ON TRANSFER LEARNING TECHNIQUES FOR MACHINE LEARNING

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Debasmit Das

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. C. S. George Lee, Chair

School of Electrical and Computer Engineering

Dr. Stanley H. Chan

School of Electrical and Computer Engineering

Dr. Guang Lin

Department of Mathematics

Dr. Guang Cheng

Department of Statistics

Approved by:

Dr. Dimitrios Peroulis

Head of the School of Electrical and Computer Engineering

This thesis is dedicated to my wife and my father.

ACKNOWLEDGMENTS

This research would not have been possible without the guidance of my major advisor, Professor C. S. George Lee, who has tirelessly discussed with me weekly on my research. As I was desperately looking for a thesis topic and initially working on some research ideas, he suggested that I change gears and investigate into different transfer learning methodologies. The decision turned out to be a blessing as transfer learning is now well regarded in the machine learning research community as a nascent and ever-expanding field with diverse application scenarios.

I would also like to thank my advisory committee, which includes Professor Stanley Chan, Professor Guang Lin and Professor Guang Cheng. They provided constructive criticism on my work that better shaped my research direction during later stages of my graduate studies. I would like to mention my colleagues in ARTLab – Jihoon Moon, Yue Cao, Sanghyun Cho and Ye Shi – who all provided a wonderful environment to discuss research ideas and to get necessary feedback.

I would also like to express my gratitude to my father, Tapas Kumar Das, for his moral support throughout my graduate program. He always believed in my ability to get a doctorate in engineering right from my school days. I would like to thank my beloved wife, Shohini Roy, for accompanying me and motivating me to complete this dissertation.

I would like to acknowledge the financial support from National Science Foundation for funding machine learning activities at ARTLab. This work is supported in part by the National Science Foundation Grant IIS-1813935. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation. I also gratefully acknowledge the support of NVIDIA Corporation for the donation of a TITAN XP GPU used for this research. Finally, I would like to thank the School of

Electrical and Computer Engineering at Purdue University for providing me teaching assistantship during the initial period of my doctoral studies.

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

ABSTRACT

Das, Debasmit Ph.D., Purdue University, May 2020. On Transfer Learning Techniques for Machine Learning. Major Professor: C. S. George Lee.

Recent progress in machine learning has been mainly due to the availability of large amounts of annotated data used for training complex models with deep architectures. Annotating this training data becomes burdensome and creates a major bottleneck in maintaining machine-learning databases. Moreover, these trained models fail to generalize to new categories or new varieties of the same categories. This is because new categories or new varieties have data distribution different from the training data distribution. To tackle these problems, this thesis proposes to develop a family of transfer-learning techniques that can deal with different training (source) and testing (target) distributions with the assumption that the availability of annotated data is limited in the testing domain. This is done by using the auxiliary data-abundant source domain from which useful knowledge is transferred that can be applied to data-scarce target domain. This transferable knowledge serves as a prior that biases target-domain predictions and prevents the target-domain model from overfitting. Specifically, we explore structural priors that encode relational knowledge between different data entities, which provides more informative bias than traditional priors. The choice of the structural prior depends on the information availability and the similarity between the two domains. Depending on the domain similarity and the information availability, we divide the transfer learning problem into four major categories and propose different structural priors to solve each of these sub-problems.

This thesis first focuses on the unsupervised-domain-adaptation problem, where we propose to minimize domain discrepancy by transforming labeled source-domain data to be close to unlabeled target-domain data. For this problem, the categories

remain the same across the two domains and hence we assume that the structural relationship between the source-domain samples is carried over to the target domain. Thus, graph or hyper-graph is constructed as the structural prior from both domains and a graph/hyper-graph matching formulation is used to transform samples in the source domain to be closer to samples in the target domain. An efficient optimization scheme is then proposed to tackle the time and memory inefficiencies associated with the matching problem. The few-shot learning problem is studied next, where we propose to transfer knowledge from source-domain categories containing abundantly labeled data to novel categories in the target domain that contains only few labeled data. The knowledge transfer biases the novel category predictions and prevents the model from overfitting. The knowledge is encoded using a neural-network-based prior that transforms a data sample to its corresponding class prototype. This neural network is trained from the source-domain data and applied to the target-domain data, where it transforms the few-shot samples to the novel-class prototypes for better recognition performance. The few-shot learning problem is then extended to the situation, where we do not have access to the source-domain data but only have access to the source-domain class prototypes. In this limited information setting, parametric neural-network-based priors would overfit to the source-class prototypes and hence we seek a non-parametric-based prior using manifolds. A piecewise linear manifold is used as a structural prior to fit the source-domain-class prototypes. This structure is extended to the target domain, where the novel-class prototypes are found by projecting the few-shot samples onto the manifold. Finally, the zero-shot learning problem is addressed, which is an extreme case of the few-shot learning problem where we do not have any labeled data in the target domain. However, we have high-level information for both the source and target domain categories in the form of semantic descriptors. We learn the relation between the sample space and the semantic space, using a regularized neural network so that classification of the novel categories can be carried out in a common representation space. This same neural network is then used in the target domain to relate the two spaces. In case we want to generate data for

the novel categories in the target domain, we can use a constrained generative adversarial network instead of a traditional neural network. Thus, we use structural priors like graphs, neural networks and manifolds to relate various data entities like samples, prototypes and semantics for these different transfer learning sub-problems. We explore additional post-processing steps like pseudo-labeling, domain adaptation and calibration and enforce algorithmic and architectural constraints to further improve recognition performance. Experimental results on standard transfer learning image recognition datasets produced competitive results with respect to previous work. Further experimentation and analyses of these methods provided better understanding of machine learning as well.

1. INTRODUCTION

1.1 Motivation and Objectives

Machine Learning (ML) is a branch of Artificial Intelligence (AI), which studies algorithms and procedures to produce decision models for a task using past data obtained in carrying out the task. These decision models are therefore not explicitly programmed for the task but are trained and built from previously obtained training samples. Thus, ML closely resembles human learning, which is based on accumulating knowledge from previous experiences. On the other hand, AI is a broader sub-field of computer science, the goal of which is to develop artificial systems that can perform tasks that normally require human-level intelligence to perform. Examples of such tasks can be visual understanding, speech recognition, natural language translation, etc. Historically, AI has fascinated and disappointed computer scientists for over half a century and therefore has an eventful past with rising and falling interests throughout. Therefore, we mention the time-line of AI research and how ML became popular in mainstream AI.

The first account of interest in AI came about with the proposal of Alan Turing's learning machine in 1950 [1]. The machine was supposed to be an intelligent system that had the capability of playing the *imitation game*. The imitation game would form the basis of the Turing test, where a human would be unable to distinguish between a human and an AI system based on conversations. It has been over more than half a century but still there has been no record of any AI system that had passed the Turing Test completely.

The next major event in AI research was the discovery of the Perceptron by Frank Rosenblatt in 1957 [2]. It was the first instance of a learning model that was able to carry out binary classification and so it generated lot of excitement in the

AI community. However, the Perceptron, being only a single-layered neural network, failed to solve the XOR classification problem as demonstrated by Minsky and Papert in 1969 [3]. This prevented further research into neural networks at that time. It also set the basis for the first AI winter in the 1970's, which resulted in abandonment and funding cuts in AI research.

The 1980s saw a surge in interest in AI because of a number of discoveries. The neocognitron [4], a type of neural network, was invented by Kunihiko Fukushima in 1980. This model served as an inspiration for the development of the Convolutional Neural Network (CNN) [5] later on. John Hopfield also popularized the Hopfield Network [6], a recurrent type of neural network with associative memory-like functions. This decade also saw the application of using back-propagation of errors for training multilayer neural networks [7]. However, given the technology of the time, with limited computational power and less amount of available training data, deeper neural networks were not realizable. This led to the rejection of connectionist-based models for almost two decades.

From the early 1990s to the late 2000s saw the development of a lot of alternative models of machine learning. This included the invention of Random Forests [8], Support-Vector Machines (SVM) [9] and the Long-Short-Term Memory (LSTM) [10] module for recurrent neural networks. The LSTM module was developed to take care of the vanishing gradient problem associated with the optimization of recurrent neural networks for sequential data. Also, the CNN was invented in 1998 [5]. Though it was structurally similar to the Neocognitron [4], the CNN could be trained using the back-propagation algorithm. However, most of these alternative methods failed to produce human-level performance for AI perception tasks. Even the CNN architecture could not be scaled up due to the computational deficiencies of the time.

The generalization performance of machine learning models on unseen data generally increases with larger amount of training data. In the 2000s, people had access to lots of publicly generated datasets and so machine-learning models were able to produce better recognition performance. Availability of abundant data was due to

the fact that the Internet became accessible to more people and sensors attached to different devices could readily upload data to the web. Photos from mobile cameras could easily be uploaded to the Internet and people could have access to them. The abundance of images on the web helped the creation of an organized database for image recognition known as the ImageNet [11]. This database consisted of around 21K categories with more than 1K images per category for dominant classes. Also, advances in the processing power of Graphic Processing Units (GPU) allowed CNNs to be trained in parallel within a reasonable amount of time. The breakthrough came in 2012, when a deep neural network architecture commonly known as *AlexNet* [12] was proposed. This deep architecture, when trained and tested on ImageNet using a GPU, produced results much better than previous approaches. The successful application of deep neural networks using a large amount of labeled data lead to the development and applications in various domains. For example, in 2012, a team from Google Brain produced a neural network that learned to recognize cats by watching videos on Youtube [13]. In 2014, Facebook researchers used deep learning to create a face recognition system [14] that produced an accuracy of around 97 percent. In 2016, Google also created an AI known as AlphaGo [15] that beat professional human players in the game of Go. There are many more examples [16–19] that show that deep learning can be expanded to other application domains as well.

Even though deep-learning-based AI has produced amazing results, the question still remains whether it has really lived up to the level of human intelligence. In fact, if one thinks deeply, machine intelligence has a long way to go in terms of *efficiency*. Most of these current methods in machine learning are resource intensive - they require lots of annotated data, they take up significant memory and they consume lot of energy. As a result, these methods cannot be used for resource-constrained scenarios, e.g., on mobile devices, in changing environments without annotations and in recognizing new and rare objects, etc. Hence, there is a need to pursue machine-learning algorithms and architectures that strive to consume less resources, i.e., require less data-labels, require less memory and less energy. The goal of producing efficient

machine-learning methods can be achieved with two different methods – develop training algorithms with less amount of labeled data or produce models with less number of parameters. Accordingly, most efficient machine-learning methods can be broadly classified into data-efficient methods and model-efficient methods as shown in Fig. 1.1.

