with 9-layer follow the ALL-CNN architecture in [Spr14], as listed in the fourth column in Table 4.1. | 58 |
| Table 4.4 | CIFAR10, CIFAR100 and SVHN Results. The dataset with + is trained with simple translation and flipping data augmentation. The (number) is the number of iteration in RNN. All the presented results are re-implemented and run by using the same settings. | 59 |
LIST OF FIGURES

Figure 2.1  SDL reconstructs data $X$ by the dictionary $D$ with the sparse representations $A$. ADL applies the dictionary $\Omega$ to data $X$ and results in the sparse representations $U$. $\| \cdot \|_p$ can be either $l_1$ norm or $l_0$ norm. If and only if $D$ and $\Omega$ are square matrices, SDL and ADL are equivalent to each other. ...................... 6

Figure 2.2  $Q(\Omega X)$ on Scene 15 Dataset ......................................................... 15
Figure 2.3  $WQ(\Omega X)$ on Scene 15 Dataset ......................................................... 17
Figure 2.4  Extended YaleB Dataset Examples ......................................................... 19
Figure 2.5  Classification Accuracy versus Dictionary Size ......................................... 20
Figure 2.6  Training Time versus Dictionary Size ......................................................... 21
Figure 2.7  AR Dataset Examples ............................................................................. 21
Figure 2.8  Caltech 101 Dataset Examples ................................................................... 22
Figure 2.9  Distributed SADL on Caltech 101: $N$ is the number of clusters used. $N = 1$ is centralized. Training set is divided into $N$ groups. ................................. 23
Figure 2.10  Training time of Distributed SADL on Caltech 101: horizontal axis is the number of cluster, vertical axis is the real training time in second. The left blue vertical axis is the time for initialization of DSADL, and the right orange vertical axis is the time for variables alternating iteration, i.e., while loop part in our Algorithm 2. ................................................................. 24
Figure 2.11  Caltech 256 Dataset Examples ................................................................. 25
Figure 2.12  Scene 15 Dataset Examples ................................................................... 26

Figure 3.1  The Illustration of JCMSPL ................................................................. 30
Figure 3.2  The framework of our purposed JCMSPL is comprised of 3 procedures. (I) A CNN is first used to extract the visual features $X$, such as GoogleNet [Sze15], VGG-19 [Sim14] and ResNet101 [He16b], and then project them by the function $A$ into the common concept space $C$. Finally, $A^T$ is employed to reconstruct the $X$ from concept space $C$. (II) The attributes/word vector $Y$ is used to embed the different classes into the semantic space. The project function $B$ maps the semantic feature $Y$ into the common concept space $C$ as well, and $B^T$ reconstruct $Y$ from $C$. (III) A block-diagonal matrix $H$ supervise and support the common concept space $C$ to be distinct and separable, where the light-yellow part shown in the class-specific information is with all entrances of 0 and the dark-yellow part is occupied by the elements of 1. 33

Figure 3.3  Ablation study results of standard ZSL on two medium-scale datasets. ...... 43
Figure 3.4  Parameters analysis for $\lambda_1,\ldots,\lambda_4$ .................................................. 43
Figure 3.5  Convergence of JCMSPL on AwA and CUB Datasets. The left-side vertical axis is for AwA dataset, while the right-side vertical axis is for CUB dataset. 45
Figure 4.1 2-layer DeTraMe-Net model. Each layer solves a DL problem, which is transformed into the combination of Transforming Learning (i.e., linear layer in brown dashed lines) and Q-Metric Learning (i.e., RNN in red dashed dot lines). A truncated 2-iterations RNN is unfolded. Sparsity is imposed by shifted-ReLU functions. In the forward pass, we first use a linear layer to learn the new representation $Z^{(1)}$ for input data $X$. The RNN is then used to iteratively learn the optimal sparse representation $U^{(1)}_\ast$. For the second layer, the sparse representation $U^{(1)}_\ast$ is used as input to learn the second layer sparse representation $U^{(2)}_\ast$. Finally, a cross-entropy loss based on $U^{(2)}_\ast$ and ground truth $Y$ is used. The parameters $W^{(i)}$, $\tilde{W}^{(i)}$, $h^{(i)}$ and $b^{(i)}$, $i = 1, 2$, in the linear layer and RNN parts are learned by back-propagation.

Figure 4.2 Classification accuracy versus number of parameters. The blue color curves are based on ResNet architecture (left axis), while the orange curves are based on PlainNet architecture (right axis). The solid line denotes DeTraMe-Net, while the dash-line denotes the original CNNs. ‘*’ denotes for CIFAR10, ‘o’ denotes for CIFAR100 and ‘+’ denotes for SVHN.

Figure D.1 Architectures of PlainNet vs. DeTraMe-PlainNet
Figure D.2 Architectures of ResNet vs. DeTraMe-ResNet
Much like in the endeavor of understanding and interpreting the complex and chaotic world, humans gradually form an ability to abstract the common descriptions of different observations and phenomena over time, one proceeds to extract common characteristics and attributes of data and to glean and gel various concepts. Such an extraction/abstraction exercise is indeed to navigate and distinguish experiences, and thereby humans can summarize and generalize the relationships among objects, events and phenomena by way of differentiating the principal patterns behind.

As classification may help a human in understanding the chaotic world after sensing an image of the real world, it is also a basic, critical and essential inference task in Machine Learning and Artificial Intelligence areas to make a machine understand and learn the world when provided collected image data. The classification task not only provides a basic understanding of objects and concepts, but it also is a sound foundation for building induction and inference tasks, such as, object detection, image segmentation, image caption, video summary and others, all of which are also important in Data Mining applications.

In Machine Learning, the classification task aims to assign a class to an unlabeled object. A learning algorithm is hence used to acquire the knowledge on the basis of the labeled objects and to then generalize to unlabeled ones. Somewhat similar (rough approximation) to human learning procures, a basic classification task is to acquire the skills of recognizing an object that they have seen before. After human beings acquire the know-how they demonstrate their expertise by recognizing objects they have never seen, i.e. a capacity to generalize. A advanced classification thus turns into transferring the knowledge of known objects to recognizing the unknown ones. Based on whether the unlabeled object is seen/known or unseen/unknown during the learning/training phase, the classification tasks in Machine Learning are divided into two taxonomies, which are respectively
named as Classification and Zero-shot Classification.

In traditional Classification tasks, many classical methods have appeared in the literature. As early as the 1950's, the proposed Bayesian classifier has been followed by Logistic Regression [Cox58], Perceptron [Ros58] and K-Nearest Neighbours [Cov67]. In the 1980s, three main classical algorithms, Decision Tree [Qui86], Support Vector Machine [Cor95] and AdaBoost [Fre95] these methods have been dominant and pervaded in machine learning applications, until computational potential has allowed Deep Learning to demonstrate its powerful potential.

In the same period, feature selection was also one of the earliest research topics in Machine Learning, because more discriminative and distinct features yield better classification performance. Early on, researchers selected the features first, called hand-craft features, used as inputs to a classifier. Towards a more comprehensive characterization of problems at hand, a join learning of features and their subsequent exploitation in classification was proposed, making such an integrated approach highly popular when class labels are given, i.e. as Supervised Learning. In Deep learning, such integrated method is referred to as end-to-end modeling.

As discussed in "In a Study of Thinking" [Aus62] in Psychology, structures of an integrated method/end-to-end model is central in inference tasks, as established in humans who in some way, define criteria for equivalence classes to simplify and facilitate discrimination among classes of objects. This inspiration in combination with our belief that humans are unconsciously always in an optimizing mode, we explore in this thesis, different optimized structures for various classification inference problems.

In Chapter 2, we attempt to use a integrated method to jointly select features and to learn a classifier for solving a Classification task for small datasets. Since the size of datasets is not very large, the classical machine learning algorithm is sufficient to achieve a human-like performance. A very popular integrated method in classical Machine Learning area is the Dictionary Learning /Sparse Representation, whose great success has been demonstrated in image denoising [Ela06], image restoration [Xu16], audio processing [Gro07] and image classification [Mai09c].

To seek a high-performance and efficient solution, a discriminative structured analysis dictionary is proposed for the classification task. In order to explore the intrinsic structure of the invariant low-dimensional spaces, a structure of the union of subspaces (UoS) is also integrated into the conventional analysis dictionary learning to enhance the capability of discrimination. A simple classifier is also simultaneously included into the formulated function to ensure a more complete and consistent classification. The solution of the algorithm is efficiently obtained by the linearized alternating direction method of multipliers. Moreover, a distributed structured analysis dictionary learning is also presented to address large scale datasets. It can group-(class-) independently train the structured analysis dictionaries by different machines/cores/threads, and therefore avoid a high computational cost. A consensus structured analysis dictionary and a global classifier are jointly learned using a distributed approach to safeguard the discriminative power and the efficiency of classification. Experiments demonstrate that our method achieves a comparable or better performance than the state-of-the-art algorithms in a variety of visual classification tasks. In addition, the
training and testing computational complexity are also greatly reduced.

In recent years, Zero-shot Classification has also increasingly been of great interest in the Machine Learning area, and has registered some great successes, in the recognition task of unseen object classes by only training on seen object classes. In order to transfer the finite and limited knowledge that is learned from seen dataset, and to generalize to the unseen one, leveraging the common underlying invariant structure between the seen and unseen data, plays an important role in such Zero-Shot Classification inference.

Thus, upon addressing the Classification task, in Chapter 3, we begin to optimize the learning structure in the Zero-Shot Classification task. Most existing Zero-Shot Learning (ZSL) methods [Soc13; Lam14; Sch13; Lam14; RP15; Shi15; Cha16a; Zha16c; Cha16b; Aka16; Xia17] typically learn a projection map between the visual feature space and the semantic space and broadly prone to the so-called projection domain shift, which is primarily due to a large domain gap between seen and unseen classes. In Chapter 3, we propose a novel inductive ZSL model which is based on projecting both visual and semantic features into a common distinct latent space with class-specific knowledge, and on reconstructing both visual and semantic features by such a distinct common space to narrow the domain shift gap. We show that all these constraints in the latent space, class-specific knowledge, reconstruction of features and their combinations enhance the robustness against the projection domain shift problem, and improve the generalization ability to unseen object classes. Comprehensive experiments on four benchmark datasets demonstrate that our proposed method is superior to state-of-the-art algorithms.

With the dramatically increasing size of data and developments of the hardware capability, such as CPU, memory and GPU, Neural Networks have recovered from their less glamorous 80's and 90's days in Machine learning area, with the advent Backpropagation [Hin85], Convolutional Neural Networks (CNN) [LeC89] and Long Short-Term Memory Network (LSTM) [Hoc97]. Its real rise coincided with the AlexNet [Kri12], a deep CNN designed in 2012, which achieved the best results in the challenging Classification task of ImageNet, with its 1.3 millions training images. And now this area is more specifically called Deep Learning. To manipulate such a tremendous dataset, the structures of such an integrated method/end-to-end model for Classification is still a critical factor to the proper discrimination of the method, allowing a better focus on selecting the distinct class features. Hence, various structures and architectures of CNNs have exploded to improve the performance of the Classification task, such as GoogleNet [Sze15], VGG [Sim14], ResNet [He16a], DenseNet [Hua17], ResNeXt [Xie17], SENet [Hu18] and so on.

In Chapter 4, we co-opt the framework of neural networks, especially, CNNs, and thereby cope with the Classification task with large amount of data using a principled optimized structure. Although the classical machine learning algorithms still work, their performances are not as expected. The main issue is that the hand-craft selection of features or those features learned by shallow structured methods can not distinctly characterize each class. A principled, powerful, efficient and end-to-end learning algorithm is therefore desired. Since Dictionary Learning (DL) has successfully been successful in the shallow integrated method to select features and to learn the classifier simul-
taneously, many Deep Dictionary Learning (DDL) methods are developed to stack the shallow DLs to form a deep one. However, DL suffers from its high delay costs due to its $L_1$ regularization term. Although many optimization solvers and differentiable programming methods have been developed to speed up the $L_1$ norm regularization, such as FISTA [Bec09], LISTA [Gre10] and SC2Net [Zho18], all these methods only solve a single layer dictionary learning, without safeguarding the discrimination of the classifier, and are never used to solve DDL methods. To improve the discrimination and yet secure the efficiency of DDL, a novel differentiable programming method is proposed in Chapter 4, where we transform a single-layer DL into a combination of a linear layer and a Recurrent Neural Network (RNN), and stack them into a deep network to solve DDL. The resulting RNN part can be flexibly built into any network architecture, and combined with a linear layer to become the DL layer. To the best of our knowledge, this is the first paper to connect DDL with the combined linear layers and RNNs, and thereby bring a novel insight in understanding the relationship between neural networks and DDL. Extensive experiments demonstrate that our method not only achieves a better performance than the original DDL, but also obtains a superior performance than the state-of-the-art CNNs.
2.1 Introduction

Sparse representation has had great success in dealing with various problems in image processing and computer vision, such as image denoising and image restoration. To obtain such sparse representations with an unknown precise model, Dictionary Learning is one choice because it results in a linear combination of sparse dictionary atoms. There are two different types of dictionary learning methods: Synthesis Dictionary Learning (SDL) and Analysis Dictionary Learning (ADL). In recent years, SDL has been prevalently and widely studied \cite{mahdizadehaghdam2018deep, Ols96, Bru09}, while ADL has received little attention.

SDL supposes that a signal lies in a sparse latent subspace and can be recovered by an associated dictionary. The local structures of the signal are well preserved in the optimal synthesis dictionary \cite{Mai08, Mai09a, Mai09b}. In contrast, ADL assumes that a signal can be transformed into a latent sparse subspace by its corresponding dictionary. In other words, ADL is to produce a sparse representation by applying the dictionary as a transform to a signal. The atoms in an analysis dictionary can be interpreted as local filters, as first mentioned in \cite{Rot09}. Sparse representations can be simply obtained by an inner product operation, when the dictionary is known. Such a fast coding supports ADL more favored than SDL in applications. The contrast of SDL and ADL is shown in Fig. 2.1.
Figure 2.1 SDL reconstructs data $X$ by the dictionary $D$ with the sparse representations $A$. ADL applies the dictionary $\Omega$ to data $X$ and results in the sparse representations $U$. $\| \cdot \|_p$ can be either $l_1$ norm or $l_0$ norm. If and only if $D$ and $\Omega$ are square matrices, SDL and ADL are equivalent to each other.

The success of dictionary learning in image processing problems has shaped much interest in task-driven dictionary learning methods for inference applications, such as image classification. The task of classification aims to assign the correct label to an observed image, which requires a much more discriminative capacity of either the dictionary or the sparse representation. Towards addressing this issue, supervised learning is often invoked when using single-view SDL [Mai09c] so as to maximize the distances between the sparse representations of each of two distinct classes. In addition, multi-view dictionary learning methods [Liu19; Jin14; Bah16; Wan16] were developed to include more information of each class.

For the supervised single-view dictionary learning methods, there are generally two strategies to address the supervised learning approaches. The first strategy is to learn multiple dictionaries or class-specific dictionaries for different classes [Wri09; Ram10; Yan11a; Wan13]. The advantage of learning multiple dictionaries is that these dictionaries characterize specific patterns and structures of each class and enhance the distances between different classes. The minimum reconstruction errors of various dictionaries are subsequently used to assign labels of new incoming images. In [Ram10], Ramirez et al. learned class-specific dictionaries with penalty for the common atoms. Yang et al. [Yan11a] then learned class-specific dictionaries and jointly applied a Fisher criterion to associative sparse representations to thereby enhance the distances between each class. A large-margin method was proposed to increase the divergence of sparse representations for the class-
specific dictionaries in [Wan13]. However, as the number of classes increases, it would be too complex and time consuming to train class-specific dictionaries with regularizing distances of each dictionary. Even though a distributed cluster could reduce the time complexity of training dictionaries, it is difficult for the distributed algorithm to communicate with each independent cluster and to compromise with other regularizations for the class-specific dictionary learning.

Another strategy is to learn a shared dictionary for all classes together with a universal classifier [Mai09c; Jia13a]. Such a joint dictionary learning enforces more discriminative sparse representations. Compared with class-specific dictionary learning, using this strategy is simpler to learn such a dictionary and classifier, and easier to test the unknown images. In [Mai09c], Mairal et al. integrated a linear classifier in a sparse representation for a dictionary learning phase. Jiang et al. then included a linear classifier and a label consistent regularization term to enforce more consistent sparse representations in each class [Jia13a]. When any large data sets are on hand, memory and computational limitations emerge, and an online learning or distributed solutions are required as a viable strategy.

Although the techniques mentioned above are all based on SDL, ADL has gradually received more attention [Nam13]. Based on the seminal work of Rubinstein et al. [Rub13a] proposing analysis K-SVD to learn an analysis dictionary, Li et al. [Li14] considered to learn ADL by using an additional inner product term of sparse coefficients to increase its discriminative power. In addition, reducing the computational complexity has been addressed in recent methods. Zhang et al. [Zha14] use Recursive Least Square method to accelerate dictionary learning by updating a dictionary based on the dictionaries in the previous iterations. Li et al. [Li15] and Dong et al. [Don16] proposed Simultaneous codeword optimization (SimCo) related algorithms to update multiple atoms in each iteration and by adding an extra incoherent regularity term to avoid linear dependency among dictionary atoms. On other hand, Li et al. [Li16; Li18] used $l_{1/2}$ norm instead of $l_1$ norm to have stronger sparsity and mathematically guaranteed a strong convergence. In [bian2016sparsity], Bian et al. proposed the Sparse Null Space (SNS) pursuit to search for the optimal analysis dictionary. However, all of these methods are proposed for the original problem of learning an analysis dictionary. To the best of our knowledge, few attempts have been carried out for task-driven ADL. For example, in [She14], Shekhar et al. [She14] learned an analysis dictionary and subsequently trained SVM for the digital and face recognition tasks. Their results demonstrate that ADL is more stable than SDL under noise and occlusion, and achieves a competitive performance. Guo et al. [Guo16] integrated local topological structures and discriminative sparse labels into the ADL and separately classified images by a $k$ Nearest Neighbor classifier. Instead of preforming ADL and SVM in separate steps, Wang [Wan17] alternately optimize ADL and SVM together to classify different patterns. In [Wan18b], Wang et al. use the K-SVD based technique to solve a joint learning of ADL and a linear classifier. Additionally, a hybrid design based on both SDL and ADL is considered in [Gu14] and [Zha17b]. A multi-view ADL was proposed in [Wan18a], which separately learns analysis dictionaries for different views and a marginalized classifier for fusing the semantic information of each view.
Inspired by these past works, and taking advantage of efficient coding by ADL, we propose a supervised ADL with a shared dictionary and a universal classifier. In addition to the classifier, a structured subspace regularization is also included into an ADL model to obtain a more structured discriminative and efficient approach to image classification. We refer to this approach as Structured Analysis Dictionary Learning (SADL). Since Sparse Subspace Clustering [Elh13] has shown that visual data in a class or category can be well captured and localized by a low dimensional subspace, and the sparse representation of the data within a class similarly share a low dimensional subspace, a structured representation is introduced to achieve a distinct representation of each class. This achieves more coherence for within-class sparse representations and more disparity for between-class representations. When sorted by the order of classes, these representations as shown later can be viewed as a block-diagonal matrix. For robustness of the sought sparse representations, we simultaneously learn a one-against-all regression-based classifier. The resulting optimization function is solved by a linearized alternative direction method (ADM) [Lin11]. This approach leads to a more computationally efficient solution than that of analysis K-SVD [Rub13a] and of SNS pursuit [bian2016sparsity]. Additionally, a great advantage of our algorithm is its extremely short on-line encoding and classification time for an incoming observed image. It is easy to understand that in contrast to the SDL encoding procedure, ADL obtains a sparse representation by a simple matrix multiplication of the learned dictionary and testing data. Experiments demonstrate that our method achieves an overall better performance than the synthesis dictionary approach. A good accuracy is achieved in the scene and object classification with a simple classifier, and at a remarkably low computational complexity to seek the best performances of facial recognition problems. Moreover, experiments also show that our approach has a more stable performance than that of SDL. Even when the dictionary size is reduced to result in memory demand reduction, our performance is still outstanding. To address large datasets, a distributed structured analysis dictionary learning algorithm is also developed while preserving the same properties as those of structured analysis dictionary learning (SADL). Experiments also show that when the dataset is sufficient, a distributed algorithm achieves as high a performance as SADL.

The following represent our main contributions,

- Both a structured representation and a classification error regularization term are introduced to the conventional ADL formulation to improve classification results. A multiclass classifier and an analysis dictionary are jointly learned.
- The optimal solution provided by the linearized ADM is significantly faster than other existing techniques for non-convex and non-smooth optimization.
- An extremely short classification time is offered by our algorithms, as they entail encoding by a mere matrix multiplication for a simple classification procedure.
- A distributed structured analysis dictionary learning algorithm is also presented.

The balance of this paper is organized as follows: we state and formulate the problem of SADL and its distributed form in Section 2.2. The resulting solutions to the optimization problems along
with the classification procedure are described in Sections 2.3 and 2.4. In Section 2.5, we analyze the convergence and complexity of our methods. The experimental comprehensively validation and results are then presented in Section 4.5. Some comments and future works are finally provided in Section 2.7.

2.2 Structured Analysis Dictionary Learning

2.2.1 Notation

Uppercase and lowercase letters respectively denote matrices and vectors throughout the paper. The transpose and inverse of matrices are represented as the superscripts $T$ and $^{-1}$, such as $A^T$ and $A^{-1}$. The identity matrix and all-zero matrix are respectively denoted as $I$ and $0$. $(a_i)_{j}$ represents the $j$th element in the $i$th column of matrix $A$.

2.2.2 Structured ADL Method

2.2.2.1 ADL Formulation

The conventional ADL problem [Rub13a] aims at obtaining a representation frame $\Omega$ with a sparse coefficient set $U$ based on the data matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{m \times n}$.

$$\arg\min_{\Omega, U} \frac{1}{2} \|U - \Omega X\|^2 + \lambda_1 \|U\|_1 \quad \text{s.t. } \Omega \in \mathbb{R}^{r \times m} \subset W,$$

(2.1)

where $U \in \mathbb{R}^{r \times n}$ and $W$ is a large class of non-trivial solutions.

2.2.2.2 Mitigating Inter-Class Feature Interference

The basic idea of our algorithm is to take advantage of the stability to perturbations and of the fast encoding of ADL. Since there is no reconstruction term in the conventional ADL, and to secure an efficient classification, the representation $U$ is used to obtain a classifier in a supervised learning mode. To strengthen the discriminative power of ADL, it is desirable to minimize the impacts of inter-class common features. We therefore propose two additional constraints on $U$ by way of:

- Minimizing interference of inter-class common features by a structural map of $U$.
- Minimizing the classification error.