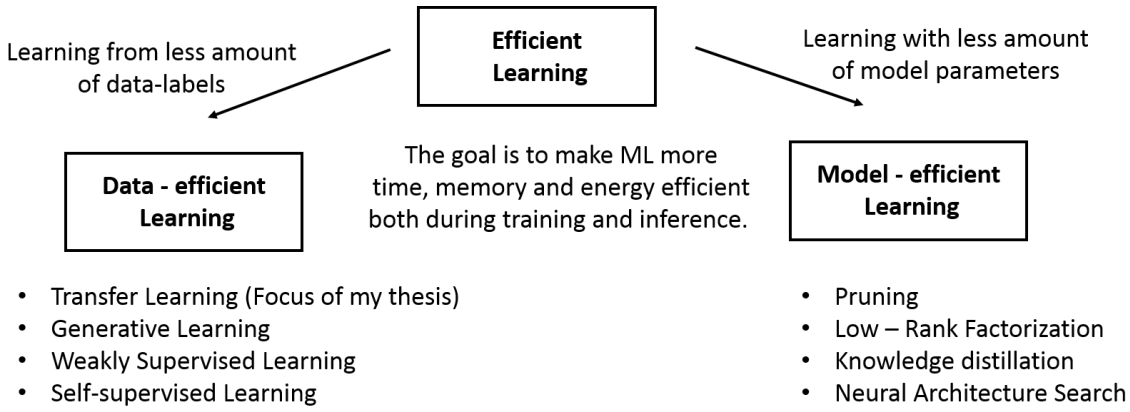


Fig. 1.1. The different types of efficient machine learning strategies.

The goal of data-efficient methods is to develop machine-learning models that can learn from less amount of labeled data. Data efficiency can be achieved by preventing overfitting in different ways. For example, transfer learning uses abundant labeled data in an auxiliary source domain to transfer knowledge to a sparsely labeled target domain. This causes target domain predictions to be biased and prevents generalization and overfitting problems. On the other hand, generative learning approaches learn to produce more samples for data-starved classes using variations of different generative models like generative adversarial networks (GANs) [20], variational autoencoder (VAE) [21], etc. Hence, traditional supervised learning methods can be used for all the categories. Alternative data-efficient methods include: (a) self-supervised learning, which learns surrogate tasks like rotation/location prediction in unlabeled images to produce features for downstream tasks and (b) weakly-supervised learning, which consists of using incomplete, inaccurate or inexact supervision to train predictive models. The effect of these data-efficient methods include: (i) less training

time because of less number of data samples and subsequently faster optimization procedure; (ii) less memory footprint because there is no need to store large amount of training samples and their labels, and (iii) less energy is consumed because of more efficient training.

On the other hand, the goal of model-efficient methods is to develop machine learning models that can be represented using less number of parameters. It is mostly used for neural-network models with deep and complex architectures. Common model-efficient methods include pruning, which is a post-processing step of removing neural-network nodes and weights that do not contribute to the performance of the model. Low-rank methods approximate model weights using tensor factorization techniques that produce less number of parameters. Knowledge distillation trains a smaller network to reproduce the outputs of a larger network. Neural architecture search is a more generic approach that learns the neural-network architecture from the data. Hence, a minimal structure is learned without redundant-model parameters. The effect of these model-efficient methods include: (i) less inference time because of smaller models and hence efficient computation; (ii) less memory footprint because of less amount of storage required for smaller models, and (iii) less energy is consumed because of more efficient computation.

In this thesis, we are mainly concerned about data-efficient methods because it is a precursor to producing efficient models. This is because less amount of labeled training data implies that simpler models can be learned. On the other hand, efficient models do not necessarily imply that they can learn from less amount of labeled data.

Human learning is more data-efficient compared to machine learning. In the object-recognition domain, humans have this inherent ability to recognize new categories of objects or new varieties of the same objects from very few demonstrations. They do that by extracting useful knowledge obtained from observing a large set of categories throughout their lives. This knowledge is then transferred to enable recognizing new categories or different varieties of the same categories efficiently.

On the other hand, current approaches in deep learning require lots of labeled data for training a high-performance model. These approaches focus on the task of obtaining large annotated datasets like ImageNet [22]. These datasets are then used for training models consisting of complex deep neural-network architectures over a long time schedule [12,23]. These models are then evaluated on a carefully constructed test dataset on which they claim to produce state-of-the-art results. Thus, most of these deep-architecture-based models for supervised learning are closed form in nature because they are evaluated on the same set of categories that they are trained on. In other words, the training and testing distributions are assumed to be the same. This scenario is very restrictive in real-world situations when new categories of objects or new varieties of similar objects are produced everyday. Current deep-learning models when tested on data from a new data distribution would fail miserably. The *generalization* ability of deep neural networks is thus limited to only data from the training distribution.

If the new distribution contains a few labeled data and we train a model using only those data, the model will not generalize well to unseen data from the new distribution. This is because of the problem of *overfitting*, where a complex model with large number of parameters tries to learn from insufficient amount of data. If we use a simpler model we may encounter *underfitting*, where the model has not enough capacity to learn the intricate complexity of the sampled data. Thus, there is a need to construct models that can deal with generalization issues arising due to new data distributions.

As previously mentioned, humans can learn from few demonstrations very easily and they do that by using their existing knowledge and apply that to new situations. In machine learning, we can do something similar by using an existing model that is akin to the existing knowledge base of the human. The existing model can be pre-trained on abundant labeled data sampled from a source distribution. The model can then be adapted to sparsely labeled data sampled from a new target distribution. This model can then be expected to generalize well to unseen test data sampled from

the target distribution. This process of transferring and adapting a model learned from a source-data distribution to a target distribution containing few/zero labeled data is known as Transfer Learning (TL). Transfer learning is thus an attempt to produce human-like data-efficiency in machine learning. Hence, we pursue TL as the topic of this thesis to resemble human-like learning.

In this thesis, we consider a family of TL techniques for recognition tasks, where the goal is to classify a data sample into one of the predefined categories. Accordingly, TL can be broadly classified into two kinds of problems depending on the type of discrepancy in the source (training) and the target (testing) distributions. In the first kind, we have the same set of categories for training and testing except that the image varieties in the training and the testing cases are different. A toy example can be real images of dogs and cats in the training distribution and cartoon images of dogs and cats in the testing distribution. This example is shown in Fig. 1.2 (b). The goal would be to classify between a dog and a cat irrespective of whether it is a cartoon or a real image. In the second kind, we have a different set of categories in the training and the testing distributions. A toy example can be real images of dogs and cats in the training distribution and real images of giraffes and tigers in the testing distribution. The goal would be to learn a model that is able to classify between giraffes and tigers having very few/zero samples of giraffes and tigers but large number of samples from the dog and the cat categories. This example is shown in Fig. 1.2 (c).

The problem associated with the first kind of distribution discrepancy, where we have the same set of categories in the training and testing distributions, is commonly known as *domain adaptation* (DA). It is also known that the training distribution is also called the source domain and the testing distribution is also called the target domain. It is assumed that we have lots of labeled data available in each of the categories of the source domain. On the other hand, the target domain may be sparsely labeled or fully unlabeled. This is because it is difficult to obtain annotations for the new domain. The problem consisting of sparsely labeled data as well as unlabeled data in the target domain is known as *semi-supervised domain adaptation*

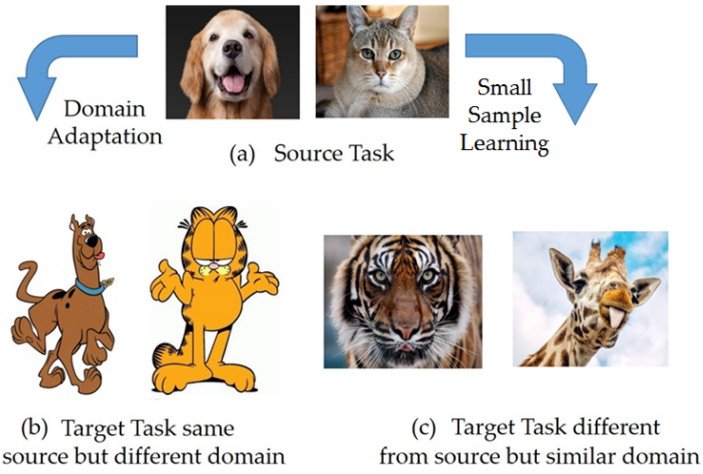


Fig. 1.2. The two kinds of transfer learning problems used in our research. The first kind is (b) domain adaptation, where the source and the target domains have the same set of categories and the second kind is (c) small sample learning, where the source and the target domains have different categories.

(SSDA). The specialized case of having only unlabeled target domain data is therefore known as *unsupervised domain adaptation* (UDA). Thus, the objective of DA is to exploit the labeled source-domain data and the unlabeled/sparsely labeled target-domain data to build a high-performance classifier for the target domain. In this thesis, we only consider the UDA case since UDA is a more challenging and a more realistic problem compared to SSDA, which can be trivially extended from UDA.

The problem associated with the second kind of discrepancy is known as *small-sample learning* (SSL). In this case, the source and the target domains have disjoint set of categories. SSL can be further classified into *few-shot learning* (FSL) or *zero-shot learning* (ZSL) depending on whether the target-domain categories are sparsely labeled or fully unlabeled, respectively. The absence of lots of labels in the target domain can be due to the presence of novel rare categories for which obtaining real-world samples is difficult. A restrictive extension to FSL exists where we only have access to the source domain models or prototypes and not the data. This problem is known as *hypothesis transfer learning* (HTL). Thus, the goal of SSL is to use

the source-domain information and sparsely labeled/unlabeled data in the target domain to build a high-performance classifier for the target domain. In this thesis, we describe approaches to each of the subproblems of SSL; that is, few-shot learning, zero-shot learning and hypothesis transfer learning. The overall categorization of transfer learning and its subproblems are shown in Fig. 1.3.

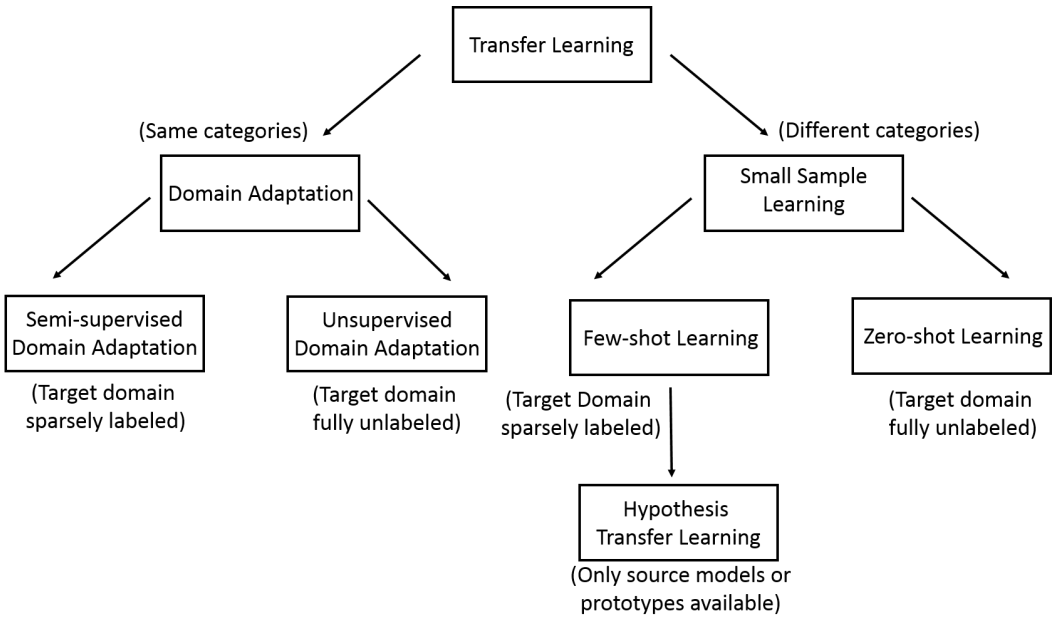


Fig. 1.3. Taxonomy of transfer learning problems.

In Fig. 1.4, we represent each of the above described TL sub-problems as a plot of the distribution discrepancy between the source and the target domains with respect to the overall information availability. Obviously, the SSL sub-problems will have more distribution discrepancy because of the presence of novel categories in the target domain. For the DA sub-problem, SSDA has more information availability due to the presence of a few labeled target domain data whereas UDA has fully unlabeled data in the target domain. For the same reason, FSL has more information availability compared to ZSL. However, it is tricky to compare information availability between HTL and ZSL problems. Since the HTL problem definition suggests no access to source data and only access to source models or prototypes, it is reasonable to put the

HTL sub-problem as having less information availability compared to ZSL. It is also important to note that the TL task becomes more difficult to perform as information from the domains becomes less available and the distribution discrepancy between the domains becomes larger. In the next section, we describe previous works on each of these TL sub-problems.

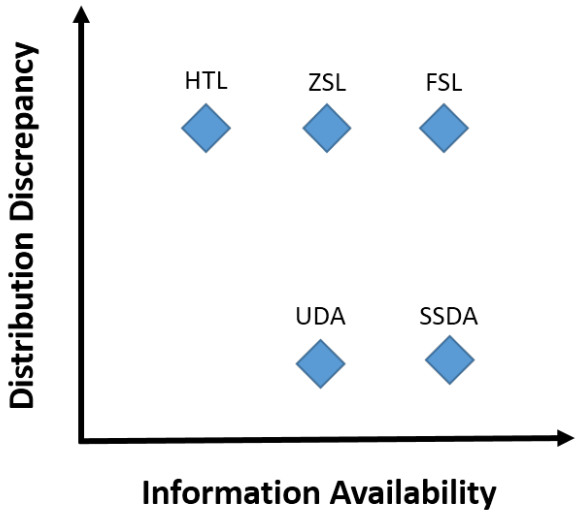


Fig. 1.4. Plot of the different TL sub-problems as a function of the distribution discrepancy and information availability. The plot also depicts the difficulty of each task. Tasks closer to the top left corner are more difficult.

1.2 Literature Survey

There is a large body of prior work on transfer learning. The first survey paper [24] introduced the definition of transfer learning and the various sub-problems along with different application areas. However, the survey paper is outdated in the terms of the available datasets and benchmarks. A more recent survey paper on transfer learning is the work by Weiss et al. [25]. It includes references to more recent papers and results. There are also survey papers specifically on each of the sub-problems of TL. As previously discussed, TL is broadly classified into the problems - Domain Adaptation

(DA) and Small Sample Learning (SSL). One can refer to [26] and [27] for a more comprehensive discussion about previous methods on DA and SSL, respectively.

In the following section, we discuss in detail, prior work on each of the sub-problems of Transfer Learning.

1.2.1 Domain Adaptation

Most of the previous DA methods can be broadly classified into two categories, depending on whether a representation is learned or not. In this work, we refer to the methods with/without representation learning as deep/non-deep learning methods interchangeably. The non-deep-learning DA methods can be divided into three categories – *parameter adaptation* methods, *instance re-weighting* methods and *feature transfer* methods. Parameter adaptation methods [28–31] were one of the earliest methods that generally adapted a trained classifier in the source domain (e.g., an SVM, Logistic Regression) in order to perform better in the target domain. Most of these methods assumed the presence of at least a small set of labeled examples in the target domain. As a result, they cannot be applied to the more challenging unsupervised setting (UDA).

Instance Re-weighting methods are the second group of methods used for DA. These methods assumed that conditional distributions were shared between the two domains. As a result, one could just use the marginal distribution of the two domains to find the ratio of the joint distributions. Using this assumption, the instance re-weighting methods involved estimating the ratio between the likelihoods of the two domains to compute the instance weight. This could be done by estimating the likelihoods in an independent manner as in [32] or by using the density ratios [33, 34]. One of the most popular measures used to weigh data instances, used in [35, 36], was the maximum mean discrepancy (MMD) [37], calculated between the distributions in the two domains. This metric is popularly used in feature transfer methods as well.

Feature transfer methods, on the other hand, do not assume similar conditional distributions in the two domains. These methods generally try to find a transformation in the feature space which facilitates domain adaptation. An early but innovative method for DA was proposed in [38], where the representation is modified such that the source features are $(\mathbf{x}_s, \mathbf{x}_s, 0)$ and the target features are $(\mathbf{x}_t, 0, \mathbf{x}_t)$, where $\mathbf{x}_s$ and $\mathbf{x}_t$ are source and target domain samples respectively. This method enabled identifying shared and domain-specific features simultaneously in each of the domains. Popular feature transformation methods include subspace-based approaches. Among these methods are the Geodesic Flow Sampling (GFS) [39, 40] and the Geodesic Flow Kernel (GFK) [41, 42] approach, where the domains are considered as samples lying on a trajectory on the Grassmannian manifold. To carry out domain adaptation, a domain is sampled from the manifold path so that the two domains can be mapped onto it. Similarly, the Subspace Alignment (SA) method [43] aligned the two domains by optimizing the Bregman divergence metric. The linear Correlation Alignment (CORAL) algorithm [44] is a statistical approach that aligned the covariance between the two domains. The Joint Geometric and Statistical Alignment (JGSA) method [45] combined both subspace and statistical methods to carry out DA. As a result, it produced better results than both statistical and subspace-based methods. Transfer Component Analysis (TCA) [46] discovered shared hidden features having similar distribution between the two domains. Chen at al. [47] proposed a reconstruction-based approach to learn a domain invariant representation. Most of these previous feature transformation methods learned a global alignment between the two domains while [48] and [49] considered local approaches to match samples between the two domains.

Deep-learning-based DA approaches learn a domain invariant representation. This kind of representation allows the samples from the source and target domains to work with a single classifier. The representation learning framework is realized using a neural network architecture. Most deep-learning methods for DA use a Siamese architecture with two streams for the two domains. The representation is learned by

optimizing some loss function. In addition to the classification loss, these methods use a discrepancy-based loss [50–54] or an adversarial-based loss [55–57] to produce the domain-invariant representation. The classification loss uses the labels available from the source domain data. The discrepancy loss aligns the two domains by minimizing some domain discrepancy metric. On the other hand, adversarial-based methods play a minimax game of generating domain-invariant representations against a domain critic. Among the discrepancy-based methods, Long et al. [50, 51] used MMD to minimize the discrepancy between the source and target representations. On the other hand, Sun et al. [54] used covariance as the discrepancy metric between the source and target representations. Yan et al. [58] used the principle of independence maximization to produce domain invariant features. In addition to the criterion of domain invariant representations, the authors of [59–61] constrained the features to be class-discriminative by ensuring small intra-class separation but large inter-class separation. Among the adversarial methods, Tzeng et al. [55] proposed a generic framework for domain-adversarial methods where the loss functions and architectures are user-defined. The Domain-Adversarial Neural Networks (DANN) [56] used a gradient reversal layer to produce features that are discriminative as well as domain-invariant. Shen et al. [57] also used an adversarial method where it minimizes the Wasserstein distance between the representations of the two domains. Lee et al. [62] extended the approach by using a sliced Wasserstein distance, instead. The authors of [63] also proposed a domain agnostic learning procedure that disentangles domain specific features from domain-agnostic ones. More recent works on adversarial domain adaptation explore contemporary ideas like cycle-consistency [64], conditional information [65], spectral penalization [66] for discriminability and transferability and domain-symmetric networks [67]. The main disadvantage of these adversarial methods is that their training is generally not stable and therefore not convergent. Moreover, empirically tuning the capacity of a discriminator requires lot of manual effort.

Between these two classes of DA methods, the state-of-the-art methods are dominated by deep learning methods. However, these approaches are quite complex and expensive, requiring re-training of the network and tuning of many hyper parameters such as the structure of the hidden adaptation layers. Non-deep-learning domain-adaptation methods do not achieve as good performance as a deep-representation approach, but they work directly with shallow/deep features and require lesser number of hyper-parameters to tune. Among the non-deep-learning based domain-adaptation methods, feature transformation methods are more generic because they directly use the feature space from the source and target domains, without any underlying assumption of the classification model. In fact, a powerful shallow-feature transformation method can be extended to deep-architecture methods, if desired, by using the features of each and every layer and then jointly optimizing the parameters of the deep architectures as well as that of the classification model. For example, correlation alignment [44] has been extended for deep architectures [54], which evidently achieve the state-of-the-art performance. Moreover, a local transformation-based approach as in [48, 49] will result in better performance than global transformation methods because it considers the effect of each and every sample in the dataset explicitly.

1.2.2 Few-shot Learning

For the purpose of few-shot learning (FSL), we generally have base categories from the source domain consisting of abundant labeled data. The target domain consists of novel categories with very few labeled data per category. It is known that direct training using the sparsely labeled target domain data will cause the model to overfit. Therefore, we need to utilize the source domain data to learn some useful transferable knowledge that enables recognition of the novel categories from the sparsely labeled data. Accordingly, previous methods can be categorized depending on the type of transferable knowledge extracted from the source domain.

The earlier methods used priors to facilitate FSL. Li et al. [68] used a global prior while Salakhutdinov et al. [69] used a super-category-level prior. For application-specific tasks like handwriting recognition, generative models have been proposed that can produce characters from parts [70] or strokes [71]. For object recognition, a hierarchical Bayesian program has been proposed to utilize compositional and causal approaches to create a probabilistic generative model for visual objects [72,73]. Some ad-hoc approaches to address few-shot learning were to carry out data augmentation by harnessing unlabeled data [74], by transformation and adding noise [75,76], and by synthesizing artificial examples [20,77–79] or using compositional representations [80, 81]. More recent methods that used generative modeling include the auto-encoder [82] and variations of adversarial-network-based architectures [83, 84]. However, most of these generative methods require lots of efforts to generate data, otherwise the generated data do not represent the actual data distribution. Thus, recent methods mostly take a metric-learning or a meta-learning approach to few-shot learning.

Metric learning approaches strive to preserve class neighborhood structure; that is, the representations are learned such that features from the same class are clustered together while features from different classes are kept far apart. As a result, novel class features are expected to have more room for classification error. Koch et al. [85] used Siamese Networks for metric-learning to match training example of a novel category to a test example. The training was carried out using an object recognition dataset. Vinyals et al. [86] proposed Matching Networks, which use a nearest-neighbor classifier in addition to an attention mechanism over the training samples. This paper was the first to introduce the episodic learning strategy for FSL. The episodic learning strategy tries to simulate the test condition required for classifying novel categories. Accordingly, training batches are sampled such that the number of classes and the number of samples per class are the same as in the novel test categories. Prototypical Networks [87] extended nearest mean classifiers [88] and learned to classify test samples by calculating Euclidean distances to prototype features. As an extension to Prototypical Networks, Sung et al. [89] learned a distance

metric instead of using a predefined distance metric. A more recent method [90] used a metric learning approach, where the metric is scaled and adapted based on the task.