2.2.2.2.1 Structural Mapping of $U$

The first constraint is to particularly ensure that the representation of each sample in the same class belong to a subspace defined by a span of the associated coefficients. This imposes the distinction among the classes and improves the identification of each class, and efficiently enhances the
divergence between classes. Specifically, we introduce a block-diagonal matrix $H \in \mathbb{R}^{s \times n}$ as shown below,

$$H = \begin{pmatrix}
    h_1^1 & h_2^1 & h_3^1 & h_4^2 & h_5^2 & h_6^3 & h_7^3 \\
    1 & 1 & 1 & 0 & 0 & 0 & 0 \\
    1 & 1 & 1 & 0 & 0 & 0 & 0 \\
    1 & 1 & 1 & 0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 1 & 1 & 0 & 0 \\
    0 & 0 & 0 & 1 & 1 & 0 & 0 \\
    0 & 0 & 0 & 0 & 1 & 1 & 0 \\
    0 & 0 & 0 & 0 & 1 & 1 \\
\end{pmatrix},$$

where $s \geq n$ is the length of the structured representation. Each diagonal block in $H$ represents a subspace of each class to force each one class to remain distinct from another with a consistent intra-class representation. Each column $h_i^j$ is a structured representation for the corresponding data point, which is pre-defined on the training labels. $H$ is not necessarily a uniformly block-diagonal matrix, and the order of samples is not important, so long as the structured representation corresponds to a given class. To mildly relax the constraint, and integrate it into the ADL function, we write

$$H = QU + \epsilon_1, \quad (2.2)$$

where $Q \in \mathbb{R}^{s \times r}$ is a matrix to be learned with $\Omega$ and $U$, $\epsilon_1$ is the tolerance.

### 2.2.2.2 Minimal Classification Error

To maintain an audit track on the desired representation, we include a classification error to make the representation $QU$ discriminative and learn an optimal regularization. This is written as

$$Y = W(QU) + \epsilon_2, \quad (2.3)$$

where $\epsilon_2$ is the tolerance, $W \in \mathbb{R}^{c \times s}$ is a linear transform, and the label matrix $Y \in \mathbb{R}^{c \times n}$ is defined as

$$Y_{ij} = \begin{cases}
    1 & \text{if image } j \text{ belongs to class } i \\
    0 & \text{otherwise}
\end{cases},$$

and $c$ is the number of classes.
2.2.2.3 Structured ADL Formulation

To account for all these constraints and to avoid overfitting by $l_2$ regularization arising $\Omega$, $Q$ and $W$, we can rewrite the ADL optimization problem as

$$
\arg\min_{\Omega,U,Q,W,\epsilon_1,\epsilon_2} \frac{1}{2} \|U - \Omega X\|_F^2 + \lambda_1 \|U\|_1 + \rho_1 \frac{1}{2} \|\epsilon_1\|_2^2 + \rho_2 \frac{1}{2} \|\epsilon_2\|_2^2 + \delta_1 \frac{1}{2} \|Q\|_2^2 + \delta_2 \frac{1}{2} \|W\|_2^2 + \lambda_2 \|\Omega\|_2^2
$$

s.t. $H = QU + \epsilon_1,$

$Y = W(QU) + \epsilon_2,$

(2.4)

where $\rho_1$ and $\rho_2$ are the penalty coefficients, $\delta_1, \delta_2, \lambda_1$ and $\lambda_2$ are tuning parameters. Recall $H$ is the structured representation, $Q$ is the structuring transformation, $Y$ is the classifier label, and $W$ is the linear classifier.

The formulated optimization function in Eq. (2.4) provides an analysis dictionary driven by the latent structure of the data yielding an improved discriminative sparse representation among numerous classes.

2.2.2.4 Distributed Structured ADL Formulation

In order to handle large datasets, we propose a distributed Structured ADL method. Since both the discriminative structure and the efficient classification need to be preserved, we introduce a global analysis dictionary, a global structuring transformation and a global classifier. In pursuing a distributed ADL, we ensure that the global variables share information with each distributed dictionary cluster, thereby ensuring that the global analysis dictionary, the structured transform and the classifier respectively reach a consensus,

$$
\|\Omega - \Omega_t\|^2, \|Q - Q_t\|^2, \|W - W_t\|^2, \forall t = 1, \ldots, N.
$$

(2.5)
Together with the consensus penalties, the distributed SADL is formulated as

\[
\arg\min_{\Omega_t, U_t, \Omega, W, \epsilon_1, \epsilon_2, t} \sum_{t=1}^{N} \left( \frac{1}{2} \|U_t - \Omega_t X_t\|_F^2 + \lambda_1 \|U_t\|_1 + \frac{\rho_1}{2} \|\epsilon_1\|_2^2 \right.

\left. + \frac{\rho_2}{2} \|\epsilon_2\|_2^2 + \frac{\xi_1}{2} \|\Omega - \Omega_t\|_2^2 + \frac{\delta_1}{2} \|Q_t\|_2^2 + \frac{\lambda_2}{2} \|\Omega_t\|_2^2 \right)

s.t. H_t = Q_t U_t + \epsilon_1,

Y_t = W_t (Q_t U_t) + \epsilon_2,

\|\omega_i\|_2^2 = 1; \forall i = 1, \ldots, r,

\|\omega_i\|_2^2 = 1; \forall i = 1, \ldots, r, \forall t = 1, \ldots, N,

\] (2.6)

where \( t \) represents the \( t \)th independent cluster, \( \Omega_t, U_t, Q_t \) and \( W_t \) are respectively the local analysis dictionary, sparse representation, structuring transformation and classifier of the \( t \)th cluster, and \( \Omega, Q, W \) are respectively the global analysis dictionary, structuring transformation and classifier. The global variables will be applied to the same efficient classification scheme as the one of SADL.

### 2.3 Algorithmic Solution

#### 2.3.1 SADL Algorithm

Due to the non-convexity of the objective function in Eq. (2.4), an augmented Lagrange formulation with dual variables \( Z^{(1)}, Z^{(2)} \) and \( \mu \) is adopted to seek an optimal solution. The augmented Lagrangian is then written as,

\[
L(\Omega, U, Q, W, Z^{(1)}, Z^{(2)}, \mu) = \frac{1}{2} \|U - \Omega X\|_F^2 + \lambda_1 \|U\|_1

+ \langle Z^{(1)}, H - QU - \epsilon_1 \rangle + \langle Z^{(2)}, Y - W(QU) - \epsilon_2 \rangle

+ \frac{\mu}{2} \|H - QU - \epsilon_1\|_2^2 + \frac{\mu}{2} \|Y - W(QU) - \epsilon_2\|_2^2

+ \frac{\rho_1}{2} \|\epsilon_1\|_2^2 + \frac{\rho_2}{2} \|\epsilon_2\|_2^2 + \frac{\delta_1}{2} \|Q\|_2^2 + \frac{\delta_2}{2} \|W\|_2^2 + \frac{\lambda_2}{2} \|\Omega\|_2^2,
\] (2.7)

where \( \lambda_1 > 0 \) is a tuning parameter. To iteratively seek the optimal solution in Eq. (2.7), the analysis dictionary \( \Omega \) and two linear transformations \( Q \) and \( W \) are first randomly initialized. The sparse representation \( U \) is initialized as \( U = 0 \), the zero matrix. In the following equations, Eq. (2.8) - Eq. (2.22), the auxiliary variables \( \eta_U, \eta_Q, \) and \( \eta_W \) are introduced to guarantee the convergence of the algorithm. The variables with superscripts which do not include parenthesis are the temporal variables of intermediate step in the calculation. Different variables are alternately updated while fixing the others, resulting in the following steps:
(1) Fix $\Omega, Q, W,$ and $\epsilon_1, \epsilon_2,$ and update $U$:

$$U_{k+1} = \tau_{\frac{\mu}{\mu+\lambda}} (U_k - \frac{U_k^1 + U_k^2 + U_k^3}{\mu \eta_U}), \quad (2.8)$$

where $\tau(\cdot)$ is the element-wise soft thresholding operator, and $U_k^1, U_k^2,$ and $U_k^3$ are as follows:

$$U_k^1 = -(\Omega_k X - U_k), \quad (2.9)$$

$$U_k^2 = -Q_k^T (Z_k^{(1)} + \mu (H - Q_k U_k - \epsilon_1)), \quad (2.10)$$

$$U_k^3 = -Q_k^T W_k^T (Z_k^{(2)} + \mu (Y - W_k Q_k U_k - \epsilon_2)). \quad (2.11)$$

(2) Fix $\Omega, U, W,$ and $\epsilon_1, \epsilon_2,$ and update $Q$:

$$Q_{k+1} = Q_k - \frac{Q_k^1 + Q_k^2}{\mu \eta_Q}, \quad (2.12)$$

$$Q_k^1 = -(Z_k^{(1)} + \mu (H - Q_k U_{k+1} - \epsilon_{1k}))/U_{k+1} + \delta_1 Q_k, \quad (2.13)$$

$$Q_k^2 = -W_k^T (Z_k^{(2)} + \mu (Y - W_k Q_k U_{k+1} - \epsilon_{2k}))/U_{k+1}^T. \quad (2.14)$$

(3) Fix $\Omega, U, Q,$ and $\epsilon_1, \epsilon_2,$ and update $W$:

$$W_{k+1} = W_k - \frac{W_k^1}{\mu \eta_W}, \quad (2.15)$$

$$W_k^1 = - (Z_k^{(2)} + \mu (Y - W_k Q_{k+1} U_{k+1} - \epsilon_{2k}))/U_{k+1}^T Q_{k+1}^T + \delta_2 W_k. \quad (2.16)$$

(4) Fix $U, Q, W,$ and $\epsilon_1, \epsilon_2,$ and update $\Omega$:

$$\Omega_{k+1}^* = \arg \min_\Omega \frac{1}{2} \|U_{k+1} - \Omega X\|^2_F. \quad (2.17)$$

The analytical solution of Eq. (2.17) can be regularized as

$$\Omega_{k+1} = U_{k+1} X^T (X X^T + \lambda_2 I)^{-1}, \quad (2.18)$$

where $\lambda_2$ is also a tuning parameter. It will be chosen by a usual way.

(5) Fix $U, \Omega, Q,$ and $W,$ and $\epsilon_2,$ and update $\epsilon_1$:

$$\epsilon_{1k+1} = \frac{1}{\rho_1 - 1}(Z_k^{(1)} + \mu (H - Q_{k+1} U_{k+1})). \quad (2.19)$$

(6) Fix $U, \Omega, Q,$ and $\epsilon_1,$ and update $\epsilon_2$:

$$\epsilon_{2k+1} = \frac{1}{\rho_2 - 1}(Z_k^{(2)} + \mu (Y - W_{k+1} Q_{k+1} U_{k+1})). \quad (2.20)$$
The dual variables $Z^{(1)}$, $Z^{(2)}$ are updated as

$$Z^{(1)}_{k+1} = Z^{(1)}_k + \mu(H - Q_{k+1}U_{k+1}), \quad \text{(2.21)}$$

$$Z^{(2)}_{k+1} = Z^{(2)}_k + \mu(Y - W_{k+1}Q_{k+1}U_{k+1}). \quad \text{(2.22)}$$

In contrast to previous ADL techniques, which train a dictionary by iterating a single row of the dictionary, *i.e.*, one atom, to avoid a trivial solution, we proceed to update a set of rows in a single step at each iteration. A fast convergence rate of the algorithm is also guaranteed by linearized ADM [Lin11] and with a closed form solution for the dictionary $\Omega$ given in Eq. (2.18). The proposed SADL algorithm \(^1\) is summarized in Algorithm 1.

---

**Algorithm 1** Structured Analysis Dictionary Learning

**Input:** Training data $X = [x_1, \ldots, x_n]$, diagonal block matrix $H$, class labels $Y$, penalty coefficients $\rho_1, \rho_2, \delta_1, \delta_2$, parameters $\lambda_1, \lambda_2$ and maximum iteration $p$;

**Output:** Analysis dictionary $\Omega$, sparse representation $U$, and linear transformations $Q$ and $W$;

1. Initialize $\Omega$, $Q$, and $W$ as random matrices, and initialize $U$ as a zero matrix;
2. **while** not converged **and** $k < p$ **do**
   3. \hspace{1em} $k = k + 1$;
   4. \hspace{1em} Update $U_k$ by (2.8);
   5. \hspace{1em} Update $Q_k$ by (2.12);
   6. \hspace{1em} Update $W_k$ by (2.15);
   7. \hspace{1em} Update $\Omega_k$ by (2.18);
   8. \hspace{1em} Update $\varepsilon_{1_k}$ by (2.19);
   9. \hspace{1em} Update $\varepsilon_{2_k}$ by (2.20);
   10. \hspace{1em} Update $Z^{(1)}_k$ by (2.21);
   11. \hspace{1em} Update $Z^{(2)}_k$ by (2.22);
3. **end while**

\(^1\)The codes are released at https://github.com/wtan0512/Demo-of-SADL
2.3.2 Distributed SADL Algorithm

The distributed SADL is similarly expressed in the augmented Lagrangian function as

\[
L_d(\Omega_t, U_t, Q_t, W_t, \Omega, Q, W, Z^{(1)}, Z^{(2)}, \mu_k) =
\sum_{t=1}^{N} \frac{1}{2} ||U_t - \Omega_t X_t||_F^2 + \lambda_1 ||U_t||_1 + \frac{\delta_{1t}}{2} ||Q_t||_2^2 + \frac{\delta_{2t}}{2} ||W_t||_2^2
\]

\[
+ \frac{\xi_{1t}}{2} ||\Omega - \Omega_t||_2^2 + \frac{\xi_{2t}}{2} ||Q - Q_t||_2^2 + \frac{\xi_{3t}}{2} ||W - W_t||_2^2
\]

\[
+ \frac{\rho_{1t}}{2} ||\epsilon_{1t}||_2^2 + \frac{\rho_{2t}}{2} ||\epsilon_{2t}||_2^2 + \frac{\lambda_{1t}}{2} ||\Omega_t||_2^2
\]

\[
+ \langle Z^{(1)}_t, H_t - Q_t U_t - \epsilon_{1t} \rangle + \langle Z^{(2)}_t, Y_t - W_t (Q_t U_t) - \epsilon_{2t} \rangle
\]

\[
+ \frac{\mu_k}{2} ||H_t - Q_t U_t - \epsilon_{1t}||_2^2 + \frac{\mu_k}{2} ||Y_t - W_t (Q_t U_t) - \epsilon_{2t}||_2^2.
\]

(2.23)

To minimize such an objective function, each variable is alternatively updated while fixing others. The distributed SADL algorithm is presented in Algorithm 2.

2.4 Classification Procedure

Both SADL and Distributed SADL have the same classification procedure because the global analysis dictionary \(\Omega\), transforming matrix \(Q\) and classifier \(W\) are obtained from the algorithms. With the analysis dictionary \(\Omega\) in hand, an observed image \(x\) can be quickly sparsely encoded as \(\Omega x\). This is in stark contrast to SDL for which a sparse representation is obtained by solving a non-smooth optimization as: \(\arg\min_\alpha ||x - D\alpha||_2^2 + ||\alpha||_1\), and highlights the remarkable improvement ADL provides. Our proposed SADL, which naturally enjoys the same encoding properties as ADL, efficiently yields a structured sparse representation \(Q(\Omega x)\) of the signal \(x\) as well. Figure 2.2 shows an example of the structured representations obtained from Scene 15 dataset. As shown, the result reflects the desired
Algorithm 2 Distributed SADL

**Input:** Training data $X = [x_1, \ldots, x_n]$, diagonal block matrix $H$, class labels $L$, penalty coefficients $\delta_1, \delta_2, \xi_1, \xi_2, \xi_3$, parameters $\lambda_1, \lambda_2$ and maximum iteration $p$;  

**Output:** Analysis dictionary $\Omega$, linear transformations $Q$ and $W$;  
Initialize $\Omega$, $Q_k$, $W_k$, $Q$, and $W$ as random matrices, initialize $U_i$ as a zero matrix, and set $\{X_t\}$ as a randomly selected partition of $X$ with $\bigcup_{i=1}^N X_t = X$;  

2: while not converged and $k < p$ do  
   4: for $t = 1 : N$ do %Here for loop can be parallelized or distributed in different clusters.  
      6: $Q_t^{k+1} = Q_t^k - \nabla_{Q_t} \left( U_t^{k+1} W_t^k Q_t^{k} + Q_t^{k} W_t^k X_t^T \right)$;  
      8: $\Omega_t^{k+1} = \left( U_t^{k+1} X_t^T + \xi_1^k, \Omega_t^k (X_t X_t^T + (\xi_1^k + \lambda_2) I) \right)$;  
      10: $Y_k^{(1)} = Q_t^{k+1} + \mu (H - Q_0 U_k^0)$;  
      12: $\mu_{k+1} = \min \{ \mu, \mu_{\max} \}$;  
      14: $\xi_3^{k+1} = \min \{ \rho \xi_3^k, \xi_3^{k_{\max}} \}$;  
   16: end for  
   18: Normalize $\Omega_t^{k+1} by $ \omega_t^T = \frac{\Omega_t^T}{\| \omega_t^T \|_2}$, $\forall i$;  
   20: $W^{k+1} = \frac{1}{N} \sum_t W_t^{k+1}$;  
end while
block diagonal structure. The ultimate desired classification goal of $x$ is accomplished by $W(Q\Omega x)$. Figure 2.3 depicts $W(Q\Omega x)$ for the example in Figure 2.2 where the horizontal axis is image index, and the vertical axis reflects the class labels, which are computed according to,

$$y = \max_j(WQ\Omega x)_j,$$

shown as the brightest ones in Figure 2.3.

2.5 Convergence

Since we have used linearized ADM method to solve our nonconvex objective function, $\eta_U, \eta_Q, \eta_W$ are introduced as the auxiliary variables. We additionally have the following

**Theorem 1.** Suppose that $\mu \geq \sqrt{2}\{\rho_1, \rho_2\}$. There exist positive values $\eta^0_U, \eta^0_Q, \eta^0_W, R$ only depending on the initialization such that for $\eta_U > \eta^0_U, \eta_Q > \eta^0_Q, \eta_W > \eta^0_W$ the sequence

$$\{\Theta_k = (\Omega_k, U_k, Q_k, W_k, \epsilon^{(1)}_k, \epsilon^{(2)}_k, Z^{(1)}_k, Z^{(2)}_k)\}_{k=1}^{\infty}$$

converges to the following set of bounded feasible stationary points of the Lagrangian\(^2\):

$$S = \{\Theta = (\Omega, U, Q, W, \epsilon^{(1)}, \epsilon^{(2)}, Z^{(1)}, Z^{(2)}) |$$

$$\|\Theta\| < R, -\nabla L_s \in \lambda \partial \|U\|_1, H = QU + \epsilon^{(1)}, Y = QUW + \epsilon^{(2)}\}$$

where $L_s$ is the smooth part of $L$, i.e.,

$$L = L_s + \lambda_1 \|U\|_1.$$

\(^2\)The norm $\|\Theta\|$ is any norm that is continuous with respect to the two norm of the components, for example their some of two norms. Also, the function $\|U\|_1$ is treated as a (convex) function of $\Theta$, which is constant with respect to other components than $U$. 

Figure 2.3 $WQ(\Omega x)$ on Scene 15 Dataset
According to Theorem 1, if we initialize $\eta_U, \eta_Q, \eta_W$ large enough, Algorithm 1 not only converges, but also generates the variable sequences with a final convergence to the stationary points. The proof of Theorem 1 can be found in Appendix A\(^3\).

### 2.6 Experiments and Results

We now evaluate our proposed SADL method on five popular visual classification datasets that have been widely used in previous works and with known performance benchmarks. They include Extended YaleB face dataset [Geo01], AR face dataset [Marne], Caltech 101 object categorization dataset [FF07], Caltech 256 objective dataset [Gri07], and Scene 15 scene image dataset [Laz06].

In our experiments, we provide a comparative evaluation of six state-of-the-art techniques and our proposed technique, including a classification accuracy as well as training and testing times. All our experiments and competing algorithms are implemented in Matlab 2015b on the server with 2.30GHz Intel(R) Xeon(R) CPU. For a fair comparison, we measure the performance of each algorithm by repeating the experiment over 10 realizations. The testing time is defined as the average processing time to classify a single image. In our tables, the accuracy in parentheses with the associated citation is that was reported in the original paper. The difference in the accuracy of our approach and of the original one might be caused by different segmentations of the training and testing samples.

#### 2.6.1 Parameter Settings

In our proposed SADL method, $\lambda_1, \lambda_2$ and maximum iteration $p$ are tuning parameters. The parameter $\lambda_1$ controls the contribution of the sparsity, and the parameter $\lambda_2$ controls the learned analysis dictionary, while $p$ is the maximum iteration number. We replace $\varepsilon_1$ and $\varepsilon_2$ by their expressions, and insert them in the optimization formula. We choose for all the experiments $\lambda_1, \lambda_2, p$ and dictionary size by 10-fold cross validation on each dataset. In addition, we also optimally tuned the parameters of all competing methods to ensure their best performance.

#### 2.6.2 State-of-the-art Methods

We compare our proposed SADL and Distributed SADL (DSADL) with the following competing techniques: The first one is a baseline, which uses the ADL method to learn a sparse representation and subsequently trains a Support Vector Machine (SVM) to classify images based on such sparse representations (ADL+SVM) [She14]. A penalty term is included to avoid similar atoms and minimize false positives. The second one is the classical Sparse Representation based Classification (SRC) [Wri09]. For this method, we do not need to train a dictionary. Instead, we use the training images as the atoms in the dictionary. In the testing phase, we obtain the sparse coefficients based on such a dictionary. The third technique that we consider in this work is a state-of-the-art dictionary learning

\[\text{You can access this at http://ieeexplore.ieee.org.}\]
method, called Label Consistent K-SVD (LC-KSVD) [Jia13a], which forces each category labels to be consistent with classification. We select the LC-KSVD2 in [Jia13a] for comparison, because it has a better classification performance. The fourth method is Discriminative Analysis Dictionary Learning (DADL) [Guo16], which incorporates a topological structure and distinct class representations to the ADL framework in order to make each class discriminative. Then a 1-nearest-neighbor classifier is used to assign the label. The fifth technique, Class-aware Analysis Dictionary Learning (CADL) [Wan17], is to learn the class-specific analysis dictionaries and jointly learn a universal SVM based on the concatenated class-specific coefficients of each class. Finally, we compare our method with the Synthesis K-SVD based Analysis Dictionary Learning (SK-SVADL) [Wan18b], which is to jointly learn ADL and a linear classifier and is solved by the K-SVD method.

2.6.3 Extended YaleB

The Extended YaleB face dataset contains in total 2414 frontal face images of 38 persons under various illumination and expression conditions, as illustrated in Figure 2.4. Due to such illumination and expression variation, YaleB is intended to test the robustness to the intra-class variation. Each person has about 64 images, each cropped to 168 × 192 pixels. We project each face image onto an n-dimensional random face feature vector. The projection is performed by a randomly generated matrix with a zero mean normal distribution whose rows are $l_2$ normalized. This procedure is similar to the one in [Jia13a]. In our experiment, $n$ is 504, i.e., each Extended YaleB face image is reduced to a 504-dimensional feature vector. Then, we randomly choose half of the images for training, and the rest for testing. The dictionary size is set to 1216 atoms, $\lambda_1 = 0.001, \lambda_2 = 0.005$ and $p = 466$.

The classification results, training and testing times are summarized in Table 2.1. Although the accuracy of the SADL method is slightly lower than SRC, DADL and CADL, it is still comparable and higher than SK-SVADL, LC-KSVD and ADL+SVM. SADL is substantially more efficient than the others in terms of numerical complexity.