Meta-learning methods for few-shot learning use a learning-to-learn scheme, where a model extracts useful transferable knowledge about the learning procedure from a large collection of tasks. This helps in quickly learning the novel task, which in our case, is recognition of novel categories. Ravi and Larochelle [91] used LSTM [10] to train a meta-learner to produce model parameter updates for optimization of a base learner on a task. This method basically learns the optimization procedure using data from a number of auxiliary tasks. The work on learning-to-learn [92] approach to few-shot learning is also closely related to learning to optimize. Finn et al. [93] built upon this work to only learn the initial parameter for gradient descent so that it optimizes the learner for a new task in a few iterations. Mishra et al. [94] introduced temporal convolutions over samples to predict the label of a test example, given a sequence of labeled samples and the unlabeled test sample. The transductive propagation network [95] classifies the whole test dataset using a graph-based label propagation mechanism. They use an end-to-end meta-learning framework to learn the feature embedding and graph construction simultaneously. Sun et al. [96] used a meta-transfer learning mechanism that shifts and scales neural network weights for new tasks. Similarly, Munkhdalai et al. [97] proposed a meta-learning scheme that shifts the neuron activations depending on task-specific parameters.

Other relevant few-shot learning methods that deserve mentioning include memory-based models [98,99] that store relevant information in a memory module and use that for comparison at test time. Attentive comparators [100] compare patches of images sequentially through an attention mechanism and then arrive at a prediction. Wang et al. [101] proposed an unconventional method to FSL by learning a transformation from small-sample-model parameters (parameters learned using less number of samples) to large-sample-model parameters (parameters learned using large number of samples). The work on few-shot learning without forgetting [102] also used a transformation but with a different distance metric and without any procedure to avoid

negative transfer. However, their transformation depends on the location of nearby source domain categories with respect to target domain novel categories. Qiao et al. [103] learned a category-agnostic mapping from activations to parameters that allowed fast generalization to novel categories. A similar idea [104] was used to imprint weights for the classification layer of the novel categories. Bertinetto et al. [105] used a differential closed-form solver based on ridge regression for fast adaptation to novel categories. Some methods extended existing machine learning concepts like graph neural networks [106] and information retrieval [107] to few-shot learning. For a more comprehensive survey on few-shot learning, one can refer to [27, 108].

1.2.3 Hypothesis-Transfer Learning

Hypothesis-Transfer Learning is a very restrictive but realistic setting with access to only source models/hypotheses and no access to source data. As a result, there has been only few works addressing this setting. Early work on hypothesis transfer learning was built around the support-vector-machine (SVM) framework and applied to prosthetics [109]. In that work, the authors proposed that the target model should be close to a pre-trained source model. The concept of adaptive SVMs has also been used in video concept detection [31]. An adaptive SVM was also developed to handle multiple source models [110]. The goal was to force the target model to be close to a linear combination of source models. The weight of each source model signified the semantic similarity between that source category and the target category. An incremental approach based on a similar concept was introduced to deal with online learning of novel categories [111]. Jie et al. [112] presented a multi-class transfer learning algorithm, which takes the advantage of priors as experts and transfers their outputs to the target samples as additional information. The problem is cast as an optimization problem within the multi-kernel-learning framework. Kuzborskij et. al. [113] studied the algorithmic stability of HTL based on biased-regularization of least squares. Kuzborskij et. al. [114] also established generalization and excess

risk bounds to show that generalization happens at a fast rate if the target task is fed with a good source hypotheses set. On the other hand, if the source hypotheses combination is bad, the usual learning rate is obtained that would have been obtained had we trained the model from scratch using the few labeled target samples. Recently, a greedy algorithm [115] for transfer learning has been developed, which selects relevant source hypotheses inspired from algorithms for feature subset selection. An SVM-based method [116] was developed that uses support vectors from the source hypothesis as privileged information for learning the target model. Furthermore, Wang et. al. [117] proposed an unorthodox method of training source models by training them in an unsupervised fashion, freeing classification to a particular set of categories. This kind of unsupervised training would allow models to be generalized to novel categories.

1.2.4 Zero-shot Learning

In Zero-shot learning (ZSL), we have disjoint categories in the source and target domain with no labeled data in the target domain. To be able to recognize unseen categories in the target domain, we train a learning model using a large collection of labeled samples from the seen categories in the source domain and then adapt it to unseen categories. For zero-shot recognition, the seen and unseen categories are related through a high-dimensional vector space known as semantic description space. Each category is assigned a unique semantic description. Examples of semantic description can be manually defined attributes [118] or automatically extracted word vectors [119]. Different ZSL methods are developed depending on how the feature and the semantic description space are related.

Most ZSL methods involve mapping from the visual feature space to the semantic description space or vice versa [120–123]. Sometimes, both the visual features and the semantic descriptors are mapped to a common feature space [124,125]. Most of these mapping-based approaches learn an embedding function for samples and semantic

descriptors. The embedding is learned by minimizing a similarity function between the embedded samples and the corresponding embedded semantic descriptors. Thus, most ZSL methods differ in the choice of the embedding and similarity functions. Lampert et. al [118] used linear classifiers, identity function and Euclidean distance for the sample embedding, semantic embedding and similarity metric, respectively. Romera-Paredes et al. [126] used linear projection, identity function and dot product. ALE [121], DEVISE [122], SJE [127] all used a bilinear compatibility framework where the projection was linear and the similarity metric was a dot product. They used different variations of pairwise ranking objective to train the model. LATEM [128] was an extension of the previous method [127], which used piecewise linear projections to account for the non-linearity. CMT [123] used a neural network to map image features to semantic descriptors with an additional novelty detection stage to detect unseen categories. SAE [129] used an auto-encoder-based approach, where the image feature is linearly mapped to a semantic descriptor as well as being reconstructed from the semantic descriptor space. DEM [120] used a neural network to map from a semantic descriptor space to an image feature space.

At the same time, a new evaluation setting known as Generalized Zero-Shot Learning (GZSL) and was introduced by Chao et al. [130]. The authors found that the performance of unseen categories in the GZSL setting was poor and proposed a shifted calibration mechanism to improve the performance. This shifted calibration mechanism was meant to lower the classification scores of the seen categories. However, the embedding methods perform poorly in the GZSL setting. This is because only the seen categories are used in learning the embedding as a result of which there is projection domain-shift and seen class biasness. As a result, *transductive* approaches have been proposed that assume access to the unlabeled testing data during the training stage. For example, Fu et al. [131] fused semantic information from multiple sources and carried out label propagation on the unlabeled test data. Kodirov et al. [132] reformulated ZSL as a dictionary learning problem where the dictionary bases of the seen and unseen classes were enforced to be similar. Rostami et al. [133] also used a

coupled dictionary for the seen and unseen classes in a common representation space for the visual feature and the semantic descriptor. The ZSL problem was also converted into a semi-supervised learning problem [134,135] where clustering was carried out on the unlabeled unseen data. However, this transductive approach is unrealistic as it assumes access to the unlabeled test data from unseen categories during the training stage. At the very least, we could carry out the test-time post-processing of the semantic descriptors.

Hybrid models expressed image features or semantic embeddings as a combination/mixture of existing seen features or semantic embeddings. For example, SSE [136] used relationship between classes to embed both the semantic and visual spaces to a common space. CONSE [137] utilized the probability of a novel class sample belonging to a base class sample to make a classification decision. SYNC [125] tried to learn a mapping between the semantic space and the parameter space. This mapping is then used to synthesize model parameters for the novel classes. However, these hybrid methods do not receive much attention currently. This is because the strict assumption that unseen classes can be represented as a combination of seen classes does not hold universally for all datasets.

Generative ZSL methods were recently proposed to tackle the GZSL problem. They convert ZSL into a supervised-learning problem by generating data for the novel categories. One of the earlier methods [138] carried out a comparative study of different generative models for GZSL. Xian et al. [139] used WGAN [140] to generate features for the novel classes. They also used an additional loss function to generate more class-discriminative features. Zhu et al. [141] also used a GAN-based framework to generate images of noisy text, where the generated images were constrained with a visual centroid-based regularization. Similarly, Li et al. [142] introduced class-specific soul samples such that the generated features are constrained to be close to these soul samples. The variational auto-encoder framework was also used to synthesize visual features for the unseen categories [143, 144]. More recently, an adversarial network was proposed that carries out metric learning in addition to visual to semantic and

semantic to visual mappings [145]. Schonfield et al. [146] used a two-VAE framework, where a latent space was introduced to reduce cross-modal discrepancies for both visual features and semantic descriptors. The authors of [147] used a multi-modal consistency network for reconstruction purposes. A detailed discussion on each of these methods can be found in a comprehensive survey on zero-shot learning [148]. A summary of all the related work and their categorization is shown in Table 1.1. In the next section, we discuss our contributions and how we solve each of the TL sub-problems.

Table 1.1. Categorization of previous works for different transfer learning sub-problems.

Problem	**Category**	**References**
Unsupervised Domain Adaptation	Instance Re-weighting	[32–36]
	Feature Transfer	[38–46, 48, 49]
	Discrepancy-based	[50–54, 58–61]
	Adversarial-based	[55–57, 62–67]
Few-shot Learning	Metric-Learning	[85–87, 89, 90]
	Meta-Learning	[91–97]
	Generative	[77–84]
	Alternative	[68, 69, 71, 98–107]
Hypothesis Transfer Learning	Combination-based	[110–112, 114, 115]
Zero-shot Learning	Embedding-based	[118, 120–123, 126–129]
	Transductive	[131–135]
	Generative	[138, 139, 141–147]
	Hybrid	[125, 136, 137]

1.3 Contributions and Organizational Overview

In this section, we describe our contributions to the different transfer learning sub-problems. Each of these transfer learning problems has different settings for domain discrepancy and information availability. Hence, it is difficult to propose a common framework for each of these problems. However, each of these transfer learning problems requires prior knowledge extracted from the source domain containing abundantly labeled data. This prior is used to facilitate learning in the target domain, which would otherwise overfit on the sparsely labeled data. In this thesis, we propose the use of a prior that can be expressed through different structures. These structures encode relationships between different entities obtained from the data. This structural encoding extracts relational knowledge from the source domain, which is more informative compared to previous approaches. Therefore, we expect that these proposed group of methods will perform much better than previous approaches. For each transfer learning sub-problem, different combinations of structures and entities are summarized in Table 1.2. A brief summary of how the structural priors are used for the four transfer learning tasks is given below -

Table 1.2. Different structural priors that we proposed for the different transfer learning sub-problems.

Problem	**Structure**	**Entity**
Unsupervised Domain Adaptation (UDA)	Graphs and Hypergraphs	Sample-Sample
Few-shot Learning (FSL)	Neural Network	Sample-Class Prototype
Hypothesis Transfer Learning (HTL)	Manifold	Class Prototype-Class Prototype
Zero-shot Learning (ZSL)	Neural Network	Sample-Semantics

- **Unsupervised Domain Adaptation:** We assume that the structural relationship between the data distribution is preserved across the domains. Hence,

graphs/hyper-graphs are used as structural priors, which are constructed from the samples in the source domain. The process is repeated in the target domain. Finally, the source and target graphs/hyper-graphs are matched and transformed to minimize the domain discrepancy.

- **Few-shot Learning:** We use neural network as the structural prior to predict the mean (prototype) and variance of the data-starved novel categories. This neural network module is learned from data-abundant source categories. For the novel category, few-shot samples are set as input to produce the class prototype from the neural network. The prototype is again used to predict the variance of that new category.

- **Hypothesis Transfer Learning:** We cannot use neural network as a structural prior because it will overfit to the source-class prototypes. Hence, we use a non-parametric prior in the form of a manifold. The source-class prototypes are used to fit the manifold on which the few-shot samples are projected to obtain the novel class prototype.

- **Zero-shot Learning:** We use neural network as a structural prior that learns the relationship between the semantic space and the sample space. The neural network can be used as an embedding to map from one space to another. Also, the neural network can be used as a generative model to synthesize samples for a data-starved novel category given the corresponding semantic descriptor as input.

Therefore, different structural priors are used to model relations between data entities at different levels - sample (low-level), prototype (mid-level) and semantics (high-level). Also, each method has additional learning stages, constraints or unique optimization procedures that further improve the performance. Now, we will discuss in details, the proposed solution for each of the four tasks.

For the UDA problem, our goal is to minimize distribution discrepancy between the source and the target domains. To minimize the domain discrepancy, we assume

that the structural relationship between the source samples is carried over to the target domain as well. To model the structural arrangement in the source domain, we use graphs or hyper-graphs as structural priors to encode the relationship between samples in the source domain. Since the structural relationship carries over to the target domain, we use a graph/hyper-graph matching formulation to transform the source domain to be closer to the target domain. This localized framework produces better domain matching than previous global methods that use statistical measures to minimize domain discrepancy. Using this structural-matching-based concept, we proposed three UDA methods.

The first method [149] uses first-order and second-order sample relations to match the source-domain data and the target-domain data. Once the relations are established, the source domain is mapped to be close to the target domain. A class-based regularization is also used to leverage the labels present in the source domain so that the matching is smooth. This transformation approach is computationally inefficient and therefore the optimization problem is decomposed into solving a series of sub-problems for which an efficient solution using a network-simplex approach exists. The second method [150] extends the first method by using an additional third-order relation followed by hyper-graph matching. A more computationally efficient method of obtaining the solution is proposed that involves solving a series of sub-problems using Alternating Direction Multiplier Method (ADMM) [151]. Moreover, an initial clustering is carried out to select the most relevant instances which reduce the number of variables to be used in the optimization approach. For the third method [152], a domain-invariant representation is learned by minimizing a novel graph matching loss function between the source and the target domain representations. Once the discrepancy is minimized, an additional refinement of the model is performed using the unlabeled target domain data. The pseudo-labels of the confident unlabeled target domain data are used to make sure that the target samples lie further from the softmax decision boundary. A brief comparison of these proposed methods is given in Table 1.3.

Table 1.3. Comparison of the proposed structural-matching-based approaches for UDA.

Method	**Matching Order**	**Representation Learning**	**Optimization Method**
I [149]	First, Second	No	Conditional Gradient + Network Simplex
II [150]	First, Second, Third	No	Conditional Gradient + ADMM
III [152]	First, Second	Yes	Stochastic Gradient Descent

For the FSL problem, we have abundantly labeled source-domain data and sparsely labeled target-domain data. Also, the categories in the source domain and the target domain are different from each other. Since the target domain has few labeled data, it is not possible to estimate the mean (prototype) and the variance of the target-domain classes. Hence, training a model on the target-domain data will cause overfitting and eventually misclassification. Thus, we seek to find a transformation from the few-shot samples to their class mean and variance [153]. This transformation is realized using a neural-network-based prior which is learned from the source-domain data. For the target-domain data, this trained neural network can transform the few-shot samples to the class mean and variance and consequently produce better classification performance. The classification scheme involves computing Mahalanobis distances to class prototypes and obtaining output class probabilities .

In addition to the proposed classification scheme, we propose a new representation which is constructed using pairwise distances between samples. This new representation is low-dimensional but discriminative and hence prevents overfitting due to the curse of dimensionality. In a way, our proposed method combines both metric-learning (learning a representation) and meta-learning (learning to classify)

approaches to FSL. Thus, it takes the advantages of both metric- and meta-learning group of methods to produce much better results than previous approaches.

The HTL problem extends FSL by further assuming that we do not have access to the source-domain data. We only have access to high-level information in the form of source-class prototypes. Previous methods used source-domain models as high-level information. However, we argue that using source-domain models do not allow for fair comparison of different methods. One can use a more powerful source-domain model to improve their performance without contributing much to their transfer learning mechanism. On the other hand, using source prototypes allow fair comparison of different methods where the results depend only on the training and the testing data. Still, we need to extract prior knowledge from the source prototypes and apply that to the target domain.

For the HTL problem, it is not possible to use a neural-network-based parametric prior because of the simple reason that it might overfit on the small number of source prototypes. Alternatively, we could use non-parametric methods that perform well in low sample situations. Inspired by a previous study [154], we assume that the source-domain-class prototypes lie on a non-linear manifold. Thus, a manifold structure that encodes relationship between class prototypes is used as the prior extracted from the source domain. We extend this assumption to the target domain, where the class prototypes are found by projecting the few shot samples onto the manifold. For classification purposes, we could use a nearest-neighbor-based classification approach. However, the target-domain prototypes might be erroneously estimated and so there is a need to use the global-class arrangement to carry out classification. This structural arrangement of the class prototypes are used to induce an absorbing Markov chain. The equilibrium probability of this absorbing Markov chain process is then used for assigning the class of a test sample. We also proposed a parametric Bayesian baseline to compare our proposed non-parametric manifold-based approach. Unlike neural networks, the Bayesian baseline has less number of parameters and hence does not

suffer from overfitting. The results of our proposed approaches can be used as a benchmark for future research in HTL.

The ZSL problem is a special case of the FSL problem where we do not even have access to the labels of the target-domain data. However, we have side information in the form of semantic descriptors. The goal is to learn a mapping that relates the sample space and the semantic space, using labeled data from the source domain. The mapping is realized using a neural-network-based structural prior and can be used both for embedding and generating data. However, the major problem with current embedding and generative approaches is that they cannot use target-domain data. As a result, most of these models are biased towards the source-domain classes and therefore cannot generalize well to target-domain classes. We propose to address this problem through use of constraints on the neural network as well as through post-processing steps [155].

We use two kinds of constraints on the neural network. The first kind is an algorithmic constraint, where we have additional loss functions that use one-to-one and pairwise matching to relate the semantic space and the feature space. This structural matching constraint allows the mapping to be generalized easily to unseen data and also prevents the hubness problem associated with high-dimensional nearest neighbor classification. The second kind of constraint is the architectural constraint, which is realized by using additional neural network modules on top of generative neural networks. The proposed constraints are realized firstly, using a discriminative module that classifies between seen and unseen categories and then using a reconstruction module that produces semantics from the samples. We also explore two additional post-processing steps. The first one is a calibration mechanism, where the classification scores of the source classes are scaled so that the results are not biased towards the source-domain categories. However, the calibration method does not use the unlabeled ground-truth test data and therefore we propose to use domain adaptation as an additional post-processing step to adapt our model to the unlabeled testing data.

Table 1.4 summarizes our contributions to all the small-sample learning problems - FSL, HTL and ZSL.

Table 1.4. Brief summary of the contributions to the small-sample learning problems. The right column describes the contributions and their type.

Problem	**Major Contribution**
FSL [153]	Representation: Relative Features Classification: Predictive Statistics
HTL	Estimation: Manifold Classification: Absorbing Markov Chain
ZSL [155]	Constraints: Structural Matching, Discrimination and Reconstruction Post-Processing: Scaled Calibration and Domain Adaptation

Our proposed algorithms and architectures can also be applied to other problems in machine learning beyond transfer learning. For example, the graph-matching metric that we proposed for UDA can be extended to generative models to measure discrepancy between the real and the synthetic data. Similarly, the concept of predicting mean and variance for FSL can be used for generating data from the novel categories. However, we need to model variances for all the features to capture a more accurate distribution. For HTL, we propose a Markov-chain-based distance function for classification. This distance function can also be used for training more discriminative features that take into consideration the global arrangement of the classes. Finally, we propose a ZSL framework that models bi-directional relationship between image features and semantic descriptors. This relation can be used in media retrieval; that is, to produce images given a description. It can also be used for image description; that is to generate a description given an image. However, we need a processing step in order to convert the description from the semantics and vice versa.

In this thesis, we have tried to unify the different approaches used for solving the different transfer learning sub-problems. All of the proposed methods use a structural prior in the form of either graphs, manifolds or neural networks that encode

relationship between samples, prototypes or semantics. These methods can be organized in the thesis as follows. Chapter 2 describes all the proposed graph and hyper-graph matching approaches to unsupervised domain adaptation. In Chapter 3, we discuss our meta-learning- and metric-learning-based two-stage approaches to few-shot learning. Chapter 4 discusses two approaches to hypothesis transfer learning – the non-parametric manifold-based approach and the parametric Bayesian approach. In Chapter 5, we discuss the proposed non-generative and generative approaches to zero-shot learning. In all these Chapters, we evaluate and analyze our methods on standard image recognition datasets. Finally, in Chapter 6, we summarize our results and discuss possible future extensions.

1.4 Publications

Portions of this research have been submitted to and/or published in the following professional journal publications and/or professional conference proceedings:
Journal Publications:

- Debasmit Das, and C. S. George Lee. "Sample-to-sample correspondence for unsupervised domain adaptation." *Engineering Applications of Artificial Intelligence (EAAI)*, (73) (2018): 80-91.

- Debasmit Das, and C. S. George Lee. "A Two-Stage Approach to Few-Shot Learning for Image Recognition." *IEEE Transactions on Image Processing (TIP)*, vol. 29, no. 12, pp. 3336-3350, Dec. 2020.

- Debasmit Das, and C. S. George Lee. "A Constrained Generative Approach to Zero-shot Object Recognition." Under review in the *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*.

Conference Publications:

- Debasmit Das, and C. S. George Lee. "Graph Matching and Pseudo-Label Guided Deep Unsupervised Domain Adaptation." *Proceedings of the Interna-*

tional Conference on Artificial Neural Networks (ICANN), Rhodes, Greece, pp. 342-352, October 4-7, 2018.

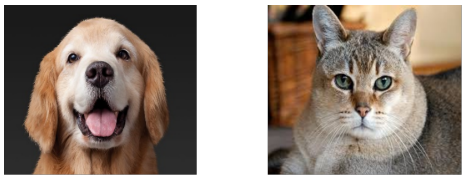
- Debasmit Das, and C. S. George Lee. "Unsupervised Domain Adaptation Using Regularized Hyper-Graph Matching," *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Athens, Greece, pp. 3758-3762, October 7-10, 2018.

- Debasmit Das, and C. S. George Lee. "Zero-shot Image Recognition Using Relational Matching, Adaptation and Calibration," *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, pp. 1-8, July 14-19, 2019.

2. UNSUPERVISED DOMAIN ADAPTATION USING GRAPH AND HYPER-GRAPH MATCHING

2.1 Introduction

In Chapter two, we tackle unsupervised domain adaptation, which is a sub-problem of transfer learning that assumes the same set of categories in both the source and target domains. The source domain is fully labeled while the target domain is fully unlabeled. However, the source and target distributions have some discrepancy as a result of which a classifier learned using source domain data would not perform well in the target domain. An example of domain discrepancy is shown in Fig. 2.1, where the source domain has frontal views of dogs and cats while the target domain has side views. Our goal is to minimize this domain discrepancy by finding a transformation that maps the source domain samples to be close to the target domain samples.