For a more thorough evaluation, we compare SADL with LC-KSVD, CADL and SK-SVADL for different dictionary sizes, and display the classification accuracy and training times in Figure 2.5 and 2.6, which are based on the average of ten realizations. We ran our experiments for dictionary sizes by the size of 38, 152, 266, 380, 494, 608, 722, 836, 950, 1064, 1178 and 1216 (all training size).
Table 2.1 Classification Results on Extended YaleB Dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Classification Accuracy(%)</th>
<th>Training Time(s)</th>
<th>Testing Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADL+ SVM[She14]</td>
<td>82.91%</td>
<td>91.78</td>
<td>1.13×10⁻³</td>
</tr>
<tr>
<td>SRC[Wri09]</td>
<td>96.51%</td>
<td>No Need</td>
<td>3.66×10⁻¹</td>
</tr>
<tr>
<td>LC-KSVD[Jia13a]</td>
<td>83.31% (96.7%[Jia13a])</td>
<td>123.07</td>
<td>1.60×10⁻²</td>
</tr>
<tr>
<td>DADL[Guo16]</td>
<td>97.35% (97.7%[Guo16])</td>
<td>10.05</td>
<td>4.55×10⁻⁵</td>
</tr>
<tr>
<td>CADL[Wan17]</td>
<td>97.05%</td>
<td>130.83</td>
<td>9.72×10⁻⁶</td>
</tr>
<tr>
<td>SK-SVDADL[Wan18b]</td>
<td>96.14% (96.9%[Wan18b])</td>
<td>113.78</td>
<td>1.34×10⁻⁴</td>
</tr>
<tr>
<td>SADL</td>
<td>96.35%</td>
<td>39.23</td>
<td>7.61×10⁻⁶</td>
</tr>
</tbody>
</table>

SADL, SK-SVDADL and CADL, the ADL methods, exhibit a more stable accuracy performance than that of LC-KSVD of the SDL methods. In particular, the accuracy of LC-KSVD significantly decreases, when the dictionary size approaches the training sample size. The significant decrease in accuracy may be caused by the trivial solution of dictionary $D$ in SDL. In addition, our method apparently has a much higher classification accuracy than LC-KSVD and a very similar accuracy as SK-SVDADL, when the dictionary size is small. As the dictionary size increases, SADL achieves a better accuracy than SK-SVDADL and approaches the accuracy of CADL. Although the accuracy of SADL is barely lower than CADL, the SADL method is also much faster than the LC-KSVD, SK-SVDADL and CADL in the training phase, especially when the dictionary size becomes larger.

Figure 2.5 Classification Accuracy versus Dictionary Size

2.6.4 AR Face

The AR Face dataset has 2600 color images of 50 females and 50 males with more facial variations than the Extended YaleB database, such as different illumination conditions, expressions and facial disguises. AR is also used to test the robustness to large intra-class variation. Each person has about
Figure 2.6 Training Time versus Dictionary Size

Figure 2.7 AR Dataset Examples

26 images of size 165 × 120. Figure 2.7 shows some sample images of faces with sunglasses or scarves. The features of the AR face image are extracted in the same way as those of the Extended YaleB face image are, but we project it to a 540 dimensional feature vector similarly to the setting in [Jia13a]. 20 images of each person are randomly selected as a training set and the other 6 images for testing. The dictionary size of the AR dataset is set to 2000 atoms, \( \lambda_1 = 0.001, \lambda_2 = 0.005 \) and \( p = 204 \).

The classification results as well as the training and testing times are summarized in Table 2.2. Comparing with other methods, our proposed SADL achieves a comparable result with the fastest training and testing time. The classification accuracy is lower than DADL, CADL and SK-SVDADL, and higher than SRC, LC-KSVD. However, our method is about 1000 times faster than SRC and LC-KSVD for the testing phase, 10 times faster than DADL and SK-SVDADL. Although SADL is only slightly faster than CADL, its training time is one-tenth of the one of CADL.

2.6.5 Caltech 101

The Caltech 101 dataset has 101 different categories of different objects and one non-object category. Most categories have around 50 images. Figure 2.8 gives some examples from the Caltech 101 dataset. Since this dataset is left-right aligned and rotated, Caltech 101 contains many different intra-class scaling variations, color pattern diversity and inter-class common features. We extract dense Scale-invariant Feature Transform (SIFT) descriptors for each image from 16 × 16 patches.
Table 2.2 Classification Results on AR Dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Classification Accuracy(%)</th>
<th>Training Time(s)</th>
<th>Testing Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADL+SVM[She14]</td>
<td>90.40%</td>
<td>218.54</td>
<td>9.10×10^{-3}</td>
</tr>
<tr>
<td>SRC[Wri09]</td>
<td>97.10%</td>
<td>No Need</td>
<td>7.41×10^{-1}</td>
</tr>
<tr>
<td>LC-KSVD[Jia13a]</td>
<td>87.78% (97.8%[Jia13a])</td>
<td>169.35</td>
<td>2.00×10^{-2}</td>
</tr>
<tr>
<td>DADL[Guo16]</td>
<td>98.32% (98.7%[Guo16])</td>
<td>47.76</td>
<td>2.68×10^{-4}</td>
</tr>
<tr>
<td>CADL[Wan17]</td>
<td>98.52% (98.8%[Wan17])</td>
<td>313.37</td>
<td>1.34×10^{-5}</td>
</tr>
<tr>
<td>SK-SVDADL[Wan18b]</td>
<td>97.38% (97.7%[Wan18b])</td>
<td>113.78</td>
<td>1.34×10^{-4}</td>
</tr>
<tr>
<td>SADL</td>
<td>97.17%</td>
<td>32.60</td>
<td>1.33×10^{-5}</td>
</tr>
</tbody>
</table>

Figure 2.8 Caltech 101 Dataset Examples

and with a 6 pixels step. Then, we apply a spatial pyramid method [Laz06] to the dense SIFT features with three segmentation sizes 1 × 1, 2 × 2, and 4 × 4 to capture the objects' features at different scales. At the same time, a 1024 size codebook is trained by k-means clustering for spatial pyramid features. Spatial pyramid features of each subregion are then concatenated together as a vector to represent one image. Due to the sparse nature of the spatial pyramid features, we use PCA to reduce each feature to 3000 dimensions. In our experiment, 30 images per class are randomly chosen as training data, and other images are used as testing data. All the steps and settings follow [Jia13a]. The dictionary size is set to 3060, λ1 = 0.001, λ2 = 1.5 and p = 827.

Table 2.3 Classification Results on Caltech 101 Dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Classification Accuracy(%)</th>
<th>Training Time(s)</th>
<th>Testing Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADL+SVM[She14]</td>
<td>66.75%</td>
<td>1943.47</td>
<td>1.33×10^{-2}</td>
</tr>
<tr>
<td>SRC[Wri09]</td>
<td>70.70%</td>
<td>No Need</td>
<td>4.34×10^{-1}</td>
</tr>
<tr>
<td>LC-KSVD[Jia13a]</td>
<td>73.67% (73.6%[Jia13a])</td>
<td>2144.90</td>
<td>2.49×10^{-3}</td>
</tr>
<tr>
<td>DADL[Guo16]</td>
<td>71.77% (74.6%[Guo16])</td>
<td>233.49</td>
<td>7.90×10^{-4}</td>
</tr>
<tr>
<td>CADL[Wan17]</td>
<td>76.83% (75%[Wan17])</td>
<td>9896.46</td>
<td>4.86×10^{-5}</td>
</tr>
<tr>
<td>SK-SVDADL[Wan18b]</td>
<td>73.39% (74.4%[Wan18b])</td>
<td>182.71</td>
<td>2.49×10^{-4}</td>
</tr>
<tr>
<td>SADL</td>
<td>74.45%</td>
<td>847.50</td>
<td>4.76×10^{-5}</td>
</tr>
<tr>
<td>DADL</td>
<td>73.49%</td>
<td>-</td>
<td>8.10×10^{-6}</td>
</tr>
</tbody>
</table>
The classification results, training and testing times are summarized in Table 2.3. Our proposed SADL achieves the second highest accuracy, while only costing one-tenth of the training time of CADL obtaining the maximum accuracy. SADL has again the shortest encoding time, which is around 10000 times faster than LC-KSVD and 10 times faster than DADL and SK-SVDADL. Note that the distributed ADL (DSADL) used only 510 atoms, but it still achieves a comparable result with the fastest testing time.

![Distributed SADL](image)

**Figure 2.9** Distributed SADL on Caltech 101: $N$ is the number of clusters used. $N = 1$ is centralized. Training set is divided into $N$ groups.

The parameters in DSADL are set as the following: $\lambda_1 = 0.001$, $\lambda_2 = 4.6$, $p = 1110$ and the penalty coefficients of the communication cost $\xi_1^t = \xi_2^t = \xi_3^t = 0.1$, $\forall t$. Figure 2.9 shows that when the number of groups is increased, the accuracy is actually lower at first because of the smaller training sample size of each independent variable. But after the communication between global variables and local independent variables are enhanced, the performance rises up very quickly to a high generalized accuracy. Distributed SADL is demonstrated that it can also obtain a very stable and excellent performance even when the number of groups is large.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Initial Variables</th>
<th>Total Training Time</th>
<th># Training Samples of Each Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Cluster</td>
<td>17.04</td>
<td>6471.9</td>
<td>3060</td>
</tr>
<tr>
<td>2 Clusters</td>
<td>2.62</td>
<td>3572.9</td>
<td>1530</td>
</tr>
<tr>
<td>4 Clusters</td>
<td>0.89</td>
<td>3235.5</td>
<td>765</td>
</tr>
<tr>
<td>6 Clusters</td>
<td>0.78</td>
<td>3194.0</td>
<td>510</td>
</tr>
<tr>
<td>10 Clusters</td>
<td>0.72</td>
<td>3148.6</td>
<td>306</td>
</tr>
</tbody>
</table>

**Table 2.4 Training Time and # Training Samples on Caltech 101 Dataset**

To further study the efficiency of distributed SADL, we conduct an experiment based on different
numbers of clusters, which is shown in Fig. 2.10. For fairness, we first utilize only one core in our CPU to run the SADL, while the 2-cluster experiment uses 2 cores to implement DSADL; 4-cluster experiment uses 4 cores on DSADL, and so on. The training time and the number of training samples of each cluster are averaged over 10 realizations and are listed in Table 2.4. It is worth noting that the training time in Table 2.3 is based on 28 cores (whole cores) in CPU, while the training time in Table 2.4 is based on only one core of the CPU. We separate the algorithm of DSADL into two parts: an initialization part and a variable updating part. The initialization part corresponds to the line 1 in Algorithm 2, and the variable updating part is started at line 2 to line 21, i.e., the while loop. The initialization part consists of simple matrix assignments, while the variable updating part has more matrix calculation, such as multiplication and inversion. It is shown in Fig. 2.10 that the running time of both the initialization part and the variable updating part quickly decrease when the numbers of clusters increase. The slopes of both curves decrease when more clusters are used, which is due to the fact that the training samples in each cluster is small enough to affect the calculation capability of each CPU core. As there are three global communication terms in Algorithm 2 after updating individual dictionaries, transforming matrix and classifier learnt, the training time with 2 clusters, is slightly more than the half the running time of 1 cluster (centralized). However, these three terms are not expensive, and Algorithm 2, with 2 clusters is still 1.8 times faster than the centralized one. We observed that the more clusters we use, the more training time is saved. Moreover, the larger data is, the more training time is also saved.

### 2.6.6 Caltech 256

The Caltech 256 is a relatively larger objective dataset, which includes 256 object categories and one clutter. There are totally 30608 images with various object location, pose, and size. Figure 2.11 shows examples of the Caltech 256 dataset, whose each category has at least 80 images. Note that...
Caltech 256 includes no rotation or alignment characteristics. Thus, it contains large intra-class diversity and inter-class similarity, such as object scale, object rotation and common patterns. The features of Caltech 256 images are extracted by using the output features of the last layer before fully connected layer of ResNet-50 [He16b] with the weights trained by ImageNet. The dimension of each feature is 2046 × 1. We randomly sample 15 images from each category for training, and test on the rest of them. To train the Distributed SADL, the dictionary size is set to 3855, dataset is divided into 3 subsets (i.e., \( t = 3 \) in Algorithm 2), \( \lambda_1 = 0.001, \lambda_2 = 0.5, \xi_1 = \xi_2 = \xi_3 = 3 \times 10^{-5}, \forall t \) and \( p = 4495 \).

<table>
<thead>
<tr>
<th>Methods (training samples)</th>
<th>Classification Accuracy(%)</th>
<th>Training Time(s)</th>
<th>Testing Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADL+SVM(15)[She14]</td>
<td>66.66%</td>
<td>3501.44</td>
<td>7.67 \times 10^{-2}</td>
</tr>
<tr>
<td>LC-KSVD(15)[Jia13a]</td>
<td>73.37%</td>
<td>3118.76</td>
<td>3.00 \times 10^{-3}</td>
</tr>
<tr>
<td>DADL[Guo16]</td>
<td>72.20%</td>
<td>417.06</td>
<td>5.42 \times 10^{-4}</td>
</tr>
<tr>
<td>CADL[Wan17]</td>
<td>75.25%</td>
<td>5586.21</td>
<td>4.83 \times 10^{-5}</td>
</tr>
<tr>
<td>SK-SVDADL[Wan18b]</td>
<td>73.35%</td>
<td>334.31</td>
<td>3.28 \times 10^{-4}</td>
</tr>
<tr>
<td>CNN Features(15)[Zei14]</td>
<td>65.70%[Zei14]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SADL(15)</td>
<td><strong>75.36%</strong></td>
<td>4829.01</td>
<td><strong>2.79 \times 10^{-5}</strong></td>
</tr>
<tr>
<td>DSADL(15)</td>
<td><strong>74.38%</strong></td>
<td>-</td>
<td><strong>2.79 \times 10^{-5}</strong></td>
</tr>
<tr>
<td>ResFeats-50(30)[Mah16]</td>
<td>75.40%[Mah16]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We use Caltech 256 to test both SADL and Distributed SADL. Our SADL achieves the highest accuracy, and our Distributed SADL also achieves a comparable performance with an extremely fast testing time, even though the dimension of the features are increased. For reference, we also compare our method with two network methods [Zei14; Mah16]. In [Zei14], Zeiler et al. constructed a convolutional network per-trained by ImageNet, and then learned an adapted convolutional network for Caltech 256 based on the features of the former network. As trained by 15 samples...
of each class, our performance is 10% higher than the CNN result. ResFeats-50[Mah16] is a most recent convolutional network method. This method is trained by 30 samples of each category with 50 layers. Though ResFeats-50 utilizes twice more training samples than ours, our result is still very comparable.

2.6.7 Scene 15

![Scene 15 Dataset Examples](image)

Figure 2.12 Scene 15 Dataset Examples

Scene 15 dataset contains a total of 15 categories of different scenes, and each category has around 200 images. The examples are listed in Figure 2.12. As different scenes contain many common components, and different components also share a large number of common features, training on the Scene 15 dataset is prone to a remarkable amount of inter-class similarity. Proceeding as for the Caltech 101 dataset, we compute the spatial pyramid features for scene images. A four-level spatial pyramid (i.e., each image is grid into $1 \times 1, 2 \times 2, 4 \times 4$ and $8 \times 8$) and a codebook of size 200 is used. The final features are obtained by applying PCA to reduce the dimension of spatial pyramid features to 3000. We randomly pick 100 images per class as training data, and use the rest of images as testing data. The settings and steps follow [Jia13a]. The dictionary size is set to 1500, $\lambda_1 = 0.001$, $\lambda_2 = 0.003$ and $p = 283$.

Table 2.6 Classification Results on Scene 15 Dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Classification Accuracy(%)</th>
<th>Training Time(s)</th>
<th>Testing Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADL+SVM[She14]</td>
<td>80.55%</td>
<td>484.41</td>
<td>$1.73 \times 10^{-4}$</td>
</tr>
<tr>
<td>SRC[Wir09]</td>
<td>91.80%</td>
<td>No Need</td>
<td>$4.06 \times 10^{-1}$</td>
</tr>
<tr>
<td>LC-KSVD[Jia13a]</td>
<td><strong>98.83%</strong> (92.9%[Jia13a])</td>
<td>390.22</td>
<td>$1.81 \times 10^{-3}$</td>
</tr>
<tr>
<td>DADL[Guo16]</td>
<td>97.81% (98.3%[Guo16])</td>
<td><strong>33.03</strong></td>
<td>$4.62 \times 10^{-4}$</td>
</tr>
<tr>
<td>CADL[Wan17]</td>
<td>98.49% (98.6%[Wan17])</td>
<td>4637.80</td>
<td>$6.02 \times 10^{-5}$</td>
</tr>
<tr>
<td>SK-SVDADL[Wan18b]</td>
<td>96.84% (97.4%[Wan18b])</td>
<td>66.79</td>
<td>$1.06 \times 10^{-4}$</td>
</tr>
<tr>
<td>SADL</td>
<td>98.50%</td>
<td>174.20</td>
<td><strong>2.41 \times 10^{-5}</strong></td>
</tr>
</tbody>
</table>
The classification results, training and testing time are summarized in Table 2.6. Our performance is slightly lower than LC-KSVD, but is still higher than all other methods. However, the testing phase is superior to the others. Note that the testing time is the fastest, and the training time is faster than CADL, LC-KSVD and ADL+SVM.

2.6.8 Comparative Evaluation

To investigate the effect of different constrains in the optimization problem in Eq. (2.4), we respectively learn ADL by neglecting one of the two constraints and test the resulting algorithms on 10 realizations. The results of 5 datasets are compared with SADL in the Table 2.7. As there is no linear classifier, training ADL with only the constraint $H = QU$, we assign the class labels $y$ for an observed image $x$ by $y = \min_c \|H_c - Q(Dx)\|_F^2$. For ADL with only the constraint $L = WU$, labels of images are assigned by the classifier $W$.

<table>
<thead>
<tr>
<th>ADL+constrains</th>
<th>YaleB</th>
<th>AR</th>
<th>Caltech 101</th>
<th>Scene 15</th>
<th>Caltech 256</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H = QU$</td>
<td>95.37%</td>
<td>96.70%</td>
<td>74.44%</td>
<td>98.46%</td>
<td>75.05%</td>
</tr>
<tr>
<td>$L = WU$</td>
<td>95.52%</td>
<td>97.12%</td>
<td>74.45%</td>
<td>98.36%</td>
<td>75.35%</td>
</tr>
<tr>
<td>SADL</td>
<td>96.35%</td>
<td>97.13%</td>
<td>74.45%</td>
<td>98.50%</td>
<td>75.36%</td>
</tr>
</tbody>
</table>

The results show that both of these two constraints exhibit a similar behavior, the universal classifier has a slightly better performance, while jointly learning two constraints achieves the best result. The results also support our goal of mitigating the inter-class common features. Therefore, our algorithm achieves a better performance in Scene 15 and Caltech 256, which have many common features among classes.

2.7 Conclusion

We proposed an image classification method referred to as Structured Analysis Dictionary Learning (SADL). To obtain SADL, we constrain a structured subspace (cluster) model in the enhanced ADL method, where each class is represented by a structured subspace. The enhancement of ADL is realized by constraining the learning by a classification fidelity term on the sparse coefficients. Our formulated optimization problem was efficiently solved by the linearized ADM method, in spite of its non-convexity due to bilinearity. Taking advantage of analysis dictionary, our method achieves a significantly faster testing time. Furthermore, a Distributed SADL (DSADL) was also proposed to address the scalability problem. Both discriminative structure and fast testing phase are well preserved in the DSADL. Even though the algorithm was run by many multi-clusters, the performance was still stable and comparable to the centralized SADL.

Our experiments demonstrate that our approach has at least a comparable, and often a better
performance than state-of-the-art techniques on five well-known datasets and achieves superior training and testing times by orders of magnitude.

A possible future direction for improving our method could be to leverage the discriminative nature of the synthesis dictionary and the efficiency of the analysis dictionary together. This can achieve a more discriminative power and high efficiency.
3.1 Introduction

In recent years, object recognition has been remarkably improved even for large-scale recognition problems, such as the ImageNet ILSVRC challenge [Rus15]. The latest deep neural networks (DNN) architectures [Ser13; Sim14; Don14; He16b; Ben13] reportedly achieve a super-human performance on the ILSVRC 1K recognition task. In spite of the existing successes, most techniques rely on the supervised training of DNNs to obtain visual representations of each category with abundant labeled training examples. In practice, however, the number of objects of different categories is not predetermined and often follows a long-tail distribution. Except for popular categories which have a large amount of training samples, most categories may have only few or no training samples. As a result, the capability of up-to-date DNN models in recognizing those scarce objects is highly limited. Additionally, it is intractable to manually collect and annotate the training samples for each object category. Simply for animal recognition problem, for example, there are about 8.7 million different animal species on earth to learn. Zero short learning (ZSL) [Soc13; Lam14] and one/few shot learning techniques [FF06; Lak15; Lak15; Vin16; Rav16] are thus proposed to overcome such limitations. In contrast to training the supervised DNNs, ZSL is to learn the capability of transferring the relevant knowledge from known objects to knowledge-less or unknown ones.
ZSL has recently received wide attention in various studies [Sch13; Lam14; RP15; Shi15; Cha16a; Zha16c; Cha16b; Aka16; Xia17]. It aims to recognize the objects with no corresponding labeled samples (unseen objects) from the knowledge obtained from the objects with a considerable number of training samples (seen objects). In order to transfer the knowledge from seen classes to unseen ones, there is a common assumption that the side information about each class is available, such as class attributes [Kan12; Lam14; Fro13] or word vectors [Fro13; Soc13]. When both seen and unseen class names are embedded in the semantic word vector space, they are then called class prototypes [Fu14]. Since the distance between word vectors or class attributes is measurable, and the visual features could also be projected onto a semantic space, the idea of conventional ZSL is to learn a general mapping or relationship between the visual features and side information from labelled seen classes, and then apply it to the unseen classes. Recognizing unseen objects is performed by projecting the visual features of unseen classes into the semantic spaces by learnt mapping and assign the label by a simple nearest neighbour (NN) search, which becomes a classical classification problem.

Figure 3.1 The Illustration of JCMSPL.
Most existing ZSL models [Lam14; Aka15; Soc13; Fro16; Ree16; LB15; Shi15; Kod15; Sho16; Zha17a] mainly focus on the projection between visual and semantic features, when the reconstruction of the original feature is not taken into account. Although seen and unseen classes share some overlapping domain, this may cause the problem of projection domain shift. Kodirov et. al [Kod17] recently proposed a novel method, called Semantic AutoEncoder (SAE), which includes a reconstruction constraint on the original visual features. Such a projection not only includes the mapping from the visual to semantic spaces, but also preserves the information for recovering the original visual features, mitigating the domain shift problem (this is in a sense, akin to accounting for the structure that assembles the features, i.e. the correlation among them). After that, Liu et. al [Liu18b] restricted the projection function in SAE to be low-rank to enhance the robustness. Even though the reconstruction of visual features demonstrates its capability of alleviating the domain shift problem, [Kod17] and [Liu18b] actually also belong to the direct projection method between visual and semantic feature spaces. All above methods do not exploit the intrinsic mapping structure between the visual and semantic space. In general, when people observe an unknown object, they usually first search for similar concepts in mind, and then match it with the semantic meaning of the same concept, and conversely. These concepts have a class-to-class map on both of the visual and semantic features. As a result, in this Chapter, we propose a novel method named Joint Concept Matching-Space Projection Learning (JCMSPL) to mimic such human thinking behavior and take the advantage of the self-reconstruction to cope with the domain shift issue. We assume there is a common concept space incorporating distinct class concepts by introducing the class-specific information, and both visual and semantic features could be precisely projected and reconstructed by such common and distinct concepts. Hence, in such a way, the domain-invariant is introduced by the distinct concept mapping and the self-reconstruction of both visual and semantic features. In addition, such a distinct common concept space can precisely one-to-one match visual and semantic features of each class.