(a) Source Domain.

(b) Target Domain.

Fig. 2.1. Discrepancy between (a) the source domain and (b) the target domain. In the source domain, the images have frontal faces while the target domain has images of the whole body from the side view-point.

Most previous methods minimize the domain discrepancy using global statistical metrics. For example, Sun et al. [44, 54] tried to minimize the covariance difference between the source and the target domain. Long et al. [50, 51] used maximum mean discrepancy to tackle domain shift between the source and the target distributions. However, as shown in the toy example in Fig. 2.2(a), where the orange and red dots indicate source and target domain samples, respectively, the source and target samples do not match appropriately when global methods are used. To address this problem, Courty et al. [49] introduced a local approach using optimal transport, where each source domain sample is matched with each target domain sample. Although, this method produces better domain adaptation, it can produce ambiguous matching as shown in Fig. 2.2(b), especially when there is large domain discrepancy. This erroneous matching is mainly because only one-to-one distances are used for matching a source sample and a target sample. Therefore, we propose to exploit the intra-domain structural arrangement using graphs/hyper-graphs to aid domain adaptation. As shown for the example in Fig. 2.2(c), graph matching between the source and the target graphs produces better sample-to-sample matching than both local and global methods. Hence, the structural prior used in this setting is a graph/hyper-graph. The graph/hyper-graph encodes the relationship between data samples and therefore the structural information of the dataset. In our approaches, we assume that the structural information is preserved across the domains. Therefore, we pursue a graph/hyper-graph matching technique to minimize the domain discrepancy and eventually transform the source domain samples to be close to the target domain samples.

To be more specific, we describe three relevant methods based on this localized graph/hyper-graph matching-based approach. These methods place a strong emphasis on establishing a sample-to-sample correspondence between each source-domain sample and each target-domain sample. Establishing correspondences between two sets of visual features have long been used in computer vision mostly for image registration [156,157]. To our knowledge, the approach of finding correspondences between

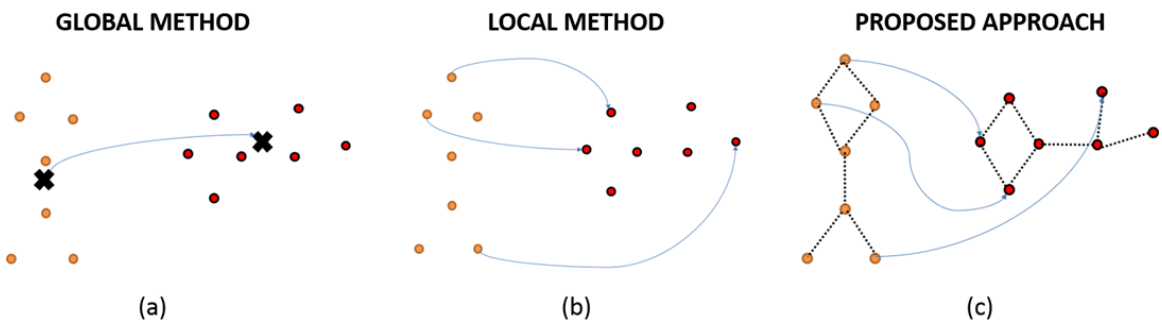


Fig. 2.2. Comparison of our proposed approach with previous work. In this diagram, the source and target domain samples are shown in orange and red, respectively.

the source-domain and the target-domain samples has never been used for domain adaptation. The only work that is similar to finding correspondences is the work on optimal transport [49]. They learned a transport plan for each source-domain sample so that they are close to the target-domain samples. Their transport plan is defined on a point-wise unary cost between each source sample and each target sample. Our approach develops a framework to find correspondences between the source and target domains that exploit higher-order relations beyond these unary relations between the source and target domains. We treat the source-domain data and the target-domain data as the source and target hyper-graphs, respectively, and our correspondence problem can be cast as a hyper-graph matching problem. The hyper-graph matching problem has been previously used in computer vision [158] through a tensor-based formulation but has not been applied to domain adaptation. Hyper-graph matching involves using higher-order relations between samples such as unary, pairwise, tertiary or more. Pairwise matching involves matching source-domain sample pairs with target-domain sample pairs. Tertiary matching involves matching source-domain sample triplets with target-domain sample triplets and so on. Thus, hyper-graph methods provide additional higher-order geometric and structural information about the data that is missing with just using unary point-wise relations between a source sample and a target sample. The advantage of using higher-order

information in graph matching is demonstrated in the example in Fig. 2.3. In Fig. 2.3, the graph on the left is constructed from the source domain while the graph on the right is constructed from the target domain. In the graph, each node represents a sample and edges represent connectivity among the samples. Samples 1 and $1'$ do not match because those samples are not the closest pair of samples. But as a group $\{1, 2, 3\}$ matches with $\{1', 2', 3'\}$ suggesting that higher-order matching can aid domain adaptation, whereas one-to-one matching between samples might not provide enough or provide incorrect information.

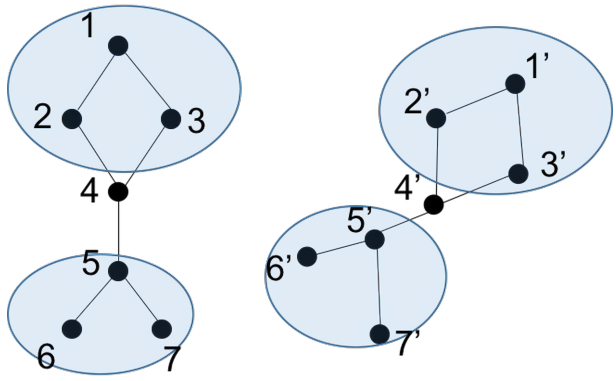


Fig. 2.3. Example showing the advantage of higher-order graph matching compared to just first-order matching.

Accordingly, in this Chapter, we describe three methods for unsupervised domain adaptation and the corresponding experimental studies. The first two methods directly work on engineered features without learning a representation while the third one utilizes representation learning. The first method [149] involves finding correspondences between samples of the labeled source domain and the unlabeled target domain. The correspondences are obtained by treating the source and target samples as graphs and using a convex criterion to match them. The criteria used are first-order and second-order similarities between the graphs as well as a class-based regularization. We have also developed a computationally efficient routine for the convex optimization, thus allowing the proposed method to be used widely. To ver-

ify the effectiveness of the proposed method, computer simulations were conducted on synthetic and image classification datasets. Results validated that the proposed local sample-to-sample matching method out-performs traditional moment-matching methods and is competitive with respect to current local domain-adaptation methods. In the second method [150], the matching between the samples in the two domains are found by treating the source and target domains as hyper-graphs and carrying out a class-regularized hyper-graph matching using first-, second- and third-order similarities between the graphs. We have also developed a computationally efficient algorithm by initially selecting a subset of the samples to construct a graph and then developing a customized optimization routine for graph-matching based on Conditional Gradient and Alternating Direction Multiplier Method. We also performed a set of experiments on a standard object recognition dataset to validate the effectiveness of our framework over state-of-the-art approaches.

In the third method [152], our solution to unsupervised domain adaptation is to learn a domain-invariant representation that is also category discriminative. Domain-invariant representations are realized by minimizing the domain discrepancy. To minimize the domain discrepancy, we propose a novel graph-matching metric between the source and target domain representations. The graph-matching loss considers the cost of matching the source and target graphs constructed from the corresponding representations. The matching consists of both node-to-node matching and edge-to-edge matching between the source and target representation graphs. This second-order matching of edges provides additional structural and geometric information about the representations that are absent on just using the first-order information [57]. The feature extraction network is iteratively optimized to minimize this graph matching loss along with minimizing the mis-classification loss using the source domain labeled data. Our proposed method adopts an adversarial learning scheme where the adversarial loss is a combination of first-order and second-order graph-based matchings between the source and target domain features. It is important to note that our matching approach is local and it considers matching between each instance of

the source and target domain representations. On the other hand, methods like CORAL [54] and those based on MMD [51, 52] are global moment-matching methods that match statistics of the domain distributions.

After the learning has converged and the source and target representations lie in support of each other, we perform an additional refinement of the model. The pseudo-labels (PL) of the confident unlabeled target domain data are used to make sure that target samples lie further from the softmax decision boundary. This allows better generalization to unseen target samples. We expect the refining step to improve the performance further. This is validated by performing experiments on standard image classification adaptation datasets. Results showed our proposed approach outperform previous domain-invariant representation learning approaches. An outline of the proposed framework with the motivations and the impacts is shown in Fig. 2.4.

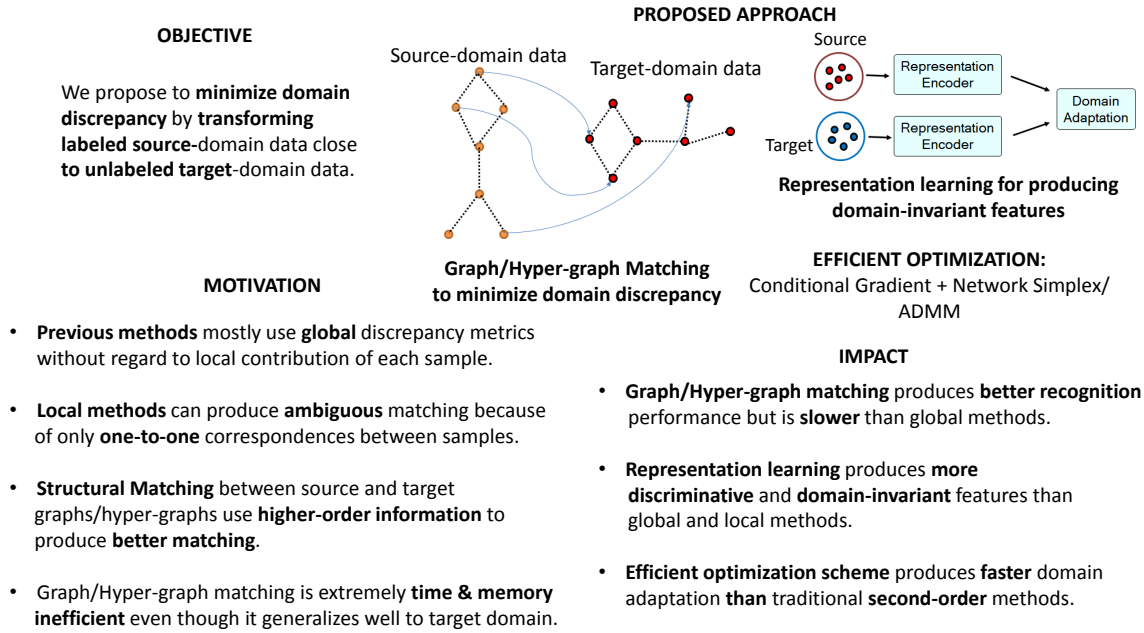


Fig. 2.4. Summary of motivations and impacts of the proposed research framework for unsupervised domain adaptation.

The structure of this Chapter is as follows. Section 2.2 describes the problem definition and notation. Section 2.3 describes the first method in details along with

the experimental details The methodology and the experiments of the second method are described in Section 2.4. In Section 2.5, we describe the third method along with the results. Finally, the whole Chapter is summarized in Section 2.6.

2.2 Problem Definition and Notation

For unsupervised domain adaptation (UDA), we have the source domain data matrix $\mathbf{X}^s \in \mathbb{R}^{n_s \times d}$, vector of source domain data labels $\mathbf{y}^s \in \mathbb{R}^{n_s}$ and the target domain data matrix $\mathbf{X}^t \in \mathbb{R}^{n_t \times d}$. Here, n_s and n_t are the number of source and target samples respectively, and d is the dimension of the feature space. The labels in $\mathbf{y}^s$ range in $\{1, 2, \cdots, C\}$. Both the domains share the same C number of classes. For the purpose of UDA, we transform the source domain data close to the target domain data such that a classifier trained on the transformed source domain predicts well on the target instances.

2.3 Proposed Sample-to-Sample Correspondence Method

In this section, we shall first define the domain adaptation problem [24, 25], and then formulate the proposed correspondence-and-mapping method for the unsupervised domain adaptation problem.

2.3.1 Correspondence-and-Mapping Problem Formulation

Our proposed approach considers the transformation $\mathbf{F}$ as a point-set registration between two point sets, where the source samples $\{\mathbf{x}_i^s\}_{i=1}^{n_s}$ are the moving point set and the target samples $\{\mathbf{x}_i^t\}_{i=1}^{n_t}$ are the fixed point set. In such a case, the registration involves alternately finding the correspondence and mapping between the fixed and moving point sets [156, 157]. The advantage of point-set registration is that it ensures explicit sample-to-sample matching and not moment matching like covariance

in CORAL [44] or MMD [50–53]. As a result, the transformed source domain matches better with the target domain. However, matching each and every sample requires an optimizing variable for each pair of source and target domain samples. If the number of samples increases, so does the number of variables and the optimization procedure may become extremely costly. We shall discuss how to deal with the computational inefficiency later.

For the case when the number of target samples equals to the number of source samples; that is, $n_t = n_s$, the correspondence can be represented by a permutation matrix $\mathbf{P} \in \{0,1\}^{n_s \times n_t}$. Element $[\mathbf{P}]_{ij} = 1$ if the source-domain sample $\mathbf{x}_i^s$ corresponds to the target-domain sample $\mathbf{x}_j^t$, and 0, otherwise. The permutation matrix $\mathbf{P}$ has constraints $\sum_i [\mathbf{P}]_{ij} = 1$ and $\sum_j [\mathbf{P}]_{ij} = 1$ for all $i \in \{1, 2, \ldots, n_s\}$ and $j \in \{1, 2, \ldots, n_t\}$. Hence, if $\mathbf{X}^s \in \mathbb{R}^{n_s \times d}$ and $\mathbf{X}^t \in \mathbb{R}^{n_t \times d}$ be the data matrix of the source-domain and the target-domain data, respectively, then $\mathbf{P}\mathbf{X}^t$ permutes the target-domain data matrix.

As soon as the correspondence is established, a linear or a non-linear mapping must be established between the target samples and the corresponding source samples. Non-linear mapping is involved when there is localized mapping for each sample, and it might also be required in case there is unequal domain shift of each class. The mapping operation should map the source-domain samples as close as possible to the corresponding target-domain samples. This process of finding a correspondence between these transformed source samples and target samples and then finding the mapping will continue iteratively till convergence. This iterative method of alternately finding the correspondence and mapping is similar to feature registration in computer vision [156, 157] but they have not been used or reformulated for unsupervised domain adaptation . In fact, the feature registration methods formulate the problem as a non-convex optimization. Consequently, these methods suffer from local minimum as in [156], and the global optimization technique such as deterministic annealing [157] does not guarantee convergence. Thus, we propose to formulate it as a convex optimization problem to obtain correspondences as a global solution. It is important to note that finding such global and unique solution to the correspondence

accurately is more important because mapping with inaccurate correspondences will undoubtedly yield bad results.

Formulating the proposed unsupervised domain adaptation problem as a convex optimization problem requires the correspondences to have the following properties: (a) *First-order similarity:* The corresponding target-domain samples should be as close as possible to the corresponding source-domain samples. This implies that we want to have the permuted target-domain data matrix $\mathbf{PX}^t$ to be close to the source-domain data matrix $\mathbf{X}^s$, which translates to minimizing the Frobenius norm $||\mathbf{PX}^t - \mathbf{X}^s||_{\mathcal{F}}^2$ in the least-squares sense . (b) *Second-order similarity:* The corresponding target-domain neighborhood should be structurally similar to the corresponding source-domain neighborhood. This structural similarity can be expressed using graphs constructed from the source and target domains. Thus, if the two domains can be thought of as weighted undirected graphs G^s, G^t, structural similarity implies matching edges between the source and the target graphs. The edges of these graphs can be expressed using the adjacency matrices. If $\mathbf{D}^s$ and $\mathbf{D}^t$ are the adjacency matrices of G^s and G^t, respectively, then these adjacency matrices can be found as,

$$[\mathbf{D}^s]_{ij} = \exp(-\frac{||\mathbf{x}_i^s - \mathbf{x}_j^s||_2^2}{\sigma_s^2})$$

$$[\mathbf{D}^t]_{ij} = \exp(-\frac{||\mathbf{x}_i^t - \mathbf{x}_j^t||_2^2}{\sigma_t^2})$$

$$[\mathbf{D}^s]_{ii} = [\mathbf{D}^t]_{ii} = 0,$$

where σ_s and σ_t can be found heuristically as the mean sample-to-sample pairwise distance in the source and target domains, respectively. For the second-order similarity, we want the permuted target domain adjacency matrix $\mathbf{PD}^t\mathbf{P}^T$ to be close to the source domain adjacency matrix (region) $\mathbf{D}^s$, where the superscript T indicates a matrix transpose operation. We formulate it as equivalent to minimizing $||\mathbf{PD}^t\mathbf{P}^T - \mathbf{D}^s||_{\mathcal{F}}^2$. While this cost term geometrically implies the cost of mis-matching edges in the constructed graphs, the first-order similarity term can be thought as

the cost of mis-matching nodes. However, the second-order similarity cost term is bi-quadratic and we want to make it quadratic so that the cost-function is convex and we can apply convex optimization techniques to it. This can be done by post-multiplying $\mathbf{P}\mathbf{D}^t\mathbf{P}^T - \mathbf{D}^s$ by $\mathbf{P}$. Using the permutation matrix properties $\mathbf{P}^T\mathbf{P} = \mathbf{I}$ (orthogonal) and $||\mathbf{A}\mathbf{P}|| = ||\mathbf{A}||$ (norm-preserving), this transformation produces the cost function $||\mathbf{P}\mathbf{D}^t - \mathbf{D}^s\mathbf{P}||_{\mathcal{F}}^2$.

Estimating the correspondence as a permutation matrix in this quadratic setting is NP-hard because of the combinatorial complexity of the constraint on $\mathbf{P}$. We can relax the constraint on the correspondence matrix by converting it from a discrete to a continuous form. The norms (i.e., Frobenius) used in the cost/regularization terms will yield a convex minimization problem if we replace $\mathbf{P}$ with a continuous constraint. Hence, if we relax the constraints on $\mathbf{P}$ to allow for soft correspondences (i.e., replacing $\mathbf{P}$ with $\mathbf{C}$), then an element of $\mathbf{C}$ matrix, $[\mathbf{C}]_{ij}$, represents the probability that $\mathbf{x}_i^s$ corresponds to $\mathbf{x}_j^t$. This matrix $\mathbf{C}$ is called doubly stochastic matrix $\mathcal{D}_B = \{\mathbf{C} \geq \mathbf{0} : \mathbf{C}\mathbf{1} = \mathbf{C}^T\mathbf{1} = \mathbf{1}\}$. $\mathcal{D}_B$ represents a convex hull, containing all permutation matrices at its vertices. (Birkhoff-von-Neumann theorem).

In addition to the graph-matching terms, we add a class-based regularization to the cost function that exploits the labeled information of source-domain data. The group-lasso regularizer $\ell_2 - \ell_1$ norm term is equal to $\sum_j \sum_c ||[\mathbf{C}]_{\mathcal{I}_c j}||_2$, where $||\cdot||_2$ is the ℓ_2 norm and $\mathcal{I}_c$ contains the indices of rows of $\mathbf{C}$ corresponding to the source-domain samples of class c. In other words, $[\mathbf{C}]_{\mathcal{I}_c j}$ is a vector consisting of elements $[\mathbf{C}]_{ij}$, where i^{th} source sample belongs to class c and the j^{th} sample is in the target domain. Minimizing this group-lasso term ensures that a target-domain sample only corresponds to the source-domain samples that have the same label.

It is important to note that the solution to the relaxed problem may not be equal or even close to the original discrete problem. Even then, the solution of the relaxed problem need not be projected onto the set of permutation matrices to get our final solution. This is because the graphs constructed using the source samples and the target samples are far from isomorphic for real datasets. Therefore, we do

not expect exact matching between the nodes (samples) of each graph (domain) and soft correspondences may serve better. As an example, consider that a source sample $\mathbf{x}_i^s$ is likely to correspond to both $\mathbf{x}_j^t$ and $\mathbf{x}_k^t$. In that case, it is more appropriate to have correspondences $[\mathbf{C}]_{ij} = 0.7$ and $[\mathbf{C}]_{ik} = 0.3$ assigned to the target samples, rather than the exact correspondences $[\mathbf{C}]_{ij} = 1$ and $[\mathbf{C}]_{ik} = 0$ or vice-versa. Thus, we can formulate our optimization problem of obtaining $\mathbf{C}$ as follows:

$$\begin{aligned} \min_{\mathbf{C}} f(\mathbf{C}) =& ||\mathbf{C}\mathbf{X}^t - \mathbf{X}^s||_{\mathcal{F}}^2/(n_s d) + \\ & \lambda_s ||\mathbf{C}\mathbf{D}^t - \mathbf{D}^s\mathbf{C}||_{\mathcal{F}}^2 + \lambda_g \sum_j \sum_c ||[\mathbf{C}]_{\mathcal{I}_c j}||_2 \\ & \text{such that } \ \mathbf{C} \geq \mathbf{0},\ \mathbf{C}\mathbf{1}_{n_t} = \mathbf{1}_{n_s},\ \text{and } \mathbf{C}^T\mathbf{1}_{n_s} = \mathbf{1}_{n_t}, \end{aligned} \tag{2.1}$$

where λ_s and λ_g are the parameters weighing the second-order similarity term and class-based regularization term, respectively; $\mathbf{1}_{n_s}$ and $\mathbf{1}_{n_t}$ are column vectors of size n_s and n_t, respectively, and the superscript T indicates a matrix transpose operation. The assumption that $n_t = n_s$ is strict and it needs to be relaxed to allow more realistic situations such as $n_t \neq n_s$. To analyze what modification is required to the optimization problem in Eq. (2.1), we explore further to understand the correspondences properly. In the case of $n_t = n_s$, we have one-to-one correspondences between each source sample and each target sample. However, for the case $n_t \neq n_s$, we must allow multiple correspondences. Initially, the constraint $\mathbf{C}\mathbf{1}_{n_t} = \mathbf{1}_{n_s}$ implies that the sum of the correspondences of all the target samples to each source sample is one. The second equality constraint $\mathbf{C}^T\mathbf{1}_{n_s} = \mathbf{1}_{n_t}$ implies that the sum of correspondences of all the source samples to each target sample is one. However, if $n_t \neq n_s$, the sum of correspondences of all the source samples to each target sample should increase proportionately by $\frac{n_s}{n_t}$ to allow for the multiple correspondences. This is reflected in the following optimization problem.