To have a more clear intuition, our proposed method is illustrated in Figure 3.1. As shown, different categories in the visual space have many overlaps of colors and backgrounds, such as the similar color of an otter and a zebra, the same sea background of the polar bear and the otter, which all result in an non-separability issue among different classes. Similarly, in the semantic space, many objects also share common attributes. For examples, both the otter and the polar bear have in common, "brown", "water" and "eat fish" tags. Thus, a distinct concept space is adopted, where both visual and semantic features of these seen classes are projected as "separable concepts". Meanwhile, the visual and semantic features of each class are in one-to-one match, much like a human associates the visual features to words. Additionally, when considering the unseen classes (such as, birds), these have much different features in both visual and semantic spaces in comparison to the seen classes (such as, otter, polar bear and zebra), and also cause domain gap between seen and unseen classes. In order to mitigate such effects, the reconstructions of both visual and semantic features are also included, which are illustrated by the dash-line two-way arrows in Figure 3.1.

Our main contributions, in this sequel, are summarized as follows:
• A novel ZSL is proposed and based on the intermediate common concept space with class-specific information to better match visual and semantic features.

• Both visual and semantic features can be reconstructed by the common concept space to mitigate the domain-shift problem.

• An efficient algorithm based on Sylvester equation is developed, and is shown to achieve state-of-arts performance on four benchmark datasets for ZSL and generalized ZSL.

The balance of the paper is organized as follows: In Section 3.2, we review the literature and background of relevance to this paper. We define the problem, formulate our novel approach and propose its algorithmic solution and recognition scheme in Section 3.3. Substantiating experimental results and evaluations as well as the convergence and complexity analyses are presented in Section 4.5. Finally, we provide some concluding remarks in Section 4.6.

3.2 Related Work

3.2.1 Class-specific Information

In order to improve classification accuracy, supervised learning methods [Jia13b; Tan18] incorporate class-specific information which makes the classifier more discriminative is expressed as a block-diagonal matrix or a simple binary label matrix. In both [Jia13b; Tan18], the class-specific information makes the labels more consistent and distinct to each other. In this work, we build on prior research to introduce a similar class-specific information matrix (a block-diagonal matrix) to support a common concept space with distinct characteristics for different classes. That is to make the projected visual and semantic features of different classes to be more discriminative.

3.2.2 Projection Learning

Various ZSL methods have recently been proposed, and can be divided into three groups primarily distinguished by their associated projection methods. (1) The first group employs a projection function from a visual space to a semantic space, and subsequently determines the class labels in the semantic space. The projection function can be a conventional regression model [Lam14; Aka15] or a deep neural network [Soc13; Fro13; Ree16; LB15]. This kind of projection methods are regarded as forward projection Learning. (2) In contrast to the first group and to alleviate the well-known hubness problem in the nearest neighbor search in high-dimensional spaces [Rad10], the second one is to learn the reverse projection function from the semantic space to the feature space [Shi15; Kod15; Sho16; Zha17a]. (3) The third group projects both the visual and semantic features to an intermediate space [Cha16a; Lu15; Zha16b]. Our proposed JCMSPL model similarly adopts an intermediate space, which further used to generate two reconstruction constraints. By using the intermediate space, the visual features can be projected onto the semantic space, and vice versa. Therefore, JCMSPL can be viewed as a hybrid of all these three groups, which indirectly
integrate both forward and reverse projections with an intermediate space for ZSL with additional and enhanced class-specific knowledge.

### 3.2.3 Projection Domain Shift

The aforementioned domain shift problem first noted in [Fu14], has been addressed in two versions of the ZSL problems has been discussed in the literature. One is inductive ZSL [Kod17; Liu18b], where the projection function only relies on the seen classes, when all the unseen data is only used for testing. Another is transductive ZSL [Kod15; Roh13; Zha18], and incorporates the unlabelled unseen classes into the projection function learning to alleviate the domain shift problem. Our JCMSPL is based on the inductive ZSL and only uses the reconstruction constraints of both visual and semantic features together with class-specific information to cope with domain shift problem.

### 3.3 Methodology

#### 3.3.1 Notation

Uppercase and lowercase bold letters respectively denote matrices and vectors throughout the paper. The transpose and inverse of matrices are respectively represented by the superscripts $T$ and $^{-1}$, as in $A^T$ and $A^{-1}$. The identity matrix and all-zero matrix are respectively denoted by $I$ and $0$.

---

**Figure 3.2** The framework of our purposed JCMSPL is comprised of 3 procedures. (I) A CNN is first used to extract the visual features $X$, such as GoogleNet [Sze15], VGG-19 [Sim14] and ResNet101 [He16b], and then project them by the function $A$ into the common concept space $C$. Finally, $A^T$ is employed to reconstruct the $X$ from concept space $C$. (II) The attributes/word vector $Y$ is used to embed the different classes into the semantic space. The project function $B$ maps the semantic feature $Y$ into the common concept space $C$ as well, and $B^T$ reconstruct $Y$ from $C$. (III) A block-diagonal matrix $H$ supervise and support the common concept space $C$ to be distinct and separable, where the light-yellow part shown in the class-specific information is with all entrances of 0 and the dark-yellow part is occupied by the elements of 1.
3.3.2 Problem Statement

The goal of ZSL is to assign the unseen class label to the unseen samples, and both unseen labels and samples are independent from the training phase.

Let $S = \{X_s, Y_s, L_s\}$ denote the set of seen classes with $c_s$ seen classes and $n_s$ labeled samples. And let $U = \{X_u, Y_u, L_u\}$ denote the set of unseen classes with $c_u$ seen classes and $n_u$ labeled samples. $X_s \in \mathbb{R}^{m \times n_s}$ and $X_u \in \mathbb{R}^{m \times n_u}$ are $m$-dimensional visual features samples in the seen and unseen sets, $Y_s \in \mathbb{R}^{d \times n_s}$ and $Y_u \in \mathbb{R}^{d \times n_u}$ are associated class-level attributes, namely semantic features. $L_s, L_u$ are respectively the corresponding label sets of seen and unseen classes. Based on the definition of ZSL, the labels of seen and unseen sets have no overlap, i.e., $L_s \cap L_u = \emptyset$, and ZSL aims to learn a classifier $p : X_u/Y_u \rightarrow L_u$, so as to predict the label for unseen classes, where $p$ is learned only on basis of the seen class sets $S = \{X_s, Y_s, L_s\}$.

3.3.3 Model Formulation

As mentioned in Section 3.1, JCMSPL incorporates a distinct common concept space with class-specific information, and both visual and semantic features self-reconstructions. Therefore, JCMSPL consists of 3 joint procedures as illustrated in Figure 4.1, and formulated as follows:

$$
\min_{A, B, C} \frac{1}{2} \|AX_s - C\|_F^2 + \frac{\lambda_1}{2} \|BY_s - C\|_F^2 + \frac{\lambda_2}{2} \|C - H\|_F^2 + \frac{\lambda_3}{2} \|X_s - A^T C\|_F^2 + \frac{\lambda_4}{2} \|Y_s - B^T C\|_F^2,
$$

(3.1)

where $A \in \mathbb{R}^{k \times m}$ is the projection matrix from the visual space to the common concept space $C \in \mathbb{R}^{k \times n_s}$, while $B \in \mathbb{R}^{k \times d}$ is the projection matrix from the semantic space to the common space. We further require the transpose matrices $A^T$ and $B^T$ to be respectively the reverse projection matrix from the common space $C$ to the visual and semantic spaces, so that visual and semantic features can also be reconstructed by the common space. $H \in \mathbb{R}^{k \times n_s}$ is a block-diagonal matrix, predefined by the class-specific information to make the common concept space distinct, and thus enhance the matching of each class more accurately. $\lambda_1, \ldots, \lambda_4$ are the turning parameters.

3.3.4 Algorithmic Solution

Since the objective functional in Eq. (3.1) is a multi-convex problem, we may reliably update the variables by a block-coordinate descent method.

**Update A:** When $B$ and $C$ are fixed, $A$ is updated by

$$
A^*_{t+1} = \arg \min_A \frac{1}{2} \|AX_s - C_t\|_F^2 + \frac{\lambda_3}{2} \|X_s - A^T C_t\|_F^2.
$$

(3.2)
As \( \|Z\|_F = \|Z^T\|_F \), Eq.(3.2) can be rewritten as

\[
A^*_{t+1} = \min_A \frac{1}{2} \|AX_s - Ct\|_F^2 + \frac{\lambda_3}{2} \|X_s^T - Ct^T A\|_F^2.
\]

(3.3)

Then, taking the derivative of Eq. (3.3) and setting it to zero, we obtain:

\[
\lambda_3 C_t C_t^T A^*_{t+1} + A^*_{t+1} X_s X_s^T = (1 + \lambda_3) C_t X_s^T.
\]

(3.4)

Denote,

\[
M_A = \lambda_3 C_t C_t^T,
\]

\[
N_A = X_s X_s^T,
\]

\[
T_A = (1 + \lambda_3) C_t X_s^T,
\]

we have:

\[
M_A A^*_{t+1} + A^*_{t+1} N_A = T_A.
\]

(3.5)

To solve Eq. (3.5), we use the following definition and theorems:

**Definition 2.** A Sylvester equation \([Syl84]\) is a matrix equation of the following form:

\[
RZ + ZS = T.
\]

(3.6)

When \( R, S \) and \( T \) are given, the problem is to find the possible matrices \( Z \) that obey this equation.

**Theorem 3.** \([Lan85]\) The sufficient condition of Eq. (3.6) to have a solution \( Z \) is that:

The matrix \( \begin{bmatrix} R & 0 \\ 0 & S \end{bmatrix} \) is similar to the matrix \( \begin{bmatrix} R & T \\ 0 & -S \end{bmatrix} \).

**Theorem 4.** \([Lan85]\) The sufficient condition for Eq. (3.6) to have a unique solution \( Z \) is that:

The eigenvalues \((\sigma_1^R, \sigma_2^R, \ldots, \sigma_r^R)\) of \( R \) and the eigenvalues \((\sigma_1^S, \sigma_2^S, \ldots, \sigma_s^S)\) of \( S \) satisfy \( \sigma_i^R \neq -\sigma_j^S, \forall i = 1, \ldots, r, \forall j = 1, \ldots, s \).

More details and proofs of Definition 1, Theorem 1 and 2 can be found in [Lan85].

By Definition 2, Eq. (3.5) is a Sylvester equation and easy to meet the sufficient condition in Theorem 4, as ZSL is based on real image data. Eq. (3.5) is thus solved efficiently by the Bartels-Stewart algorithm [Bar72], which can be implemented by a single line code: Sylvester in MATLAB\(^1\).

**Update B:** When \( A \) and \( C \) are fixed, \( B \) is updated by

\[
B^*_{t+1} = \arg \min_B \frac{\lambda_1}{2} \|BY_s - Ct\|_F^2 + \frac{\lambda_4}{2} \|Y_s - B^T C_t\|_F^2.
\]

(3.7)

Similarly updating \( A \), we have the Sylvester equation related to \( B \):

\[
M_B B^*_{t+1} + B^*_{t+1} N_B = T_B.
\]

(3.8)

\(^1\)https://www.mathworks.com/help/matlab/ref/sylvester.html
where

\[ M_B = \lambda_4 C_t C_t^T, \]
\[ N_B = \lambda_1 Y_s Y_s^T, \]
\[ T_B = (\lambda_1 + \lambda_4) C_t Y_s^T. \]

**Update C:** When \( A \) and \( B \) are fixed, \( C \) is updated by

\[
C^*_{t+1} = \arg\min_C \frac{1}{2} \|A_{t+1} X_s - C\|_F^2 + \frac{\lambda_1}{2} \|B_{t+1} Y_s - C\|_F^2 + \frac{\lambda_2}{2} \|C - H\|_F^2 + \frac{\lambda_3}{2} \|X_s - A^T_{t+1} C\|_F^2 + \frac{\lambda_4}{2} \|Y_s - B^T_{t+1} C\|_F^2.
\]

(3.9)

Derivating Eq. (3.9) and setting it to zero, yields its analytical solution as follows:

\[
C^*_{t+1} = \left((1 + \lambda_1 + \lambda_2) I + \lambda_3 A_{t+1} A^T_{t+1} + \lambda_4 B_{t+1} B^T_{t+1}\right)^{-1} \left(\lambda_2 H + (1 + \lambda_3) A_{t+1} X_s + (\lambda_1 + \lambda_4) B_{t+1} Y_s\right).
\]

(3.10)

We follow the updating steps in each iteration of our algorithm as summarized in Algorithm 1.

---

**Algorithm 3** Joint Concept Matching-Space Projection Learning

**Input:** Visual features training set \( X_s \), semantic features training set \( Y_s \), tuning parameters \( \lambda_1, \ldots, \lambda_4 \), and maximum iteration \( t_{MAX} \).

**Output:** The visual projection function \( A \), the semantic projection function \( B \) and the distinct common space \( C \).

1: Initialize \( A, B \) and \( C \) as random matrices;
2: while not converged and \( t < t_{MAX} \) do
3: \( t = t + 1 \);
4: Update \( A_t \) by solving Eq. (3.5);
5: Update \( B_t \) by solving Eq. (3.8);
6: Update \( C_t \) by Eq. (3.10);
7: end while

---

**3.3.5 Zero-shot Recognition**

After we obtain the projection matrices \( A \) and \( B \), zero-shot recognition can be subsequently performed in two ways:

1. With projection matrices \( A \) and \( B^T \) in hand, whenever a new test sample \( x_{u_i} \in X_u \) is given, the associated semantic feature \( \hat{y}_{u_i} \) of unseen class are easily reconstructed by the visual features using the **equation:**

   (1)
\[
\hat{y}_{ui} = B^T A x_{ui},
\]

(3.11)

The test data in the semantic space can be classified by a simple Nearest Neighbour (NN) classifier based on the distance between the estimated semantic representation \(\hat{y}_{ui}\) and the prototype projections in the semantic space \(Y_u\). The label \(l_{ui}\) for the unseen sample is assigned by,

\[
l_{uj} = \min_{c_j} D(\hat{y}_{ui}, Y_{uj}),
\]

(3.12)

where \(Y_{uj}\) is the prototype attribute vector of the \(c_j\)-th unseen class, \(D\) is an arbitrary distance function.

(2) With projection matrices \(A^T\) and \(B\) in hand, whenever a new test sample \(y_{ui} \in Y_u\) is given, the associated visual feature \(\hat{x}_{ui}\) of an unseen class are easily reconstructed by the semantic features thorough the following method:

\[
\hat{x}_{ui} = A^T B y_{ui}.
\]

(3.13)

The test data in the visual space can be classified by a simple Nearest Neighbour (NN) classifier based on the distance between the estimated visual representation \(\hat{x}_{ui}\) and the prototype projections in the visual space \(X_u\). The label \(l_{ui}\) for the unseen sample is assigned by,

\[
l_{ui} = \min_{c_j} D(\hat{x}_{ui}, X_{uj});
\]

(3.14)

where \(X_{uj}\) is the \(c_j\)-th unseen class prototype projected in the feature space, \(D\) is an arbitrary distance function.

We next validate the proposed approach of both strategies.

### 3.4 Experiments

#### 3.4.1 Datasets and Settings

##### 3.4.1.1 Datasets

Four benchmark datasets are used to evaluate the state-of-art methodologies along with our own. Animals with Attributes (AwA) [Lam14], CUB-200-2011 Birds (CUB) [Wah11], and SUN Attribute (SUN) [Pat14] are three widely used medium-scale datasets in existing ZSL works. But they are not large enough to show the capability of the original motivation of ZSL for scaling up visual recognition. Thus, the ILSVRC2012/ILSCRC2012 (ImNet) [Rus15] is then selected as a large-scale dataset in [Fu16].
AwA [Lam14] consists of 30475 images of 50 animal classes with 85 associated class-level attributes, which is a coarse-grained dataset. 40 classes are used for training, while the remaining 10 classes with 6180 images are used for testing.

CUB [Wah11] is a fine-grained dataset with 11788 images for 200 different types of bird species which are annotated by 312 attributes. The first standard zero-shot split was introduced in [Aka16], where 150 classes are for training and 50 classes are for testing.

SUN [Pat14] is also a fine-grained dataset, which includes 14340 images for 717 types of different scenes categories which are annotated by 120 attributes. Following the split in [Lam14] 645 out of 717 classes are used as a training set, and the remaining 72 classes are for testing.

ImNet [Rus15] contains 218000 images and 1000-dimensional class-level attributes. Following the split in [Fu16], the 1000 classes of ILSVRC2012 are used as seen classes, when the 360 classes of ILSVRC2010 are used as unseen classes, which are not included in ILSVRC2012.

For a fair comparison against published results, we use the same above training (seen) and testing (unseen) splits for our ZSL evaluation. The summary of all those datasets are listed in the Table 3.1.

Table 3.1 The details of four evaluated datasets. Notation: ‘SS’- the semantic space; ‘A’- the attribute, ‘W’- the word vector; ‘SS-D’- the dimension of the semantic space.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#Images</th>
<th>SS</th>
<th>SS-D</th>
<th># Seen/#Unseen Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AwA [Lam14]</td>
<td>30475</td>
<td>A</td>
<td>85</td>
<td>40/10</td>
</tr>
<tr>
<td>CUB [Wah11]</td>
<td>11788</td>
<td>A</td>
<td>312</td>
<td>150/50</td>
</tr>
<tr>
<td>SUN [Pat14]</td>
<td>14340</td>
<td>A</td>
<td>64</td>
<td>645/72</td>
</tr>
<tr>
<td>ImNet [Rus15]</td>
<td>218000</td>
<td>W</td>
<td>1000</td>
<td>1000/360</td>
</tr>
</tbody>
</table>

3.4.1.2 Semantic Spaces

Generally, there are two different types of attributes. One is the attribute annotations, which is for the medium datasets, and another one is the word vector representation, which is used for large-scale datasets. The word vector, word2vec [Mik13], representation is obtained by training a skip-gram text model on a corpus of 4.6M Wikipedia documents.

3.4.1.3 Visual Spaces

In recent ZSL modes, all the visual features are extracted from Convolutional Neural Networks (CNNs) [Sim14; Sze15; He16b] that are pre-trained by the 1K classes in ILSVRC 2012 [Rus15]. In our experiments, the visual features are extracted from pre-trained GoogleNet [Sze15]. It is worthy noting that the visual features used in most compared methods are GoogleNet features, except Table 3.2, where a number of most ZSL models used VGG19 [Sim14] and ResNet101 [He16b] features. Since the source codes of such models are not released, we can not report the results based on
GoogleNet, instead the results reported in the original paper are listed in Table 3.2. But note that, as a demonstration in [Li17], the VGG19 and ResNet101 features usually achieve better performances than the GoogLeNet features in the ZSL task. Since we use GoogleNet features which are not stronger features, it is fair for such comparisons in Table 3.2.

Table 3.2 The standard ZSL classification accuracy (%). For ImNet, hit @5 is reported. For visual Features: G - GoogleNet [Sze15]; V - VGG19 [Sim14]; R - ResNet101 [He16b].

<table>
<thead>
<tr>
<th>Medium Datasets</th>
<th>Large Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>V-Features</td>
</tr>
<tr>
<td>RPL [Shi15]</td>
<td>G</td>
</tr>
<tr>
<td>SSE [Zha15]</td>
<td>V</td>
</tr>
<tr>
<td>JLSE [Zha16b]</td>
<td>V</td>
</tr>
<tr>
<td>SynC [Cha16a]</td>
<td>G</td>
</tr>
<tr>
<td>SAE [Kod17]</td>
<td>G</td>
</tr>
<tr>
<td>LAD [Jia17]</td>
<td>V</td>
</tr>
<tr>
<td>SCoRe [Mor17]</td>
<td>V</td>
</tr>
<tr>
<td>LESD [Din17]</td>
<td>V/G</td>
</tr>
<tr>
<td>CVA [Mis18]</td>
<td>V/R</td>
</tr>
<tr>
<td>f-CLSWGAN [Xia18]</td>
<td>R</td>
</tr>
<tr>
<td>VSZL [Wan18c]</td>
<td>V</td>
</tr>
<tr>
<td>LSAE [Liu18b]</td>
<td>R</td>
</tr>
<tr>
<td>AAW [Kol18]</td>
<td>V</td>
</tr>
<tr>
<td>LSD[Che19]</td>
<td>V</td>
</tr>
<tr>
<td>BZSL[She19]</td>
<td>V/G</td>
</tr>
<tr>
<td>JCMSPL (V → S)</td>
<td>G</td>
</tr>
<tr>
<td>JCMSPL (S → V)</td>
<td>G</td>
</tr>
</tbody>
</table>

3.4.1.4 Parameter Settings

In JCMSPL, there are four tuning parameters $\lambda_1, \ldots, \lambda_4$ in Eq. (3.1). Following [Shi15; Kod17], the parameters are tuned by class-wise cross-validation of the training set. As SUN dataset has multiple splits, in our experiments, we report the average performance of the same splits that are used in [Cha16a].

3.4.1.5 ZSL Settings

**Standard ZSL**: The standard ZSL setting is widely used in recent ZSL works [Aka15; Ree16]. The seen and unseen classes are split following Table 3.1.

**Generalized ZSL**: The generalized ZSL setting has recently emerged [Rah18; Buc17], whose testing
set includes both seen and unseen samples. Such setting is clearly more reflective of real world scenarios.

3.4.1.6 Evaluation Metrics

**Standard ZSL:** The multi-way classification accuracy as previous works are used for three medium-scale datasets, when the flat hit @K classification accuracy as in [Fu16] is used for the large-scale dataset. Hit @K means that for a testing sample, the top K assigned labels should include the correct label. In the experiment, hit @5 accuracy is reported for over all test samples.

**Generalized ZSL:** Three metrics are used in the generalized ZSL. The first one is $acc_s$, the accuracy of classifying seen samples within all classes, which includes both seen and unseen samples. The second one is, $acc_u$ the accuracy of classifying unseen samples within all classes. The third one is $HM$, which is the harmonic mean of $acc_s$ and $acc_u$, i.e.,

$$HM = \frac{2 \times acc_s \times acc_u}{acc_s + acc_u}.$$

3.4.1.7 Competitive Methods

16 existing ZSL methods are used for performance comparison for three medium-scale datasets and 8 state-of-art are used for the large-scale ones, where RPL [Shi15], SSE [Zha15], SJE [Aka15], JLSE [Zha16b], SynC [Cha16a], SAE [Kod17], LAD [Jia17], SCoRe [Mor17], LESD [Din17], CVA [Mis18], f-CLSWGAN [Xia18], VSZL [Wan18c], LESAE [Liu18b], AAW [Kol18], LSD [Che19] and BZSL [She19] are used for medium-scale ones, and DeViSE [Fro13], ConSE [Nor13], AMP [Fu15], SS-Voc [Fu16], SAE [Kod17], CVA [Mis18], VSZL [Wan18c], and LESAE [Liu18b] are used for the large one. These ZSL methods cover a wide range of recent and representative ZSL models and achieve state-of-the-art results.

3.4.2 Experiment Results

3.4.2.1 Standard ZSL

The comparative result of various datasets under standard ZSL settings are listed in Table 3.2. All these comparative results are based on inductive ZSL. That is, no unlabeled unseen samples are incorporated in the training phase. Based on Table 3.2, our method achieves the best results on all three medium-scale datasets and a comparable result on the large dataset, which demonstrates that the reconstructions of both visual and semantic mitigate the domain shift problem and the class-specific distinct common space makes the visual and semantic features match more precisely. For all three medium datasets, our method improves about 1% performance over the strongest competitors. For the large dataset, although our method is only slightly lower than the LESAE, it is still much better than other competitive methods.
3.4.2.2 Generalized ZSL

Table 3.3 The comparative results(%) of generalized ZSL as setting in [Cha16b]. All the results are tests on GoogleNet visual features.

<table>
<thead>
<tr>
<th>Method</th>
<th>AwA</th>
<th>CUB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>acc_s</td>
<td>acc_u</td>
</tr>
<tr>
<td>DAP [Lam14]</td>
<td>77.9</td>
<td>2.4</td>
</tr>
<tr>
<td>IAP [Lam14]</td>
<td>76.8</td>
<td>1.7</td>
</tr>
<tr>
<td>ConSE [Nor13]</td>
<td>75.9</td>
<td>9.5</td>
</tr>
<tr>
<td>APD [Rah18]</td>
<td>43.2</td>
<td>61.7</td>
</tr>
<tr>
<td>GAN [Buc17]</td>
<td>81.3</td>
<td>32.3</td>
</tr>
<tr>
<td>SAE [Kod17]</td>
<td>67.6</td>
<td>43.3</td>
</tr>
<tr>
<td>JCMSPL (ours)</td>
<td>48.3</td>
<td>56.4</td>
</tr>
</tbody>
</table>

Following the same setting of [Cha16b], we extract out 20% of seen class data samples and mix them with the unseen class samples. The generalized ZSL of AwA and CUB are listed in Table 3.3, which includes 6 competitive methods. Although the HM score of AwA is slightly lower than SAE, it is still comparable, and its acc_u, the unseen to all class classification accuracy, is still higher than SAE, thus demonstrating a better generalization capability. For the CUB dataset, our method achieves the highest HM and acc_u, which again shows that our method is favored over the generalized ZSL setting. The high accuracy for acc_s reflects that the method is overfitting when training for seen classes and is difficult to generalize to unseen ones.

3.4.3 Further Evaluations

3.4.3.1 Ablation Study

Our JCMSPL model can also be simplified as follows:

1. When $\lambda_2 = 0$, then the class-specific information is not used, our JCMSPL reduces to the joint space projection with reconstruction, and denoted as JCMSPL1, i.e.,

$$\min_{A,B,C} \frac{1}{2} \|AX_s - C\|_F^2 + \frac{\lambda_1}{2} \|BY_s - C\|_F^2$$

$$+ \frac{\lambda_3}{2} \|X_s - AT C\|_F^2 + \frac{\lambda_4}{2} \|Y_s - BT C\|_F^2.$$ 

2. When $\lambda_3 = 0$ and $\lambda_4 = 0$, the class-specific information is used, but the reconstructions of both visual and semantic space are not taken in account. Our JCMSPL reduces to the joint concept
matching space projection without reconstruction, which is denoted as JCMSPL0, \textit{i.e.},

\[
\min_{A, B, C} \frac{1}{2} \| A X_s - C \|_F^2 + \frac{\lambda_1}{2} \| B Y_s - C \|_F^2 + \frac{\lambda_2}{2} \| C - H \|_F^2.
\]

This is similar to the third group of projection learning in the literature but with the class information.

(3) When \(\lambda_2 = 0, \lambda_3 = 0\) and \(\lambda_4 = 0\), both the class-specific information and the reconstructions of both visual and semantic space are not used. Our JCMSPL then reduces to the joint concept matching space projection without reconstruction and any class-specific information, which is similar to the third group of projection learning in the literature [Cha16a; Zha16b]. This is denoted as intermediate space projection learning (IPL), \textit{i.e.},

\[
\min_{A, B, C} \frac{1}{2} \| A X_s - C \|_F^2 + \frac{\lambda_1}{2} \| B Y_s - C \|_F^2.
\]

(4) When \(\lambda_1, \lambda_2, \lambda_3\) and \(\lambda_4 = 0\) and using the semantic space instead of the intermediate/common space, JCMSPL is finally reduced to the original forward projection learning method [Aka15], denoted as FPL, \textit{i.e.},

\[
\min_{A} \frac{1}{2} \| A X_s - Y_s \|_F^2.
\]

To evaluate the contribution of proposed Full JCMSPL method, its simple reduced FPL, IPL, JCMSPL0, JCMSPL1 are compared with same standard splits of AwA and CUB datasets. The standard ZSL accuracy of hereabove simple JCMSPL methods are listed and shown in Table 3.4 and Fig. 3.3.

<table>
<thead>
<tr>
<th>Projection Method</th>
<th>AwA</th>
<th>CUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPL</td>
<td>72.7%</td>
<td>40.2%</td>
</tr>
<tr>
<td>IPL</td>
<td>77.1%</td>
<td>46.5%</td>
</tr>
<tr>
<td>JCMSPL0</td>
<td>82.4%</td>
<td>60.3%</td>
</tr>
<tr>
<td>JCMSPL1</td>
<td>84.7%</td>
<td>61.6%</td>
</tr>
<tr>
<td>Full JCMSPL</td>
<td>86.2%</td>
<td>62.6%</td>
</tr>
</tbody>
</table>

The ablation study results shows that: (1) When comparing the IPL and FPL, it shows that the intermediate space respectively brings 4.4% and 6.3% improvements for AwA and CUB datasets. The common space introduces the latent representations of both visual and semantic features that
Figure 3.3 Ablation study results of standard ZSL on two medium-scale datasets.

Figure 3.4 Parameters analysis for $\lambda_1, \ldots, \lambda_4$.

enhance their similarities. (2) The result of JCMSPL0 = IPL + class-specific information also has a 5% – 13% gain in comparison to IPL, which validates the effectiveness of class-specific information. Even with presence of good results, the class-specific information also introduce a great than 1% gains, as comparing JCMSPL1 with Full JCMSPL. Class-specific latent space well matches the visual and
semantic features class-to-class. (3) As the shown result of JCMSPL1 = IPL + reconstructions vs. IPL, both reconstructions of visual and semantic features are important for ZSL, which results in 7%—14% improvements. (4) To combine with all these class-specific information and feature reconstructions, our full JCMSPL has significant improvements ranging from 9%—16% when comparing to IPL and achieves 1%—2% improvements when comparing to JCMSPL1, the one only with such reconstruction constraints.

### 3.4.3.2 Parameters Analysis

Figure 3.4 shows that the values of $\lambda_1, \ldots, \lambda_4$ that achieve the best performance in different datasets. The detailed settings of four benchmark datasets are listed in Table 3.5.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>AwA</th>
<th>CUB</th>
<th>SUN</th>
<th>ImNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>$1e-3$</td>
<td>1</td>
<td>$1e-4$</td>
<td>$1e-5$</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>$1e3$</td>
<td>$1e-3$</td>
<td>$1e-2$</td>
<td>$1e-5$</td>
</tr>
<tr>
<td>$\lambda_3$</td>
<td>$1e7$</td>
<td>$1e4$</td>
<td>$1e-4$</td>
<td>$1e1$</td>
</tr>
<tr>
<td>$\lambda_4$</td>
<td>$1e2$</td>
<td>$1e-1$</td>
<td>$1e-4$</td>
<td>$1e-4$</td>
</tr>
</tbody>
</table>

First we evaluate the effect of the constraint that imposed by projecting the semantic features to common distinct space by tuning the $\lambda_1$ during our algorithm training phase. Since visual features have different structures from semantic features, the weights of both visual feature mapping and semantic features mapping should be different, too. In Figure 3.4a, it is shown that our algorithm achieves better performance, when the weights of semantic feature mapping is in the range of $[10^{-5}, 10]$ for all four datasets.

We then evaluate $\lambda_2$, the influence of class-specific information term during training. From the evaluation in Figure 3.4b, it is readily seen that the classification accuracy tends to be better during the range of $[10^2, 10^6]$ for AwA, the range of $[10^{-3}, 10^{-2}]$ for CUB, the range of $[10^{-5}, 10^{-2}]$ for SUN and the range of $[10^{-5}, 1]$ for ImNet. Comparing with the magnitude of other parameters in each dataset, the class-specific information term has large weights for AwA and SUN datasets, which imparts the distinct knowledge of different class for a better matching between visual and semantic features. Although its weight is small for CUB dataset, the accuracy drops when its weight $\lambda_2$ decreases. This validates that the class-specific information term is also helpful for zero-shot recognition to be robust against the domain shift issue, and match the same class visual and semantic features.

In addition, through the analysis of parameter $\lambda_3$ and $\lambda_4$ (Figure 3.4c and 3.4e), they show that the ranges of both $\lambda_3$ and $\lambda_4$ that achieve promising performances, are respectively less than $10^8$ and $10^3$ on four datasets. Although four different datasets have different magnitudes, the $\lambda_3$ and $\lambda_4$ are relatively larger than $\lambda_1$ and $\lambda_2$ (the detailed values is listed in Table 3.5). This shows that the reconstructions of both visual and semantic features can improve the zero-shot learning ability,
narrow the domain shift gap, and explore more intrinsic structure within seen data.

### 3.4.3.3 Convergence and Complexity Analysis

**Convergence Analysis:** In light of the non-convexity of Eq. (3.1), the convergence of our algorithm is not guaranteed by standard results. We hence separately prove the convergence of our algorithm: let \( f(A, B, C) \) be the loss function of Eq. (3.1), then the following result follows.

**Theorem 5.** The sequence \( \{\Theta_t = (A_t, B_t, C_t)\}_{t=1}^{\infty} \) converges to the following set of feasible stationary points of the loss function \( f \), which is bounded by a universal constant \( R \) depending on the initialization\(^2\):

\[
Q = \{\Theta = (A, B, C) \mid \|\Theta\| < R\}.
\]

Theorem 10 states that Algorithm 1 not only converges, but also generates a solution sequence that eventually converges to stationary points of the underlying optimization. Theorem 10 is proved in the Appendix B.

**Complexity Analysis:** Since the solution complexity of sylvester equation only depends on the dimension of rows, the complexities of updating \( A \) and \( B \) are only related with the dimension of features, rather than the sample size \( n \). That is, the complexities of updating \( A \) and \( B \) are respectively \( O(m^3) \) and \( O(d^3) \), while the complexity of updating \( C \) is \( O(k^3 + k^2 m + k^2 d + kmn_s + kdn_s) \leq O(max(k,m,d,n_s)^3) \).

![Figure 3.5 Convergence of JCMSPL on AwA and CUB Datasets. The left-side vertical axis is for AwA dataset, while the right-side vertical axis is for CUB dataset.](image)

The empirical results also show that our proposed JCMSPL algorithm converges very fast. Figure 3.5 illustrates the convergence curve of JCMSPL on two medium-scale datasets. It is clearly shown that the value of our objective function decreases quickly and stabilizes at 35, and our algorithm

\(^2\)The norm \( \|\Theta\| \) is any norm which is continuous with respect to the 2-norm of the components, for example their sum of 2-norms.
converges in 35 iterations for all four datasets. In light of its convergence and complexity properties, which are validated by empirical results, our proposed JCMSPL algorithm is well suited to practical problem in realistic scenarios.

3.5 Conclusion

We proposed a novel inductive ZSL method by incorporating the class-specific information in a common latent space and the reconstructions of both visual and semantic features. In contrast to most of the existing ZSL methods, they neither consider the reconstructions of features nor involve the class-specific information in latent common space. Such class-specific latent space provides more distinct information, and such reconstructions also enhance the robustness by mitigating the domain shifts. Our proposed JCMSPL leverages the intrinsic structure of visual and semantic features as well as their class-level matching. An efficient algorithm is developed and followed by a theoretically rigorous algorithm analysis. Extensive experiments on four benchmark datasets demonstrate that our proposed JCMSPL method yields superior classification performances for both standard and generalized ZSL than other well-established inductive ZSL methods.
4.1 Introduction

Dictionary Learning/Sparse Coding has demonstrated its high potential in exploring the semantic information embedded in high dimensional noisy data. It has been successfully applied for solving different inference tasks, such as image denoising [Ela06], image restoration [Xu16], image super-resolution [Zho12; Ska16], audio processing [Gro07] and image classification [Zha16a].

While Synthesis Dictionary Learning (SDL) has been greatly investigated and widely used, the Analysis Dictionary Learning (ADL)/Transform Learning, as a dual problem, has been getting greater attention for its robustness property among others [Rub13b; Bia16; Tan16]. DL based methods have primarily focused on learning one-layer dictionary and its associated sparse representation. Other variations on the classification theme have also been appearing with a goal of addressing some recognized limitations, such as task-driven dictionary learning [Mai09c], first introduced to jointly learn the dictionary, its sparse representation, and its classification objective. In [Aha06], a label consistent term is additionally considered. Class-specific dictionary learning has been recently shown to improve the discrimination in [Ram10; Yan11b; Wan13] at the expense of a higher
DeTraMe – Net for 2 – layer Deep Dictionary Learning

**Figure 4.1** 2-layer DeTraMe-Net model. Each layer solves a DL problem, which is transformed into the combination of Transforming Learning (i.e., linear layer in brown dashed lines) and Q-Metric Learning (i.e., RNN in red dashed dot lines). A truncated 2-iterations RNN is unfolded. Sparsity is imposed by shifted-ReLU functions. In the forward pass, we first use a linear layer to learn the new representation $Z^{(1)}$ for input data $X$. The RNN is then used to iteratively learn the optimal sparse representation $U^{(1)}$. For the second layer, the sparse representation $U^{(1)}$ is used as input to learn the second layer sparse representation $U^{(2)}$. Finally, a cross-entropy loss based on $U^{(2)}$ and ground truth $Y$ is used. The parameters $W^{(i)}$, $\tilde{W}^{(i)}$, $h^{(i)}$, and $b^{(i)}$, $i = 1, 2$, in the linear layer and RNN parts are learned by back-propagation.

complexity. On the ADL side, more and more efficient classifiers [Guo16; Wan17; Wan18b; Tan18; Tan19a] have resulted from numerous research efforts, and have yielded to an outperformance of SDL in both training and testing phases [Tan19b].

DL methods with their associated sparse representation, present significant computational challenges addressed by different techniques, including K-SVD [Aha06; Rub13b], SNS-ADL [Bia16] and Fast Iterative Shrinkage-thresholding Algorithm (FISTA) [Bec09]. Meant to provide a practically faster solution, the alternating minimization of FISTA still exhibited limitations and a relatively high computational cost.

To address these computational and scaling difficulties, differentiable programming solutions have also been developed, to take advantage of the efficiency of neural networks. LISTA [Gre10] was first proposed to unfold iterative hard-thresholding into an RNN format, thus speeding up SDL. Unlike conventional solutions for solving optimization problems, LISTA uses the forward and backward passes to simultaneously update the sparse representation and dictionary in an efficient manner. In the same spirit, sparse LSTM (SLSTM) [Zho18] adapts LISTA to a Long Short Term Memory structure to automatically learn the dimension of the sparse representation.

Although the aforementioned differentiable programming methods are efficient at solving a single-layer DL problem, the latter formulation still does not yield the best performance in image classification tasks. With the fast development of deep learning, Deep Dictionary Learning (DDL) methods [Tar16; Mah17] have thus come into play. In [Hua18], a deep model for ADL followed by a
SDL is developed for image super-resolution. Also, [Mah19] deeply stacks SDLs to classify images by achieving promising and robust results. Unsupervised DDL approaches have also been proposed, with promising results [Mag18; Gup20].

However, to the best of our knowledge, no DDL model which can provide both a fast and reliable solution has been proposed.

The proposed work herein, aims at ensuring the discriminative ability of single-layer DL while providing the efficiency of end-to-end models. To this end, we propose a novel differentiable programming method to jointly learn a deep metric together with an associated transform. Cascading these canonical structures will exploit and strengthen the structure learning capacity of a deep network, yielding what we refer to a Deep Transform and Metric Learning Network (DeTraMe-Net). This newly proposed approach not only increases the discrimination capabilities of DL, but also affords a flexibility of constructing different DDL or Deep Neural Network (DNN) architectures. As will be later shown, this approach also resolves usually arising initialization and gradient propagation issues in DDL.

As shown in Figure 4.1, in each layer of DeTraMe-Net, the DL problem is decomposed as a transform learning one, i.e. a linear layer part cascaded with a nonlinear component using a learned metric. The latter, referred to as Q-Metric Learning, is realized by an RNN. One of the contributions of our work is to show how DDL can theoretically be reformulated as such a combination of linear layers and RNNs. Decoupling the metric and the dual frame operator (pseudo-inverse of dictionary) into two independent variables is also shown to introduce additional flexibility, and to improve the power of DL. On the practical side, and to achieve a faster and simpler implementation, we impose a block-diagonal structure for Q-Metric Learning leading to parallel processing of independent channels. Moreover, a convolutional operator is also introduced to decrease the number of parameters, thus leading to a Convolutional-RNN. Additionally, the Q-Metric Learning part may be viewed as a non-separable activation function that can be flexibly included into any architecture. As a result, different new DeTraMe networks may be obtained by integrating Q-Metric Learning into various CNN architectures such as Plain CNN [Spr14] and ResNets [He16a]. The resulting DeTraMe-Nets-based architectures are demonstrated to be more discriminative than generic CNN models.

Although the authors of [Wan15] and [Liu18a] also used a CNN followed by an RNN for respectively solving super-resolution and sense recognition tasks, they directly used LISTA in their model. In turn, our method actually solves the same problem as LISTA. In addition, in [Wan15] and [Liu18a], a sparse representation was jointly learned, while a more discriminative DDL approach is achieved in our work. We also formally derive the linear and RNN-based layer structure from DDL, thus providing a theoretical justification and a rationale to such approaches. This may also open an avenue to new and more creative and performing alternatives.

Our main contributions are summarized below:

• We theoretically transform one-layer dictionary learning into transform learning and Q-Metric learning, and deduce how to convert DDL into DeTraMe-Net.

• Such joint transform learning and Q-Metric learning are successfully and easily implemented
as a tandem of a linear layer and an RNN. A convolutional layer can be chosen for the linear part, and the RNN can also be simplified into a Convolutional-RNN. To the best of our knowledge, this is the first work which makes an insightful bridge between DDL methods and the combination of linear layers and RNNs, with the associated performance gains.

- The transform and Q-Metric learning uses two independent variables, one for the dictionary and the other for the dual frame operator of the dictionary. This bridges the current work to conventional SDL while introducing more discriminative power, and allowing the use of faster learning procedures than the original DL.

- The Q-Metric can also be viewed as a parametric non-separable nonlinear activation function, while in current neural network architectures, very few non-separable nonlinear operators are used (softmax, max pooling, average pooling). As a component of a neural network, it can be flexibly inserted into any network architecture to easily construct a DL layer.

- The proposed DeTraMe-Net is demonstrated to not only improve the discrimination power of DDL, but to also achieve a better performance than state-of-the-art CNNs.

The paper is organized as follows: In Section 4.2, we introduce the required background material. We derive the theoretical basis for our novel approach in Section 4.3. Its algorithmic solution is investigated in Section 4.4. Substantiating experimental results and evaluations are presented in Section 4.5. Finally, we provide some concluding remarks in Section 4.6.

### 4.2 Preliminaries

#### 4.2.1 Notation

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A ), ( \mathbf{a}<em>i ), ( a</em>{i,j} )</td>
<td>A Matrix</td>
</tr>
<tr>
<td>( A^\top ), ( A^{(-1)} )</td>
<td>The transpose and inverse of matrices</td>
</tr>
<tr>
<td>( I )</td>
<td>The Identity Matrix</td>
</tr>
<tr>
<td>( a_{i,j} )</td>
<td>The ( i^{th} ) row and ( j^{th} ) column element of a matrix ( A )</td>
</tr>
<tr>
<td>( \mathbf{a} )</td>
<td>A Vector</td>
</tr>
<tr>
<td>( a_i )</td>
<td>The ( i^{th} ) element of vector ( \mathbf{a} )</td>
</tr>
</tbody>
</table>

#### 4.2.2 Dictionary Learning for Classification

In task-driven dictionary learning [Mai09c], the common method for one-layer dictionary learning classifier is to jointly learn the dictionary matrix \( \mathbf{D} \), the sparse representation \( \mathbf{a} \) of a given vector
x, and the classifier parameter C. Let \( (x_j)_{1 \leq j \leq N} \) be the data and \( (y_j)_{1 \leq j \leq N} \) the associated labels. Task-driven DL can be expressed as finding

\[
\arg\min_{D, (a_j)_{1 \leq j \leq N}, C} \sum_{j=1}^{N} f(x_j, D, a_j) + g(x_j, y_j, D, a_j, C). \tag{4.1}
\]

In SDL, we learn the composition of a dictionary and a sparse reconstruction in order to reconstruct or synthesize the data, hence yielding the standard formulation,

\[
f(x, D, a) = \frac{1}{2} \|x - Da\|_2^2 + \lambda \|a\|_1, \quad \lambda \in (0, +\infty). \tag{4.2}
\]

Alternatively, in ADL, we directly operate on the data using a dictionary, leading to,

\[
f(x, D, a) = \frac{1}{2} \|a - Dx\|_2^2 + \lambda \|a\|_1, \quad \lambda \in (0, +\infty). \tag{4.3}
\]

The term \( g(x, y, D, a, C) \) may correspond to various kinds of loss functions, such as least-squares, cross-entropy, or hinge loss.

### 4.2.3 Deep Dictionary Learning

An efficient DDL approach [Mah19] consists of computing

\[
\hat{y} = \varphi(Cx^{(s)}), \tag{4.4}
\]

where \( \hat{y} \) denotes the estimated label, \( C \) is the classifier matrix, \( \varphi \) is a nonlinear function, and

\[
x^{(s)} = P^{(s)} \circ M_{D^{(s)}} \circ P^{(s-1)} \circ M_{D^{(s-1)}} \circ \cdots \circ P^{(1)} \circ M_{D^{(1)}}(x^{(0)}), \tag{4.5}
\]

where \( \circ \) denotes the composition of operators. For every layer \( r \in \{1, \ldots, s\} \), \( P^{(r)} \) is a reshaping operator, which is a tall matrix. Moreover, \( M_{D^{(r)}} \) is a nonlinear operator computing a sparse representation within a synthesis dictionary matrix \( D^{(r)} \). More precisely, for a given matrix \( D^{(r)} \in \mathbb{R}^{m_r \times k_r} \),

\[
M_{D^{(r)}} : \mathbb{R}^{m_r} \to \mathbb{R}^{k_r}
\]

\[
x \rightarrow \arg\min_{a \in \mathbb{R}^{k_r}} L^R(D^{(r)}, a, x), \tag{4.6}
\]

with

\[
L^R(D^{(r)}, a, x) = \frac{1}{2} \|x - D^{(r)}a\|_F^2 + \lambda \psi_r(a) + \frac{\alpha}{2} \|a\|_2^2 + (d^{(r)})^\top a, \tag{4.7}
\]

where \( (\lambda, \alpha) \in (0, +\infty)^2 \), \( d^{(r)} \in \mathbb{R}^{k_r} \), and \( \psi_r \) is a function in \( \Gamma_0(\mathbb{R}^{k_r}) \), the class of proper lower semicontinuous convex functions from \( \mathbb{R}^{k_r} \) to \( (-\infty, +\infty) \). A simple choice consists in setting \( d^{(r)} \) to
zero, while adopting the following specific form for $\psi_r$;

$$\psi_r = \|\cdot\|_1 + \iota_{[0, +\infty)^r},$$  \hspace{1cm} (4.8)

where $\iota_S$ denotes the indicator function of a set $S$ (equal to zero in $S$ and $+\infty$ otherwise). Note that Eq. (4.6) corresponds to the minimization of a strongly convex function, which thus admits a unique minimizer, so making the operator $M_{D^r}$ properly defined.

### 4.3 Joint Deep Metric and Transform Learning

#### 4.3.1 Proximal interpretation

Our goal here is to establish an equivalent but more insightful solution for $M_D$ in each layer.

**Claim 1:** $M_D$ can be solved by a proximal operator of a transform learning with a metric $Q$:

$$M_D(x) = \text{prox}_{\lambda \psi}^Q (Fx - c).$$  \hspace{1cm} (4.9)

To simplify notation, we omit the superscript which denotes the layer in Eq. (4.6) which, in turn, aims at finding the sparse representation $a$. For every $D \in \mathbb{R}^{m \times k}$, $a \in \mathbb{R}^k$, and $x \in \mathbb{R}^m$, Eq. (4.7) can thus be re-expressed as follows:

$$L^R(D, a, x) = \frac{1}{2} \|a - Fx + c\|_Q^2 + \lambda \psi(a),$$  \hspace{1cm} (4.11)

with

$$Q = D^T D + \alpha I, \quad F = Q^{-1} D^T, \quad c = Q^{-1} d,$$

and $\|\cdot\|_Q = \sqrt{(\cdot)^T Q (\cdot)}$ denotes the weighted Euclidean norm induced by $Q$. Determining the optimal sparse representation $a$ of $x \in \mathbb{R}^m$ is therefore, equivalent to computing the proximity operator in Eq. (4.11), that is Eq. (4.9):

$$M_D(x) = \arg\min_{a \in \mathbb{R}^k} \tilde{L}^R(D, a, x) = \text{prox}_{\lambda \psi}^Q (Fx - c).$$  \hspace{1cm} (4.13)

This thus establishes a re-expression of the solution of the representation procedure as the proximity operator of $\lambda \psi$ within the metric induced by the symmetric definite positive matrix $Q$ [Com10; Cho14]. Furthermore, it shows that the SDL can be equivalently viewed as an ADL formulation.
involving the dictionary matrix $F$, provided that a proper metric is chosen.

### 4.3.2 Multilayer representation

Consequently, by substituting Eq. (4.13) in Eqs. (4.4) and (4.5), the DDL model can be re-expressed in a more concise and comprehensive form as

$$
\hat{y} = \varphi \circ A^{(s+1)} \circ \text{prox}_{\lambda \psi_s} \circ A^{(s)} \circ \text{prox}_{\lambda \psi_{s-1}} \circ \cdots \circ \text{prox}_{\lambda \psi_1} \circ A^{(1)}(x^{(0)}),
$$

(4.14)

where, for $1 \leq r \leq s$, the affine operators $A^{(r)}$ mapping $z^{(r-1)} \in \mathbb{R}^{k_{r-1}}$ to $z^{(r)} \in \mathbb{R}^{k_r}$ by an analysis transform $W^{(r)}$ and a shift term $c^{(r)}$, and explicitly as,

$$
\forall r \in \{1, \ldots, s\}, A^{(r)} : \mathbb{R}^{k_{r-1}} \rightarrow \mathbb{R}^{k_r},
\quad z^{(r-1)} \rightarrow W^{(r)} z^{(r-1)} - c^{(r)}
$$

(4.15)

with $k_0 = m_1$ and

$$
\forall r \in \{2, \ldots, s\},
\quad W^{(r)} = F^{(r)} P^{(r-1)},
\quad W^{(s+1)} = C P^{(s)}
$$

(4.16)

Eq. (4.15) shows that, for each layer $r$, we obtain a structure similar to a linear layer by treating $W^{(r)}$ as the weight operator and $c^{(r)}$ as the bias parameter, which are referred as the Transform learning part in DeTraMe method. In standard Forward Neural Networks (FNNs), the activation functions can be interpreted as proximity operators of convex functions [Com18]. Eq. (4.14) attests that our model is more general, in the sense that different metrics are introduced for these operators. In the next section, we propose an efficient method to learn these metrics in a supervised manner.
4.4 Q-Metric Learning

4.4.1 Prox computation

Reformulation (4.14) has the great advantage to allow us to benefit from algorithmic frameworks developed for FNNs, provided that we are able to compute efficiently

$$\text{prox}_{\lambda \psi}^Q(Z) = \arg\min_{U \in \mathbb{R}^{k \times N}} \frac{1}{2} \|U - Z\|^2_{F, Q} + \lambda \psi'(U),$$

(4.17)

where $\| \cdot \|_{F, Q} = \sqrt{\text{tr}(\cdot (\cdot)^\top)}$ is the Q-weighted Frobenius norm. Hereabove, $Z$ is a matrix where the $N$ samples associated with the training set have been stacked columnwise. A similar convention is used to construct $X$ and $Y$ from $(x_j)_{1 \leq j \leq N}$ and $(y_j)_{1 \leq j \leq N}$. An elastic-net like regularization is chosen by setting $\psi = \| \cdot \|_1 + \iota_{[0, +\infty)}^{k \times N} + \frac{\beta}{2} \| \cdot \|^2_F$ with $\beta \in (0, +\infty)$. We have, in particular, observed that the last quadratic term has a positive influence in increasing stability and avoiding overfitting. As $Q = D^\top D + \alpha I$ in Eq. (4.12), Eq. (C.2) is actually equivalent to solving the following optimization problem:

$$\minimize_{U \in [0, +\infty)^{k \times N}} \frac{1}{2} \|D(U - Z)\|^2_F + \frac{\alpha}{2} \|U - Z\|^2_F + \frac{\beta}{2} \|U\|^2_F + \lambda \|U\|_1.$$  

(4.18)

**Claim 2:** We show next that the solution of Eq. (4.18) is obtained as an iteration of the form:

$$U_{t+1} = \text{ReLU} \left((h_1^\top) \odot Z + \tilde{W}(U_t - Z) - b_1^\top \right).$$

(4.19)

Various iterative splitting methods could be used to find the unique minimizer of the above optimized convex function [Boy04; Kom14]. Our purpose is to develop an algorithmic solution for which classical NN learning techniques can be applied in a fast and convenient manner. By subdifferential calculus, the solution $U$ to the problem (4.18) satisfies the following optimality condition:

$$0 \in Q(U - Z) + \beta U + \lambda \hat{\psi}(U),$$

(4.20)

where $\hat{\psi} = \| \cdot \|_1 + \iota_{[0, +\infty)^{k \times N}}$. Element-wise rewriting of Eq. (4.20) yields, for every $i \in \{1, \ldots, k\}$, and $j \in \{1, \ldots, N\}$,

$$0 \in \sum_{\ell=1}^k q_{i, \ell}(u_{i,j} - z_{\ell,j}) + \beta u_{i,j} + \begin{cases} (-\infty, \lambda] & \text{if } u_{i,j} = 0 \\ \lambda & \text{if } u_{i,j} > 0 \\ \emptyset & \text{if } u_{i,j} < 0. \end{cases}$$

(4.21)

Let us adopt a block-coordinate approach and update the $i$-th row of $U$ by fixing all the other ones.
As $Q$ is a positive definite matrix, $q_{i,j} > 0$ and Eq. (4.21) implies that

$$u_{i,j} = \begin{cases} \frac{q_{i,i} z_i j - v_{i,j}}{q_{i,i} + \beta} & \text{if } q_{i,j} z_{i,j} > (q_{i,i} + \beta) v_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{4.22}$$

where $v_{i,j} = \frac{\lambda + \sum_{1 \leq \ell \leq k} q_{i,\ell} (u_{\ell,j} - z_{\ell,j})}{q_{i,i} + \beta}$. Let

$$\tilde{W} = \left( \frac{q_{i,i} z_{i,j} - v_{i,j}}{q_{i,i} + \beta} \delta_{i=0} \right)_{1 \leq i,j \leq k},$$

$$h = \left( \frac{q_{i,i}}{q_{i,i} + \beta} \right)_{1 \leq i \leq k} \in [0,1]^k,$$

$$b = \left( \frac{\lambda}{q_{i,i} + \beta} \right)_{1 \leq i \leq k} \in [0,+\infty)^k,$$

$$1 = [1, \ldots, 1]^T \in \mathbb{R}^N,$$

where $(\delta_{\ell})_{\ell \in \mathbb{Z}}$ is the Kronecker sequence (equal to 1 when $\ell = 0$ and 0 otherwise). Then, Eq. (4.22) suggests that the elements of $U$ can be globally updated, at iteration $t$, as shown in Eq. (C.1), with $\circ$ denoting the Hadamard (element-wise) product. Note that a similar expression can be derived by applying a preconditioned forward-backward algorithm [Cho14] to Eq. (4.18), where the preconditioning matrix is $\text{Diag}(q_{1,1}, \ldots, q_{k,k})$, which has been detailed in Appendix C. The implementation of the method allowing us to compute the proximity operator in (C.2) is summarized below:

**Algorithm 4** Q-Metric ReLU computation

**Input:** matrix $\tilde{W}$ and $Z$, vectors $h$ and $b$, and maximum iteration number $t_{\text{max}}$

**Output:** Sparse Representation $U^*$

1. Initialize $U_0$ as the null matrix and set $t = 0$
2. **while** not converged and $t < t_{\text{max}}$ **do**
3. Update $U_{t+1}$ according to Eq. (C.1)
4. $t \leftarrow t + 1$

### 4.4.2 RNN implementation

Given $\tilde{W}$, $h$, and $b$, Alg. (4) can be viewed as an RNN structure for which $U_t$ is the hidden variable and $Z$ is a constant input over time. By taking advantage of existing gradient back-propagation techniques for RNNs, $(\tilde{W}, h, b)$ can thus be directly computed in order to minimize the global loss $L$. This shows that, thanks to the re-parameterization in Eq. (C.10), Q-Metric Learning has been recast as the training of a specific RNN.

Note that $Q$ is a $k \times k$ symmetric matrix. In order to reduce the number of parameters and ease
of optimizing them, we choose a block-diagonal structure for $Q$. In addition, for each of the blocks, either an arbitrary or convolutive structure can be adopted. Since the structure of $Q$ is reflected by the structure of $\tilde{W}$, this leads in Eq. (C.1) to fully connected or convolutional layers where the channel outputs are linked to non-overlapping blocks of the inputs. In our experiments on images, Convolutional-RNNs have been preferred for practical efficiency.

### 4.4.3 Training procedure

We have finally transformed our DDL approach in an alternation of linear layers and specific RNNs. This not only simplifies the implementation of the resulting DeTraMe-Net by making use of standard NN tools, but also allows us to employ well-established stochastic gradient-based learning strategies. Let $\rho_t > 0$ be the learning rate at iteration $t$, the simplified form of a training method for DeTraMe-Nets is provided in Alg. 5. The constraints on the parameters of the RNNs have been imposed by projections. In Alg. 5, $P_S$ denotes the projection onto a nonempty closed convex set $S$ and $D_0$ is the vector space of $k \times k$ matrices with diagonal terms equal to 0.

### 4.5 Experiments and Results

In this section, our DeTraMe-Net method is evaluated on three popular datasets, namely CIFAR10 [Kri09], CIFAR100 [Kri09] and Street View House Numbers (SVHN) [Net11]. Since the common NN architectures are plain networks such as ALL-CNN [Spr14] and residual ones, such as ResNet [He16a] and WideResNet [Zag16], we compare DeTraMe-Net with these three respective state-of-the-art architectures.

#### 4.5.1 Architectures

Since we break SDL into two independent linear layer and RNN parts, RNNs can be flexibly inserted into any nonlinear layer of a deep neural network. After choosing convolutional linear layers, we can construct two different architectures when inserting RNN into Plain Networks and residual blocks. One is to replace all the RELU activation layers in PlainNet with Q-Metric ReLU, leading to DeTraMe-PlainNet. Another is to replace the RELU layer inside the block in ResNet by Q-Metric ReLU, giving rise to DeTraMe-ResNet. When replacing all the RELU layers, DeTraMe-PlainNet becomes equivalent to DDL as explained in Section 4.4. When only replacing a single RELU layer in the ResNet architecture, a new DeTraMe-ResNet structure is built. The detailed architectures are illustrated in Appendix D.

For the PlainNet, we use a 9 layer architecture similar to ALL-CNN [Spr14] with dropouts, as listed in Table 4.1. For the ResNet architecture, we follow the setting in [He16a], the first layer is a $3 \times 3$ convolutional layer with 16 filters. 3 residual blocks with output map size of 32, 16, and 8 are then used with 16, 32 and 64 filters for each block. The network ends up with a global average pooling and a fully-connected layer. The parameters listed in Table 4.2 are respectively chosen equal to
Algorithm 5 Deep Metric and Transform Learning

Initialization:
1: for \( r = 1, \ldots, s + 1 \) do
2: \( \text{Randomly initialize } W_0^{(r)}, c_0^{(r)}, W_0^{(r)}, h_0^{(r)}, \text{ and } b_0^{(r)}. \)
3: end for
4: \( t = 0. \)
5: while not converged and \( t < t_{\text{max}} \) do
6: Forward pass:
7: \( U_0^{(0)} = X \)
8: for \( r = 1, \ldots, s + 1 \) do
9: \( Z_t^{(r)} = W_t^{(r)} U_{t-1}^{(r)} - c_t^{(r)} \)
10: if \( r \leq s \) then
11: \( U_t^{(r)} = \text{prox}_{Q_t^{(r)}} (Z_t^{(r)}) \) by Alg. 4
12: end if
13: \( \hat{Y}_t = \varphi (Z_{t+1}^{(s+1)}) \)
14: Loss: \( L' (\theta_t) = L(Y, \hat{Y}_t), \theta_t \): vector of all parameters
15: Backward pass:
16: for \( r = 1, \ldots, s + 1 \) do
17: \( W_{t+1}^{(r)} = W_t^{(r)} - \rho_t \frac{\partial L'}{\partial W_t^{(r)}} (\theta_t) \)
18: \( c_{t+1}^{(r)} = c_t^{(r)} - \rho_t \frac{\partial L'}{\partial c_t^{(r)}} (\theta_t) \)
19: end for
20: for \( r = 1, \ldots, s \) do
21: \( \tilde{W}_{t+1}^{(r)} = \text{Proj}_{B_{\rho \theta}} \left( W_t^{(r)} - \rho_t \frac{\partial L'}{\partial W_t^{(r)}} (\theta_t) \right) \)
22: \( h_{t+1}^{(r)} = \text{Proj}_{B_{\rho \theta}} \left( h_t^{(r)} - \rho_t \frac{\partial L'}{\partial h_t^{(r)}} (\theta_t) \right) \)
23: \( b_{t+1}^{(r)} = \text{Proj}_{B_{\rho \theta}} \left( b_t^{(r)} - \rho_t \frac{\partial L'}{\partial b_t^{(r)}} (\theta_t) \right) \)
24: end for
25: \( t = t + 1 \)
26: end while

\( n = 1, 3, 9, 18, 27 \) for ResNet 8, 20, 56, 110 and 164-layer networks, and we respectively use \( n = 2, q = 4 \) and \( n = 2, q = 8 \) for WideResNet 16-4 and WideResNet 16-8 networks as suggested in [Zag16].

For DeTraMe-Net, we use convolutional RNNs having the same filter size (resp. number of channels) as those in the convolutional layer before. The number of parameters of each model as well as the number of iterations performed in RNNs, are indicated in Table 4.4.

4.5.2 Datasets and Training Settings

CIFAR10 [Kri09] contains 60,000 32 × 32 color images divided into 10 classes. 50,000 images are used for training and 10,000 images for testing. CIFAR100 [Kri09] is also constituted of 32 × 32 color images. However, it includes 100 classes with 50,000 images for training and 10,000 images for testing. SVHN [Net11] contains 630,420 color images with size 32 × 32. 604,388 images are used for training and 26,032 images are used for testing.

For CIFAR datasets, the normalized input image is 32×32 randomly cropped after 4×4 padding on
Table 4.1 Model Description of PlainNet

<table>
<thead>
<tr>
<th>DeTraMe-Net 3-layer</th>
<th>PlainNet 3-layer</th>
<th>PlainNet 6-layer</th>
<th>PlainNet 9-layer</th>
<th>PlainNet 12-layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 × 3 conv 96</td>
<td>3 × 3 conv 96 RELU</td>
<td>3 × 3 conv 96 RELU</td>
<td>3 × 3 conv 96 RELU</td>
<td>3 × 3 conv 96 RELU</td>
</tr>
<tr>
<td>+ Q-Metric: 3 × 3 conv 96</td>
<td>+ Q-Metric: 3 × 3 conv 96</td>
<td>+ Q-Metric: 3 × 3 conv 96 RELU</td>
<td>+ Q-Metric: 3 × 3 conv 96 RELU</td>
<td>+ Q-Metric: 3 × 3 conv 96 RELU</td>
</tr>
</tbody>
</table>

Input 32 × 32 RGB Image with dropout(0.2)

Table 4.2 ResNet Model [He16a]

<table>
<thead>
<tr>
<th>output map size</th>
<th>32 × 32</th>
<th>16 × 16</th>
<th>8 × 8</th>
</tr>
</thead>
<tbody>
<tr>
<td># layers</td>
<td>1 + 2n</td>
<td>2n</td>
<td>2n</td>
</tr>
<tr>
<td>#filters</td>
<td>16</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>WideResNet #filters</td>
<td>16 × q</td>
<td>32 × q</td>
<td>64 × q</td>
</tr>
</tbody>
</table>

Global Average Pooling
Softmax

each sides of the image and random flipping, similarly to [He16a; Zag16]. No other data augmentation is used. For SVHN, we normalize the range of the images between 0 and 1. All the models are trained on an Nvidia V100 32Gb GPU with 128 mini-batch size. The models of both PlainNet and ResNet architectures are trained by SGD optimizer with momentum equal to 0.9 and a weight decay of 5 × 10^{-3}. On CIFAR datasets, the algorithm starts with a learning rate of 0.1. 300 epochs are used to train the models, and the learning rate is reduced at the 150-th and 225-th epochs. On SVHN dataset, a learning rate of 0.01 is used at the beginning and is then divided by 10 at the 80-th and 120-th epochs within a total of 160 epochs. The same settings are used as in [Zag16].

4.5.3 Results

4.5.3.1 DeTraMe-Net vs. DDL

Table 4.3 DeTraMe-Net vs. DDL: Both DeTraMe-Net and DDL with 9-layer follow the ALL-CNN architecture in [Spr14], as listed in the fourth column in Table 4.1.

<table>
<thead>
<tr>
<th>Model</th>
<th># Parameters</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDL 9</td>
<td>1.4M</td>
<td>0.9304</td>
<td>0.6876</td>
</tr>
<tr>
<td>DeTraMe-Net 9</td>
<td>3.0M</td>
<td><strong>0.9340</strong></td>
<td><strong>0.7034</strong></td>
</tr>
</tbody>
</table>
First, we compare our results with those achieved by the DDL approach in [Mah19]. As we

**Table 4.4 CIFAR10, CIFAR100 and SVHN Results.** The dataset with + is trained with simple translation and flipping data augmentation. The (number) is the number of iteration in RNN. All the presented results are re-implemented and run by using the same settings.

<table>
<thead>
<tr>
<th>Network Architectures</th>
<th># Parameters</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
<th>SVHN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original</td>
<td>DeTraMe-Net</td>
<td>Original</td>
<td>DeTraMe-Net</td>
</tr>
<tr>
<td>PlainNet 3-layer</td>
<td>0.094M</td>
<td>0.261M</td>
<td>0.4248</td>
<td>0.8867 (5)</td>
</tr>
<tr>
<td>PlainNet 6-layer</td>
<td>1.016M</td>
<td>1.929M</td>
<td>0.8634</td>
<td>0.9241 (2)</td>
</tr>
<tr>
<td>PlainNet 9-layer</td>
<td>1.370M</td>
<td>2.984M</td>
<td>0.9036</td>
<td>0.9340 (2)</td>
</tr>
<tr>
<td>PlainNet 12-layer</td>
<td>2.366M</td>
<td>3.908M</td>
<td>0.9108</td>
<td>0.9361 (2)</td>
</tr>
<tr>
<td>ResNet 8</td>
<td>0.074M</td>
<td>0.123M</td>
<td>0.8782</td>
<td>0.8941 (3)</td>
</tr>
<tr>
<td>ResNet 20</td>
<td>0.268M</td>
<td>0.413M</td>
<td>0.9214</td>
<td>0.9253 (3)</td>
</tr>
<tr>
<td>ResNet 56</td>
<td>0.848M</td>
<td>0.994M</td>
<td>0.9365</td>
<td>0.9375 (3)</td>
</tr>
<tr>
<td>ResNet 110</td>
<td>1.719M</td>
<td>1.867M</td>
<td>0.9374</td>
<td>0.9377 (2)</td>
</tr>
<tr>
<td>ResNet 164</td>
<td>2.590M</td>
<td>2.738M</td>
<td>0.9359</td>
<td>0.9439 (2)</td>
</tr>
<tr>
<td>WideResNet 16-4</td>
<td>3.583M</td>
<td>5.136M</td>
<td>0.9525</td>
<td>0.9531 (2)</td>
</tr>
<tr>
<td>WideResNet 16-8</td>
<td>10.783M</td>
<td>16.983M</td>
<td>0.9572</td>
<td>0.9579 (2)</td>
</tr>
</tbody>
</table>

break the dictionary and its pseudo inverse into two independent variables, a higher number of parameters is involved in DeTraMe-Net than in [Mah19]. However, DeTraMe-Net presents two main advantages: The first one is a better capability to discriminate: in Table 4.3, compared to DDL, DeTraMe-Net respectively achieves 0.36% and 1.58% improvements on CIFAR10 and CIFAR100 datasets. The second advantage is that DeTraMe-Net is implemented in a network framework, with no need for extra functions to compute gradients at each layer. Moreover, by taking advantage of the developed techniques in neural networks, DeTraMe-Net does not meet the difficulties of sensitivity to initialization and gradient propagation that the original DDL approach faces.

### 4.5.3.2 DeTraMe-Net vs. Generic CNNs

We next compare DeTraMe-Net with generic CNNs with respect to three different aspects: **Accuracy**, **Parameter number** and **Capacity**.

**Accuracy.** As shown in Table 4.4, with the same architecture, using DeTraMe-Net structures achieves an overall better performance than all various generic CNN models do. For PlainNet architecture, DeTraMe-Net increases the accuracy with a median of 4.61% on CIFAR10, 5.91% on CIFAR100 and 0.45% on SVHN, and respectively increases the accuracy of at least 1.53%, 2.16%, 0.13% on these three datasets. For ResNet architecture, DeTraMe-Net also consistently increases the accuracy with a median of 0.10% on CIFAR10, 0.84% on CIFAR100 and 0.10% on SVHN, and at least 0.03%, 0.53%, 0.06% on all datasets.

**Parameter number.** Although, for a given architecture, DeTraMe-Net improves the accuracy, it involves more parameters. However, as demonstrated in Figure 4.2, for a given number of parameters, DeTraMe-Net outperforms the original CNNs over all three datasets. Plots corresponding to DeTraMe-Net for both PlainNet and ResNet architectures are indeed above those associated with
Figure 4.2 Classification accuracy versus number of parameters. The blue color curves are based on ResNet architecture (left axis), while the orange curves are based on PlainNet architecture (right axis). The solid line denotes DeTraMe-Net, while the dash-line denotes the original CNNs. ‘*’ denotes for CIFAR10, ‘o’ denotes for CIFAR100 and ‘+’ denotes for SVHN.

### Capacity

In terms of depth, comparing improvements with PlainNet and ResNet, shows that the shallower the network, the more accurate. It is remarkable that DeTraMe-Net leads to more than 42% accuracy increase for PlainNet 3-layer on CIFAR10, CIFAR100 and SVHN datasets. When the networks become deeper, they better capture discriminative features of the classes, and albeit with smaller gains, DeTraMe-Net still achieves a better accuracy than a generic deep CNN, e.g. around 0.8% higher than ResNet 164 on CIFAR10 and CIFAR100. In terms of width, we use WideResNet-16-4 and WideResNet-16-8 as two reference models, since both of them include 16 layers but have different widths. Table 4.4 shows that increasing width is beneficial to DeTraMe-Net. Since the original models have already achieved excellent performance for CIFAR10 and SVHN, DeTraMe-Nets with various widths show similarly slightly improved accuracies. However, for CIFAR100, enlarging the width for DeTraMe-Net leads to an increase in the accuracy gain from 0.82% to 1.03%.

### 4.6 Conclusion

Starting from a DDL formulation, we have shown that it is possible to reformulate the problem in a standard optimization problem with the introduction of metrics within standard activation operators. This yields a novel Deep Transform and Metric Learning problem. This has allowed us
to show that the original DDL can be performed thanks to a network mixing linear layer and RNN algorithmic structures, thus leading to a fast and flexible network framework for building efficient DDL-based classifiers. Our experiments show that the resulting DeTraMe-Net performs better than the original DDL approach and state-of-the-art generic CNNs.
CHAPTER

5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this dissertation, we first explored a simple integration of feature selection and classifier learning for small datasets and their associated optimized discriminative structure to solve the Classification task. This is referred to as Structured Analysis Dictionary Learning (SADL). To seek an efficient and discriminative solution, we constrained a structured subspace to enhance the discrimination of the analysis dictionary learning (ADL) method, while a classification fidelity was also jointly constrained on its associated sparse representations. Additionally, a linearized ADM method was adopted to efficiently solve our proposed optimization problem. Meanwhile, taking the advantage of the fast encoding property of ADL, the time complexity of both training and testing phases are greatly reduced. Moreover, the multi-cluster version SADL was also purposed. Such Distributed SADL safeguards both discrimination and fast encoding properties of SADL and also yields a very stable solution. We evaluated our SADL and DSADL on five popular datasets, and showed them to not only achieve at least a comparable, but often a better performance than state-of-the-art techniques, with also a superior training and testing times by orders of magnitude.

Secondly, we proposed to optimize the structure for the advanced classification, Zero-shot Classification task. In contrast to conventional Zero-Shot Learning (ZSL) methods, where neither the reconstruction nor the class-specific information are incorporated, we proposed a novel inductive ZSL method by introducing the class-specific information and the reconstructions of both visual and semantic features in a common latent space to increase the discriminative capability and the robustness against the domain shift between seen and unseen classes, which is referred as JCMSPL. The intrinsically invariant structure between visual and semantic features were leveraged, and
a distinct class-to-class matching was involved into such structure. Both the efficient algorithm and its rigorous theoretical analysis were developed and proved. Comprehensive experiments demonstrated that our proposed JCMSPL achieved superior classification performances for both standard and generalized ZSL than other well-established inductive ZSL methods.

Finally, we extended the feature selection procedure and optimized the intrinsic structure among data into a deep end-to-end framework. Starting from a Deep Dictionary Learning (DDL) formulation, we reformulated it into a standard optimization problem with the introduction of metrics within standard activation operators to yield a novel Deep Transform and Metric Learning (DeTraMe) capability. Thanks to a network that mixes linear layer and a Recurrent Neural Network, the original DDL can be performed in a fast and flexible network framework for improving its discrimination and efficiency. Our experiments show that the resulting DeTraMe-Net performed better than the original DDL approach and state-of-the-art generic CNNs. We think that the bridge we established between DDL and Deep Neural Network will help in further understanding and controlling these powerful tools so as to attain better performance and properties.

5.2 Future Work

Our work has further uncovered other potential research avenues. The first is to improve the discrimination ability of the classifier, the innovative ideas in optimized structure learning can be explored along the following three directions:

1. Discrimination Ability based on Extra Constraints
   In our considering neural network, we solved the hard training issues of DDL, to realize afterwards that more constraints related with the discrimination in traditional DL methods can be introduced in deep dictionary learning to improve the performance and leading to the new DNN architectures.

2. Discrimination Ability based on Structures

   - While the usual structures start with one scale, multi-scale can be introduced into the feature selection processing, such as GoogleNet[Sze15].

   - Since self-attention is an important mechanism to capture the image information in human visual system, it is also worthy to leverage the self-attention operator in the entire deep structures to extract the contextual relationship of both intra- and inter-classes.

   - Ensemble/Fusion models can be another prospective direction to improve the feature discrimination and to establish the discriminative deep structures. As different model emphasize on various feature selection, ensemble or fusion of such features can also yield more information and discrimination.
3. Discrimination Ability based on Domain Shift

Although we assume when the data is large enough, the training and testing data would be following the same distribution, there is always some domain gap among the training and testing data as the data is sampled from the real world. Thus, if we could learn a function that map between the training and testing data, the domain gap can be effectively narrowed. The easy version of such mapping function is Data Augmentation methodology, which uses the pre-defined functions, such as shift, rotation, flipping, etc. To take advantage of the Generative Adversarial Network technology, another way is to jointly learn a generator to generate more different images for the classifier to adapt the domain.


APPENDICES
APPENDIX

A PROOF OF THE ALGORITHM 1 IN CHAPTER 2

Take the Lagrangian function

\begin{equation}
L(\Omega, U, Q, W, \varepsilon^{(1)}, \varepsilon^{(2)}, Z^{(1)}, Z^{(2)}) = \frac{1}{2} \| U - \Omega X \|_F^2 + \lambda_1 \| U \|_1 + \frac{\rho_1}{2} \| \varepsilon_1 \|_2^2 + \frac{\rho_2}{2} \| \varepsilon_2 \|_2^2 \\
+ \langle Z^{(1)}, H - QU - \varepsilon_1 \rangle + \langle Z^{(2)}, Y - W QU - \varepsilon_2 \rangle \\
+ \frac{\mu}{2} \| H - QU - \varepsilon_1 \|_2^2 + \frac{\mu}{2} \| Y - W QU - \varepsilon_2 \|_2^2 \\
+ \frac{\delta_1}{2} \| Q \|_2^2 + \frac{\delta_2}{2} \| W \|_2^2 + \frac{\lambda_2}{2} \| \Omega \|_2^2.
\end{equation}

(A.1)

Our algorithm can be written as the one in Alg. 6. Let us proceed by introducing two simple lemmas:

**Lemma 6.** Consider a differentiable function \( f \) with an \( L \)-Lipschitz continuous derivative and another arbitrary convex function \( g \). For any arbitrary point \( x \) define

\[ x^+ = \text{prox}_{\tau g}(x - \tau \nabla f(x)), \]

where \( \tau > 0 \) is a step size and

\[ \text{prox}_{\tau g}(y) = \arg \min_x \frac{1}{2} \| x - y \|_2^2 + \tau g(x). \]
At each iteration \( k + 1 \), compute:

\[
U_{k+1} = \tau \frac{\lambda_{1}}{\mu} \nabla_{U} L_{\omega}(U_{k}, Q_{k}, W_{k}, \Omega, \Theta_{k}, \varepsilon_{k}, \Xi_{k}, Z_{k}),
\]

(A.2)

\[
Q_{k+1} = Q_{k} - \frac{1}{\mu} \nabla_{Q} L(U_{k+1}, Q_{k+1}, W_{k}, \Omega, \Theta_{k+1}, \varepsilon^{(1)}_{k+1}, \varepsilon^{(2)}_{k+1}, Z^{(1)}_{k}, Z^{(2)}_{k}),
\]

(A.3)

\[
W_{k+1} = W_{k} - \frac{1}{\mu} \nabla_{W} L(U_{k+1}, Q_{k+1}, W_{k+1}, \Omega, \Theta_{k+1}, \varepsilon^{(1)}_{k+1}, \varepsilon^{(2)}_{k+1}, Z^{(1)}_{k}, Z^{(2)}_{k}),
\]

(A.4)

\[
\Omega_{k+1} = \arg\min_{\Omega} L(U_{k+1}, Q_{k+1}, W_{k+1}, \Omega, \Theta_{k+1}, \varepsilon^{(1)}_{k+1}, \varepsilon^{(2)}_{k+1}, Z^{(1)}_{k}, Z^{(2)}_{k}),
\]

(A.5)

\[
\varepsilon^{(1)}_{k+1} = \arg\min_{\varepsilon^{(1)}} L(U_{k+1}, Q_{k+1}, W_{k+1}, \Omega_{k+1}, \Theta_{k+1}, \varepsilon^{(1)}_{k+1}, \varepsilon^{(2)}_{k+1}, Z^{(1)}_{k}, Z^{(2)}_{k}),
\]

(A.6)

\[
\varepsilon^{(2)}_{k+1} = \arg\min_{\varepsilon^{(2)}} L(U_{k+1}, Q_{k+1}, W_{k+1}, \Omega_{k+1}, \Theta_{k+1}, \varepsilon^{(1)}_{k+1}, \varepsilon^{(2)}_{k+1}, Z^{(1)}_{k}, Z^{(2)}_{k}),
\]

(A.7)

\[
Z^{(1)}_{k+1} = Z^{(1)}_{k} + \mu(H - Q_{k+1}U_{k+1} - \varepsilon^{(1)}_{k+1}),
\]

(A.8)

\[
Z^{(2)}_{k+1} = Z^{(2)}_{k} + \mu(Y - W_{k+1}Q_{k+1}U_{k+1} - \varepsilon^{(2)}_{k+1}),
\]

(A.9)

Then, we have

\[
F(x^+) - F(x) \leq \left( \frac{L}{2} - \frac{1}{\tau} \right) \|x - x^+\|^2,
\]

where \( F(x) = f(x) + g(x) \).

Proof. Notice that by the definition of the proximal operator prox, there exists a subgradient \( \xi \in \partial g(x^+) \) such that

\[
x^+ = x^- - \tau \xi,
\]

where \( x^- = x - \tau \nabla f(x) \). Then, we have

\[
g(x) \geq g(x^+) + \langle x - x^0, \xi \rangle.
\]

On the other hand,

\[
f(x) \geq f(x^+) + \langle x - x^+, \nabla f(x) \rangle - \frac{L}{2} \|x - x^+\|^2.
\]

Adding the two inequalities yields

\[
F(x) \geq F(x^+) + \langle x - x^+, \nabla f(x) + \xi \rangle - \frac{L}{2} \|x - x^+\|^2.
\]

Now noticing that \( \tau (\nabla f(x) + \xi) = x - x^+ \) completes the proof. \( \Box \)

Lemma 7. Consider a differentiable function \( f \) and a convex function \( g \). Suppose that a point \( x \)
satisfies
\[ \text{prox}_{\tau g}(x - \tau \nabla f(x)) = x. \]

Then, \( x \) is a stationary point of \( F = f + g \), i.e. \( -\nabla f(x) \in \partial g(x) \).

**Proof.** From the definition of the proximal operator there exists a vector \( \xi \in \partial g(x) \) such that \( x = x - \tau \nabla f(x) - \tau \xi \). We conclude that \( -\nabla f(x) = \xi \), which completes the proof. \( \square \)

Next, we make a simple but crucial observation about our algorithm:

**Lemma 8.** For Algorithm 6 the following holds in every iteration \( k \):

\[
Z_{k+1}^{(1)} = \rho_1 \epsilon_{k+1}^{(1)},
\]

\[
Z_{k+1}^{(2)} = \rho_2 \epsilon_{k+1}^{(2)},
\]

and as a result,

\[ \|Z_{k+1}^{(1)} - Z_k^{(1)}\| = \rho_1 \|\epsilon_{k+1}^{(1)} - \epsilon_k^{(1)}\|, \quad (A.10) \]

\[ \|Z_{k+1}^{(2)} - Z_k^{(2)}\| = \rho_2 \|\epsilon_{k+1}^{(2)} - \epsilon_k^{(2)}\|. \quad (A.11) \]

**Proof.** From the \( \epsilon^{(1)} \) update rule (A.6), we have the following optimality condition

\[
\rho_1 \epsilon_{k+1}^{(1)} - Z_k^{(1)} - \mu(H - Q_{k+1}U_{k+1} - \epsilon_{k+1}^{(1)}) = 0. \quad (A.12)
\]

Combining with dual variable \( Z_{k+1}^{(1)} \) update rule (A.8), we obtain

\[
Z_{k+1}^{(1)} = \rho_1 \epsilon_{k+1}^{(1)}. \quad (A.13)
\]

The result for \( Z_{k+1}^{(2)} \) is similarly obtained. \( \square \)

We take \( L_k = L(\Omega_k, U_k, Q_k, W_k, \epsilon_k^{(1)}, \epsilon_k^{(2)}, Z_k^{(1)}, Z_k^{(2)}, \mu_k) \) for \( k = 0, 1, 2, \ldots \) and notice that the change in \( L_k \) can be controlled by the following result:

**Lemma 9.**

\[
L_{k+1} - L_k \\
\leq \left( \frac{\alpha_{k,U}}{2} - \mu \eta_U \right) \|U_{k+1} - U_k\|^2 + \left( \frac{\alpha_{k,Q}}{2} - \mu \eta_Q \right) \|Q_{k+1} - Q_k\|^2 \\
+ \left( \frac{\alpha_{k,W}}{2} - \mu \eta_W \right) \|W_{k+1} - W_k\|^2 - \frac{m_\Omega}{2} \|\Omega_{k+1} - \Omega_k\|^2 \\
+ \left( \frac{\rho_1^2}{\mu} - m_{\epsilon^{(1)}} \right) \|\epsilon_{k+1}^{(1)} - \epsilon_k^{(1)}\|^2 + \left( \frac{\rho_2^2}{\mu} - m_{\epsilon^{(2)}} \right) \|\epsilon_{k+1}^{(2)} - \epsilon_k^{(2)}\|^2, \quad (A.14)
\]

where

\[
\alpha_{k,U} = 1 + \mu \|Q_k^T Q_k + Q_k^T W_k^T W_k Q_k\|_s,
\]
\[ a_{k,Q} = \delta_1 + \mu \| W_k^T W_k \|_2 \| U_{k+1} U_{k+1}^T \|_2, \]
\[ a_{k,W} = \delta_2 + \mu \| Q_{k+1} U_{k+1} U_{k+1}^T Q_{k+1}^T \|_2, \]
\[ m_\Omega = \sigma_{\min}(X X^T), \]
\[ m_\epsilon(1) = \rho_1 + \mu, \quad m_\epsilon(2) = \rho_2 + \mu. \]

**Proof.** Respectively denote by \( \Delta L_{k,U}, \Delta L_{k,Q}, \Delta L_{k,W}, \Delta L_{k,\Omega}, \Delta L_{k,\epsilon(i)}, \Delta L_{k,Y(i)} \) for \( i = 1, 2 \), the change in \( L \) corresponding to the update of \( U, Q, W, \Omega, \epsilon(i) \) and \( Y(i) \) in Eq. (2-9). Notice that

\[ L_{k+1} - L_k = \Delta L_{k,U} + \Delta L_{k,Q} + \Delta L_{k,W} + \Delta L_{k,\Omega} \]
\[ + \Delta L_{k,\epsilon(1)} + \Delta L_{k,\epsilon(2)} + \Delta L_{k,Z(1)} + \Delta L_{k,Z(2)}. \]

Notice that by taking \( f(U) = L_s(\Omega_k, U, Q_k, W_k, \epsilon_k^{(1)}, \epsilon_k^{(2)}, Z_k^{(1)}, Z_k^{(2)}), g(U) = \lambda_1 \| U \|_1 \) and \( \tau = 1/\mu \eta_U \), and recalling Lemma 6, we have

\[ \Delta L_{k,U} \leq \left( \frac{a_{k,U}}{2} - \mu \eta_U \right) \| U_{k+1} - U_k \|^2, \quad (A.15) \]

where we use the fact that \( f(U) \) is quadratic, hence possessing \( a_{k,U} \)-Lipschitz derivatives with \( a_{k,U} \) being the largest singular value of the Hessian. Similarly, by taking \( f(Q) = L(\Omega_k, U_{k+1}, Q_k, \epsilon_k^{(1)}, \epsilon_k^{(2)}, Z_k^{(1)}, Z_k^{(2)}), g(Q) = 0, \tau = 1/\mu \eta_Q \) and

\[ f(W) = L(\Omega_k, U_{k+1}, Q_{k+1}, W_k, \epsilon_k^{(1)}, \epsilon_k^{(2)}, Z_k^{(1)}, Z_k^{(2)}), g(W) = 0, \tau = 1/\mu \eta_W \]

and utilizing Lemma 6, we respectively obtain

\[ \Delta L_{k,Q} \leq \left( \frac{a_{k,Q}}{2} - \mu \eta_Q \right) \| Q_{k+1} - Q_k \|^2, \quad (A.16) \]
\[ \Delta L_{k,W} \leq \left( \frac{a_{k,W}}{2} - \mu \eta_W \right) \| W_{k+1} - W_k \|^2. \quad (A.17) \]

Next, notice that the function \( f(\Omega) = L(\Omega_k, U_{k+1}, Q_{k+1}, W_{k+1}, \epsilon_k^{(1)}, \epsilon_k^{(2)}, Z_k^{(1)}, Z_k^{(2)}) \) is quadratic and \( m_\Omega \) is strongly convex, where \( m_\Omega \) is the smallest singular value of Hessian. Hence,

\[ \Delta L_{k,\Omega} = f(\Omega_k) - \min_\Omega f(\Omega) \geq -\frac{m_\Omega}{2} \| \Omega_{k+1} - \Omega_k \|^2 \quad (A.18) \]

Similarly, taking

\[ f(\epsilon^{(1)}) = L(\Omega_{k+1}, U_{k+1}, Q_{k+1}, W_{k+1}, \epsilon_k^{(1)}, \epsilon_k^{(2)}, Z_k^{(1)}, Z_k^{(2)}) \]

and

\[ f(\epsilon^{(2)}) = L(\Omega_{k+1}, U_{k+1}, Q_{k+1}, W_{k+1}, \epsilon_k^{(1)}, \epsilon_k^{(2)}, Z_k^{(1)}, Z_k^{(2)}), \]

we obtain

\[ \Delta L_{k,\epsilon(i)} \leq -\frac{m_\epsilon(i)}{2} \| \epsilon_{k+1}^{(i)} - \epsilon_k^{(i)} \|^2, \quad i = 1, 2. \quad (A.19) \]
Finally, notice that

\[
\Delta L_{k,Z^{(1)}} = \left( Z_{k+1}^{(1)} - Z_k^{(1)}, H - Q_k U_{k+1} - \epsilon_{k+1}^{(1)} \right)
\]

\[
= \left( Z_{k+1}^{(1)} - Z_k^{(1)}, \frac{1}{\mu} \left( Z_{k+1}^{(1)} - Z_k^{(1)} \right) \right) = \frac{1}{\mu} \left\| Z_{k+1}^{(1)} - Z_k^{(1)} \right\|^2
\]

\[
= \frac{\rho^2_1}{\mu} \left\| \epsilon_{k+1}^{(1)} - \epsilon_k^{(1)} \right\|^2.
\]

Similarly, we obtain

\[
\Delta L_{k,Z^{(2)}} = \frac{\rho^2_2}{\mu} \left\| \epsilon_{k+1}^{(2)} - \epsilon_k^{(2)} \right\|^2.
\]

Summing the inequalities in Eq. (A.15), Eq. (A.16), Eq. (A.17), Eq. (A.18), Eq. (A.19), Eq. (9) and Eq. (A.20) completes the proof.

Now, we have the following theorem:

**Theorem 10.** Suppose that \( \mu \geq \sqrt{2} (\rho_1, \rho_2) \). There exist positive values \( \eta_U^0, \eta_Q^0, \eta_W^0 \) only depending on the initial values such that for \( \eta_U > \eta_U^0, \eta_Q > \eta_Q^0, \eta_W > \eta_W^0 \) the sequence \( \{L_k\}^\infty_{k=1} \) is positive and decreasing, hence convergent.

**Proof.** First define

\[
L_{k,e}(\Omega, U, Q, W) = L(\Omega, U, Q, W, \epsilon_k^{(1)}, \epsilon_k^{(2)}, Z_k^{(1)}, Z_k^{(2)}).
\]

According to Lemma 8, for \( k = 1, 2, \ldots \), we have

\[
L_{k,e} = \frac{1}{2} \left\| U - \Omega X \right\|^2_F + \lambda_1 \left\| U \right\|_1
\]

\[
+ \rho_1 \left( \epsilon_k^{(1)}, H - QU - \epsilon_k^{(1)} \right) + \frac{\mu}{2} \left\| H - QU - \epsilon_k^{(1)} \right\|^2_2 + \frac{\rho_1}{2} \left\| \epsilon_k^{(1)} \right\|^2_2
\]

\[
+ \rho_2 \left( \epsilon_k^{(2)}, Y - WQU - \epsilon_k^{(2)} \right) + \frac{\mu}{2} \left\| Y - WQU - \epsilon_k^{(2)} \right\|^2_2 + \frac{\rho_2}{2} \left\| \epsilon_k^{(2)} \right\|^2_2
\]

\[
+ \frac{\delta_1}{2} \left\| Q \right\|^2_F + \frac{\delta_2}{2} \left\| W \right\|^2_2 + \frac{\lambda_2}{2} \left\| \Omega \right\|^2_F
\]

\[
= \frac{1}{2} \left\| U - \Omega X \right\|^2_F + \lambda_1 \left\| U \right\|_1
\]

\[
+ \frac{\mu}{2} \left\| H - QU - \left( 1 - \frac{\rho_1}{\mu} \right) \epsilon_k \right\|^2_2 + \frac{\rho_1}{2} \left( 1 - \frac{\rho_1}{\mu} \right) \left\| \epsilon_k^{(1)} \right\|^2_2
\]

\[
+ \frac{\mu}{2} \left\| Y - WQU - \left( 1 - \frac{\rho_2}{\mu} \right) \epsilon_k \right\|^2_2 + \frac{\rho_2}{2} \left( 1 - \frac{\rho_2}{\mu} \right) \left\| \epsilon_k^{(2)} \right\|^2_2
\]

\[
+ \frac{\delta_1}{2} \left\| Q \right\|^2_F + \frac{\delta_2}{2} \left\| W \right\|^2_2 + \frac{\lambda_2}{2} \left\| \Omega \right\|^2_F.
\]

Hence, we have \( L_{k,e} \geq 0 \) for \( \mu > \max(\rho_1, \rho_2) \). In particular, we obtain that \( L_k = L_{k,e}(\Omega_k, U_k, Q_k, W_k) \geq 0 \). Now, we use complete (strong) induction to show that \( L_{k+1} \geq L_k \) for \( k = 1, 2, \ldots \). Suppose that this holds for \( k = 1, 2, \ldots, r \). We conclude that \( L_r \leq L_1 \). Now, notice that from (B.9) and the fact that
\( L_t = L_{t,e}(\Omega_t, U_t, Q_t, W_t) \) we obtain for \( \mu > \max\{\rho_1, \rho_2\} \) that

\[
\|Q_t\|^2 \leq \frac{2L_1}{\delta_1}, \quad \|W_t\|^2 \leq \frac{2L_1}{\delta_2},
\]

which leads to the following:

\[
\alpha_{t,U} \leq 1 + \mu \left( \|Q_t\|^2 + \|Q_t\| \|W_t\| \right) \leq 1 + \frac{2L_1 \mu}{\delta_1} \left( 1 + \frac{2L_1}{\delta_2} \right).
\]

According to (A.15), by selecting \( \eta_U > \left[ 2 + \frac{2L_1 \mu}{\delta_1} \right] / 2\mu \), we have that

\[
\Delta L_{t,U} \leq -\frac{1}{2} \|U_{t+1} - U_t\|^2.
\]

which subsequently yields,

\[
L_{t,e}(\Omega_t, U_{t+1}, Q_t, W_t) \leq L_t \leq L_1.
\]

Then according to (B.9) for \( \mu > \max\{\rho_1, \rho_2\} \), we have that

\[
\|U_{t+1}\|_1 \leq \frac{L_1}{\lambda_1}.
\]

We conclude that

\[
\alpha_{t,Q} \leq \delta_1 + \mu \|W_t\|^2 \|U_{t+1}\|_1^2 \leq \delta_1 + \mu \frac{2L_1^2}{\lambda_1 \delta_2}.
\]

Now, by taking \( \eta_U > \left[ 1 + \delta_1 + \frac{2L_1^2}{\lambda_1 \delta_2} \right] / 2\mu \) in (A.16) we have that

\[
\Delta L_{t,Q} \leq -\frac{1}{2} \|Q_{t+1} - Q_t\|^2.
\]

This also results in

\[
L_{t,e}(\Omega_t, U_{t+1}, Q_{t+1}, W_t) \leq L_{t,e}(\Omega_t, U_{t+1}, Q_t, W_t) \leq L_t \leq L_1,
\]

which using (B.9) for \( \mu > \max\{\rho_1, \rho_2\} \) leads to

\[
\|Q_{t+1}\|^2 \leq \frac{2L_1}{\delta_1},
\]

and hence

\[
\alpha_{t,W} \leq \delta_2 + \mu \|Q_{t+1}\|^2 \|U_{t+1}\|_1^2 \leq \delta_2 + \mu \frac{2L_1^2}{\delta_1}.
\]

Now, by choosing \( \eta_W > \left[ 1 + \delta_2 + \frac{2L_1^2}{\lambda_1 \delta_1} \right] / 2\mu \) we conclude from (A.17) that

\[
\Delta L_{t,W} \leq -\frac{1}{2} \|W_{t+1} - W_t\|^2.
\]

(A.24)
Finally, by choosing $\mu > \sqrt{2} \max\{\rho_1, \rho_2\}$, we obtain from Lemma 9 that

$$
L_{t+1} - L_t \leq -\frac{1}{2} \|U_{t+1} - U_t\|_2^2 + \frac{1}{2} \|Q_{t+1} - Q_t\|_2^2 - \frac{1}{2} \|W_{t+1} - W_t\|_2^2
- \frac{\mu_0}{2} \|W_{t+1} - U_t\|_2^2 - \frac{\rho_1}{2} \|\varepsilon_{t+1}^{(1)} - \varepsilon_t^{(1)}\|_2^2 - \frac{\rho_2}{2} \|\varepsilon_{t+1}^{(2)} - \varepsilon_t^{(2)}\|_2^2.
$$

(A.25)

We conclude that $L_{t+1} \leq L_t$ which completes the proof. 

We finally obtain the following corollary which clarifies the statement and gives the proof of our main result in Theorem 10:

**Corollary 1.** Suppose that $\mu \geq \sqrt{2}\{\rho_1, \rho_2\}$. There exist positive values $\eta_U^0, \eta_Q^0, \eta_W^0, R$ only depending on the initialization such that for $\eta_U > \eta_U^0, \eta_Q > \eta_Q^0, \eta_W > \eta_W^0$ the sequence

$$
\{\Theta_k = (\Omega_k, U_k, Q_k, W_k, \varepsilon_k^{(1)}, \varepsilon_k^{(2)}, Z_k^{(1)}, Z_k^{(2)})\}_{k=1}^{\infty}
$$

satisfies the following:

1. The parameters for $k = 0, 1, 2, ...$ are bounded by $R$, i.e

   $$
   \|\Theta_k\| = \max\left\{\|\Omega_k\|, \|U_k\|, \|Q_k\|, \|W_k\|, \|\varepsilon_k^{(1)}\|, \|\varepsilon_k^{(2)}\|, \|Z_k^{(1)}\|, \|Z_k^{(2)}\|\right\} < R.
   $$

   Hence, they are confined in a compact set.

2. Any convergence subsequence of $\{\Theta_k\}$ converges to a point $\Theta^* \in S$.

3. $\text{dist}(\Theta_k, S)$ converges to zero, where

   $$
   \text{dist}(\Theta, S) = \min_{\Theta' \in S} \|\Theta' - \Theta\|.
   $$

**Proof.** 1) is simply obtained by noticing (B.9) and the fact that $L_{k+1}(\Omega_k, U_k, Q_k, W_k) = L_k \leq L_1$, since $\{L_k\}$ is decreasing. For 2), note that since the sequence $\{L_k\}$ is convergent, we have $\lim_{k \to \infty} L_{k+1} - L_k = 0$, which according to (A.25) yields

$$
\lim_{k \to \infty} \|U_{k+1} - U_k\|_2^2 = \lim_{k \to \infty} \|Q_{k+1} - Q_k\|_2^2 = \lim_{k \to \infty} \|W_{k+1} - W_k\|_2^2
= \lim_{k \to \infty} \|\Omega_{k+1} - \Omega_k\|_2^2 = \|\varepsilon_{k+1}^{(i)} - \varepsilon_k^{(i)}\|_2^2 = 0,
$$

for $i = 1, 2$. Also from Lemma 8 we have that

$$
\lim_{k \to \infty} \|Z_{k+1}^{(i)} - Z_k^{(i)}\|_2^2 = 0.
$$

We conclude that

$$
\lim_{k \to \infty} \left\| \tau \frac{\lambda_1}{\mu_0 1 - \mu \eta_U} \nabla_U L_{t}(U_k, Q_k, W_k, \bar{\Omega}_t, \varepsilon_k^{(1)}, \varepsilon_k^{(2)}, Z_k^{(1)}, Z_k^{(2)}) - U_k \right\|_2^2 = 0,
$$

82
\[
\lim_{k \to \infty} \left\| \nabla Q L(U_k, Q_k, W_k, \Omega_k, \epsilon_k, Z_k) \right\|_2 = 0,
\]

\[
\lim_{k \to \infty} \left\| \nabla W L(U_k, Q_k, W_k, \Omega_k, \epsilon_k, Z_k) \right\|_2 = 0,
\]

\[
\lim_{k \to \infty} \left\| \nabla_{\Theta} L(\Theta_k) \right\|_2 = 0.
\]

Moreover, note that the Lagrangian \( L \) is \( L_\Omega \)-second order Lipschitz with respect to \( \Omega \) (fixing the rest) with \( L_\Omega = \|X X^T\|_\Omega \). We obtain that

\[
\left\| \nabla_{\Theta} L(U_k, Q_k, W_k, \Omega_k, \epsilon_k, Z_k) \right\|_2 \leq \frac{L_\Omega^2 \|\Omega_k - \Omega\|_2}{2},
\]

which yields

\[
\lim_{k \to \infty} \left\| \nabla_{\Theta} L(U_k, Q_k, W_k, \Omega_k, \epsilon_k, Z_k) \right\|_2 = 0.
\]

Similarly, we obtain

\[
\lim_{k \to \infty} \left\| \nabla_{\epsilon} L(U_k, Q_k, W_k, \Omega_k, \epsilon_k, Z_k) \right\|_2 = 0,
\]

\[
\lim_{k \to \infty} \left\| \nabla_{\epsilon} L(U_k, Q_k, W_k, \Omega_k, \epsilon_k, Z_k) \right\|_2 = 0.
\]

Now, consider a subsequence of \( \{\Theta_k\} \) converging to a point \( \Theta_* = (\Omega_*, U_*, Q_*, W_*, \epsilon_*^{(1)}, \epsilon_*^{(2)}, Z_*^{(1)}, Z_*^{(2)}) \). Since the argument of the above limits are continuous, we obtain

\[
\tau \frac{\lambda_k}{\mu_k} \left( U_* - \frac{1}{\mu_k} \nabla_{\epsilon} L_\Theta(\Theta_*), \Omega_* \right) - U_* = 0,
\]

\[
\nabla Q L(\Theta_*) = 0, \quad \nabla W L(\Theta_*) = 0, \quad \nabla_{\epsilon} L(\Theta_*) = 0,
\]

\[
\nabla_{Z^{(1)}} L(\Theta_*) = H - Q_* U_* - \epsilon_*^{(1)} = 0,
\]

\[
\nabla_{Z^{(2)}} L(\Theta_*) = Y - W_* Q_* U_* - \epsilon_*^{(2)}.
\]

According to Lemma 7, we conclude that \( \Theta_* \in S \). For 3), suppose that the claim is not true. Then, according to 1) there exists a convergent subsequence of \( \{\Theta_k\} \) which is \( \gamma \)-distant from \( S \), i.e.,

\[
\text{dist}(\Theta_k, S) = \gamma > 0.
\]

Then, the convergence point is also \( \gamma \)-distant from \( S \) which contradicts 2) and completes the proof.

\[\Box\]
From Algorithm 1 in our paper, our primal problem can be written as follows:

\[
f(A, B, C) = \frac{1}{2} \|AX_s - C\|_F^2 + \frac{\lambda_1}{2} \|BY_s - C\|_F^2 \\
+ \frac{\lambda_2}{2} \|C - H\|_F^2 + \frac{\lambda_3}{2} \|X_s - A^TC\|_F^2 + \frac{\lambda_4}{2} \|Y_s - B^TC\|_F^2.
\] (B.1)

It may also be rewritten as Algorithm 7 below.

**Algorithm 7** Joint Concept Matching-Space Projection Learning

At each iteration \(t+1\), compute:

\[
A_{t+1} = \arg\min_A f(A_t, B_t, C_t); \tag{B.2}
\]

\[
B_{t+1} = \arg\min_B f(A_{t+1}, B_t, C_t); \tag{B.3}
\]

\[
C_{t+1} = \arg\min_C f(A_{t+1}, B_{t+1}, C_t); \tag{B.4}
\]

We take \(f_t = f(A_t, B_t, C_t)\) for \(t = 0, 1, 2, \ldots\) and note that the change in \(f_t\) can be controlled by the following result:
Theorem 11. Denote the following:

\[ f_{t+1} - f_t \leq \frac{-m_A^t}{2} \|A_{t+1} - A_t\|^2 - \frac{m_B^t}{2} \|B_{t+1} - B_t\|^2 \]

\[-\frac{m_C^t}{2} \|C_{t+1} - C_t\|^2, \quad (B.5)\]

where

\[ m_A^t = \sigma_{\min}(\lambda_3 C_t^T C_t + X_s^T X_s) \geq \sigma_{\min}(X_s^T X_s) > 0, \]

\[ m_B^t = \sigma_{\min}(\lambda_4 C_t^T C_t + \lambda_1 Y_s^T Y_s) \geq \sigma_{\min}(\lambda_1 Y_s^T Y_s) > 0, \]

\[ m_C^t = \sigma_{\min}((1 + \lambda_1 + \lambda_2)I + \lambda_3 A_{t+1}^T A_{t+1} + \lambda_3 B_{t+1}^T B_{t+1}) \geq 1. \]

The sequence \( \{f_t\}_{t=1}^{\infty} \) is positive and decreasing, hence convergent.

**Proof.** Respectively denote by \( \Delta f_{t,A}, \Delta f_{t,B}, \Delta f_{t,C} \), the change in \( f \) corresponding to the update of \( A, B, C \) in Eq. (B.2) - (B.4). Notice that

\[ f_{t+1} - f_t = \Delta f_{t,A} + \Delta f_{t,B} + \Delta f_{t,C}. \]

The function \( g(A) = f(A, B, C_t) \) is quadratic and is \( m_A^t \)-strongly convex, where \( m_A^t \) is the smallest singular value of a Hessian. Hence,

\[ \Delta f_{t,A} = g(A_{t+1}) - \min_A g(A) \leq -\frac{m_A^t}{2} \|A_{t+1} - A_t\|^2. \quad (B.6) \]

Similarly, for updating \( B \) and \( C \), we obtain

\[ \Delta f_{t,B} = g(B_{t+1}) - \min_B g(B) \leq -\frac{m_B^t}{2} \|B_{t+1} - B_t\|^2, \quad (B.7) \]

\[ \Delta f_{t,C} = g(C_{t+1}) - \min_C g(C) \leq -\frac{m_C^t}{2} \|C_{t+1} - C_t\|^2. \quad (B.8) \]

Summing the inequalities in Eq. (B.6), Eq. (B.7), and Eq. (B.8) then Eq. (B.5) is proved.

Our primal loss functional is as follows:

\[ f_{t,e}(A, B, C) = \frac{1}{2} \|AX_s - C\|^2_F + \frac{\lambda_1}{2} \|BY_s - C\|^2_F \]

\[ + \frac{\lambda_2}{2} \|C - H\|^2_F + \frac{\lambda_3}{2} \|X_s - A^T C\|^2_F + \frac{\lambda_4}{2} \|Y_s - B^T C\|^2_F. \quad (B.9) \]

Hence, we have \( f_{t,e} \geq 0 \). In particular, we obtain \( f_{t+1} = f_{t,e}(A_{t+1}, B_{t+1}, C_{t+1}) \geq 0 \). Now, we use an induction to show that \( f_{t+1} \leq f_t \) for \( t = 1, 2, ..., \). Suppose that this holds for \( t = 1, 2, ..., k \). We conclude that \( f_t \leq f_1 \). Since \( m_A^t > 0, m_B^t > 0 \) and \( m_C^t \geq 1, \forall t \), according to Eq. (B.5), we conclude that \( f_{t+1} \leq f_t \leq f_1 \) which completes the proof. □
We finally obtain the following corollary which clarifies the statement and gives the proof of our main result in Theorem 10: Corollary

**Corollary 2.** The sequence \( \{ \Theta_t = (A_t, B_t, C_t) \}_{t=1}^{\infty} \) satisfies the following:

a. The parameters for \( t = 0, 1, 2, \ldots \) are bounded by \( R \) which only depends on the initialization, i.e.,

\[
\|\Theta_t\| = \max \{ \|A_t\|, \|B_t\|, \|C_t\| \} < R.
\]

Hence, they are confined in a compact set.

b. Any convergent subsequence of \( \{ \Theta_t \} \) converges to a point \( \Theta^* \in Q \).

c. \( \text{dist}(\Theta_t, Q) \) converges to zero, where

\[
\text{dist}(\Theta, Q) = \min_{\Theta' \in Q} \|\Theta' - \Theta\|.
\]

**Proof.** Part a is simply obtained by noticing (B.9) and the fact that \( f_{t+e}(A_t, B_t, C_t) = f_t \leq f_1 \), since \( \{ f_t \} \) is decreasing. For part b, note that since the sequence \( \{ f_t \} \) is convergent, we have

\[
\lim_{t \to \infty} f_{t+1} - f_t = 0,
\]

which according to (B.5) yields

\[
\lim_{t \to \infty} \|A_{t+1} - A_t\|_2 = \lim_{t \to \infty} \|B_{t+1} - B_t\|_2 = \lim_{t \to \infty} \|C_{t+1} - C_t\|_2.
\]

Moreover, note that the loss function \( f \) is \( f_A \)—second order Lipschitz with respect to \( A \) (fixing the rest) with \( f_A = \|\lambda_3 C^T C_t + X^T X_s\|_r \). We obtain that

\[
\|\nabla_A f(A_t, B_t, C_t)\|_2 \leq f^2_A \|A_{t+1} - A_t\|_2,
\]

which yields

\[
\lim_{t \to \infty} \|\nabla_A f(A_t, B_t, C_t)\|_2 = 0.
\]

Similarly, we obtain

\[
\lim_{t \to \infty} \|\nabla_B f(A_{t+1}, B_t, C_t)\|_2 = 0,
\]

\[
\lim_{t \to \infty} \|\nabla_C f(A_{t+1}, B_{t+1}, C_t)\|_2 = 0.
\]

Now, take a subsequence of \( \{ \Theta_t \} \) converging to a point \( \Theta^* = (A^*, B^*, C^*) \). Since the arguments of the above limits are continuous we obtain

\[
\nabla_A f(\Theta^*) = 0, \quad \nabla_B f(\Theta^*) = 0, \quad \nabla_C f(\Theta^*) = 0.
\]

Therefore, \( \Theta^* \in Q \). For part c, suppose that the claim is not true. Then, according to part a there exists a convergent subsequence of \( \{ \Theta_t \} \) which is \( \epsilon \)—distant from \( Q \), i.e., \( \text{dist}(\Theta_k, Q) = \epsilon > 0 \). Then, the convergence point is also \( \epsilon \)—distant from \( Q \) which contradicts part b and completes the proof. \( \square \)
APPENDIX

C

ALTERNATIVE DERIVATION OF ALGORITHM 4 IN CHAPTER 4

We have presented in our paper a simple approach for deriving the recursive model:

\[ U_{t+1} = \text{ReLU} \left( (h1^\top) \odot Z + \tilde{W}(U_t - Z) - b1^\top \right), \quad (C.1) \]

in order to compute

\[ \text{prox}^Q_{\lambda \psi}(Z) = \arg\min_{U \in \mathbb{R}^{k \times N}} \frac{1}{2} \| U - Z \|_F^2 + \lambda \psi(U). \quad (C.2) \]

We propose an alternative approach which is based on the classical forward-backward algorithm for solving the nonsmooth convex optimization problem in (C.2). The \( t \)-th iteration of the preconditioned form of this algorithm reads

\[ U_{t+1} = \text{prox}^\Theta_{\gamma \lambda \psi}(U_t - \gamma \Theta^{-1} Q(U_t - Z)) \quad (C.3) \]

where \( \gamma \) is a positive stepsize and \( \Theta \) is a preconditioning symmetric definite positive matrix, and \( U_0 \in \mathbb{R}^{k \times N} \). The algorithm is guaranteed to converge to the solution to (C.2) provided that

\[ \gamma < \frac{2}{\| \Theta^{-1/2} Q \Theta^{1/2} \|_S}, \quad (C.4) \]

where \( \| \cdot \|_S \) denotes the spectral norm. Eq. (C.3) can be reexpressed as

\[ U_{t+1} = \text{prox}^\Theta_{\gamma \lambda \psi}(I - \gamma \Theta^{-1} Q(U_t - Z) + Z). \quad (C.5) \]
Assume now that $\Theta$ is a diagonal matrix $\text{Diag}(\theta_1, \ldots, \theta_k)$ where, for every $i \in \{1, \ldots, k\}$, $\theta_i > 0$. When the sparsity promoting penalization is chosen equal to
\[
\psi = \| \cdot \|_1 + t_{[0, +\infty)} \cdot \| \|_F^2 + \frac{\beta}{2\lambda} \| \cdot \|_F^2,
\] (C.6)
the proximity operator involved in (C.5) simplifies as
\[
\forall U = (u_{i,j})_{1 \leq i \leq k, 1 \leq j \leq N} \in \mathbb{R}^{k \times N},
\prox_{\gamma \lambda \psi}^\Theta = \left( \prox_{\gamma \lambda \theta^{-1}_i \rho}^\theta (u_{i,j}) \right)_{1 \leq i \leq k, 1 \leq j \leq N} (C.7)
\]
where $\rho = \lambda | \cdot | + t_{[0, +\infty)} + \frac{\beta}{2} (\cdot)^2$. In addition, for every $u \in \mathbb{R}$ and $i \in \{1, \ldots, k\}$,
\[
\prox_{\gamma \lambda \theta^{-1}_i \rho}^\theta (u) = \arg\min_{v \in [0, +\infty)} \theta_i (v - u)^2 + \gamma \lambda |v| + \frac{\beta}{2} v^2. (C.8)
\]
After some simple algebra, this leads to
\[
\prox_{\gamma \lambda \theta^{-1}_i \rho}^\theta (u) = \text{ReLU} \left( \frac{\theta_i}{\theta_i + \gamma \beta} u - \frac{\gamma \lambda}{\theta_i + \gamma \beta} \right). (C.9)
\]
Altogether (C.5), (C.7), and (C.9) allow us to recover an update equation of the form (C.1), where
\[
\hat{W} = (\Theta + \gamma \beta I)^{-1}(\Theta - \gamma Q),
\]
\[
h = \left( \frac{\theta_i}{\theta_i + \gamma \beta} \right)_{1 \leq i \leq k},
\]
\[
b = \left( \frac{\gamma \lambda}{\theta_i + \gamma \beta} \right)_{1 \leq i \leq k}. (C.10)
\]
Note that, if $\gamma = 1$ and, for every $i \in \{1, \ldots, k\}$, $\theta_i = q_i, i$, $\hat{W}$ is a matrix with zeros on its main diagonal.
D.1 DeTraMe-PlainNet

To replace all the RELU activation layers in PlainNet with Q-Metric ReLU leads to DeTraMe-PlainNet. Since all the RELU layers are replaced by Q-Metric ReLu, DeTraMe-PlainNet becomes equivalent to DDL.

D.2 DeTraMe-ResNet

Replacing the RELU layer inside the block in ResNet by Q-Metric ReLU, allows us to build a new structure called DeTraMe-ResNet.

In our experiments, for ResNet architecture, the RNN part accounting for Q-Metric learning, makes use of $3 \times 3$ filters.
Figure D.1 Architectures of PlainNet vs. DeTraMe-PlainNet

Figure D.2 Architectures of ResNet vs. DeTraMe-ResNet