Problem UDA-1

$$\min_{\mathbf{C}} f(\mathbf{C}) = ||\mathbf{C}\mathbf{X}^t - \mathbf{X}^s||_{\mathcal{F}}^2/(n_s d) + \lambda_s||\mathbf{C}\mathbf{D}^t - (\frac{n_t}{n_s})\mathbf{D}^s\mathbf{C}||_{\mathcal{F}}^2 \qquad (2.2)$$
$$+ \lambda_g \sum_j \sum_c |||[\mathbf{C}]_{\mathcal{I}_c j}||_2$$

such that $\mathbf{C} \geq \mathbf{0},\ \mathbf{C}\mathbf{1}_{n_t} = \mathbf{1}_{n_s},$ and $\mathbf{C}^T\mathbf{1}_{n_s} = (\frac{n_s}{n_t})\mathbf{1}_{n_t}$

for $n_t \neq n_s$.

2.3.2 Correspondence-and-Mapping Problem Solution

Problem UDA-1 is a constrained convex optimization problem and can easily be solved by interior-point methods [159]. In general, the time complexity of these interior-point-methods for conic programming is $O(N^{3.5})$, where N is the total number of the variables [160]. If we have n_s and n_t as source and target samples, respectively, then the time complexity becomes $O(n_s^{3.5} n_t^{3.5})$. Also, the interior-point method is a second-order optimization method. Hence, it requires storage space of the Hessian, which is $O(N^2) \sim O(n_s^2 n_t^2)$. This space complexity is more alarming and does not scale well with an increasing number of variables. If n_t and n_s are greater than 100 points, it results in memory/storage-deficiency problems in most personal computers. Thus, we need to employ a different optimization procedure so that the proposed UDA approach can be widely used without memory-deficiency problem. We could think of first-order methods of solving the constrained optimization problem, which require computing gradients but do not require storing the Hessians.

First-order methods of solving the constrained optimization problem can be broadly classified into projected-gradient methods and conditional gradient (CG) methods [161]. The projected-gradient method is similar to the normal gradient-descent method except that for each iteration, the iterate is projected back into the constraint set. Generally, the projected gradient-descent method enjoys the same convergence rate as the unconstrained gradient-descent method. However, for the projected gradient-descent method to be efficient, the projection step needs to be inexpensive. With an

increasing number of variables, the projection step can become costly. Furthermore, the full gradient updating may destroy the structure of the solutions such as sparsity and low rank. The conditional gradient method, on the other hand, maintains the desirable structure of the solution such as sparsity by solving the successive linear minimization sub-problems over the convex constraint set. Since we expect our correspondence matrix $\mathbf{C}$ to be sparse, we shall employ the conditional gradient method for our problem. In fact, [162] points out that convex optimization problems over convex hulls of atomic sets, which are relaxations of NP-hard problems are directly suitable for the conditional gradient method. This is similar to the way we formulate our problem by relaxing $\mathbf{P}$ matrix to $\mathbf{C}$ as shown in Algorithm 1.

Algorithm 1: Conditional Gradient Method (CG).

Given : $\mathbf{C}_0 \in \mathcal{D}$, $t = 1$

Repeat

$\mathbf{C}_d = \arg\min_{\mathbf{C}} \mathrm{Tr}(\nabla_C f(\mathbf{C}_0)^T \mathbf{C}), \quad \text{such that} \quad \mathbf{C} \in \mathcal{D}$

$\mathbf{C}_1 = \mathbf{C}_0 + \alpha(\mathbf{C}_d - \mathbf{C}_0), \quad \text{for} \quad \alpha = \frac{2}{t+2}$

$\mathbf{C}_0 = \mathbf{C}_1 \quad \text{and} \quad t = t + 1$

Until Convergence or Fixed Number of Iterations

Output : $\mathbf{C}_0 = \arg\min_{\mathbf{C}} f(\mathbf{C}) \quad \text{such that} \quad \mathbf{C} \in \mathcal{D}$

As described in Algorithm 1 of the conditional gradient method, we have to solve the linear programming problem, $\min_{\mathbf{C}} \mathrm{Tr}(\nabla_C f(\mathbf{C}_0)^T \mathbf{C})$, such that $\mathbf{C} \in \mathcal{D} = \{\mathbf{C} : \mathbf{C} \geq \mathbf{0}, \mathbf{C}\mathbf{1}_{n_t} = \mathbf{1}_{n_s}, \mathbf{C}^T\mathbf{1}_{n_s} = (\frac{n_s}{n_t})\mathbf{1}_{n_t}\}$. Here $\mathrm{Tr}(\cdot)$ is the Trace operator. The gradient $\nabla_{\mathbf{C}} f$ can be found from the equation:

$$\nabla_{\mathbf{C}} f = \nabla_{\mathbf{C}} f_1/(n_s d) + \lambda_s \nabla_{\mathbf{C}} f_2 + \lambda_g \nabla_{\mathbf{C}} f_3, \tag{2.3}$$

where f_1, f_2, and f_3 are $||\mathbf{C}\mathbf{X}^t - \mathbf{X}^s||_{\mathcal{F}}^2$, $||\mathbf{C}\mathbf{D}^t - (\frac{n_t}{n_s})\mathbf{D}^s\mathbf{C}||_{\mathcal{F}}^2$, and $\sum_j \sum_c ||[\mathbf{C}]_{\mathcal{I}_c j}||_2$, respectively.

The gradients are obtained as follows-

$$\nabla_{\mathbf{C}} f_1 = 2(\mathbf{C}\mathbf{X}^t - \mathbf{X}^s)(\mathbf{X}^t)^T$$

$$\nabla_{\mathbf{C}} f_2 = 2\mathbf{C}\mathbf{D}^t(\mathbf{D}^t)^T - 2r\mathbf{D}^s\mathbf{C}(\mathbf{D}^t)^T - 2r(\mathbf{D}^s)^T\mathbf{C}\mathbf{D}^t + 2r^2(\mathbf{D}^s)^T\mathbf{D}^s\mathbf{C}$$

where $r = \frac{n_t}{n_s}$ and

$$\frac{\partial f_3}{\partial [\mathbf{C}]_{ij}} = \begin{cases} \frac{[\mathbf{C}]_{ij}}{||[\mathbf{C}]_{\mathcal{I}_c(i)j}||_2} , & \text{if } ||[\mathbf{C}]_{\mathcal{I}_c(i)j}||_2 \neq 0; \\ 0 , & \text{otherwise;} \end{cases}$$

Here, $c(i)$ is the class corresponding to the i^{th} sample in the source domain and $\mathcal{I}_c(i)$ contains the indices of source samples belonging to class $c(i)$. After the gradient $\nabla_{\mathbf{C}} f$ is found from $\nabla_{\mathbf{C}} f_1$, $\nabla_{\mathbf{C}} f_2$, $\nabla_{\mathbf{C}} f_3$ using Eq. 2.3, we need to solve for the linear programming problem.

The linear programming problem can be solved easily using simplex methods used in solvers such as MOSEK [163]. However, using such solvers would not make our method competitive in terms of time efficiency. Hence, we convert this linear programming problem into a *min-cost flow* problem, which can then be solved very efficiently using a network simplex approach [164].

Let the gradient $\nabla_{\mathbf{C}} f(\mathbf{C}_0)$ be $\mathbf{G}/n_s$ and the correspondence matrix variable be $\mathbf{C} = n_s\mathbf{T}$. Then, the linear programming (LP) problem translates to $\min_{\mathbf{T}} \text{Tr}(\mathbf{G}^T\mathbf{T})$ such that $\mathbf{T} \geq \mathbf{0}, \mathbf{T}\mathbf{1}_{n_t} = \mathbf{1}_{n_s}/n_s, \mathbf{T}^T\mathbf{1}_{n_s} = \mathbf{1}_{n_t}/n_t$. This LP problem has an equivalence with the min-cost flow problem on the following graph:

- The graph is bipartite with n_s source nodes and n_t sink nodes.
- The supply at each source node is $1/n_s$ and the demand at each sink node is $1/n_t$.
- Cost of the edge connecting the i^{th} source node to the j^{th} sink node is given by $[\mathbf{G}]_{ij}$. Capacity of each edge is ∞.

Using this configuration, the min-cost flow problem is solved using the network simplex. Details of the network-simplex method is omitted and one can refer [164]. The network simplex method is an implementation of the traditional simplex method for LP problems, where all the intermediate operations are performed on graphs. Due

to the structure of min-cost flow problems, network-simplex methods provide results significantly faster than traditional simplex methods. Using this network-simplex method, we obtain the solution $\mathbf{T}^*$, where $[\mathbf{T}^*]_{ij}$ is the flow obtained on the edge connecting the i^{th} source node to the j^{th} sink node. From that, we obtain $\mathbf{C}_d = n_s\mathbf{T}^*$ and proceed with that iteration of conditional gradient (CG) method as in *Algorithm 1*. In the above CG method, we also need an initial $\mathbf{C}_0$ and $\mathbf{C}_0$ can be defined as the solution to the LP problem, $\min_{\mathbf{C}} \text{Tr}(\mathbf{D}^T\mathbf{C})$ such that $\mathbf{C} \in \mathcal{D} = \{\mathbf{C} : \mathbf{C} \geq 0, \mathbf{C}\mathbf{1}_{n_t} = \mathbf{1}_{n_s}, \mathbf{C}^T\mathbf{1}_{n_s} = (\frac{n_s}{n_t})\mathbf{1}_{n_t}\}$, where $[\mathbf{D}]_{ij} = ||\mathbf{x}_i^s - \mathbf{x}_j^t||_2$. This is also solved by the network simplex approach after converting this LP problem into its equivalent min-cost flow problem as described previously. After we obtain $\mathbf{C}^*$ from the CG algorithm, it is then used to find the corresponding target samples $\mathbf{X}_c^t = \mathbf{C}^*\mathbf{X}^t$. Then, the mapping $\mathbf{M}(\cdot)$ from the source domain to the target domain is found by solving the following regression problem $\mathbf{M}(\cdot) : \mathcal{X}^s \rightarrow \mathcal{X}^t$, with each row of $\mathbf{X}^s$ as an input data sample and the corresponding row of $\mathbf{X}_c^t$ as an output data sample. The choice of regressors can be linear functions, neural networks, and kernel machines with proper regularization. Once the mapping $\mathbf{M}^*(\cdot)$ is found out, a source-domain sample $\mathbf{x}^s$ can be mapped to the target domain by applying $\mathbf{M}^*(\mathbf{x}^s)$. This completes one iteration of finding the correspondence and the mapping. For the next cycle, we solve Problem UDA with the mapped source samples as $\mathbf{X}^s$ and subsequently find the new mapping. The number of iterations N_T of alternatively finding correspondence and mapping is an user-defined variable. The full domain adaptation algorithm is outlined in Algorithm 2.

2.3.3 Experimental Results and Discussions

To evaluate and validate the proposed sample-sample correspondence and mapping method for unsupervised domain adaptation, computer simulations were performed on a toy dataset and then on image classification and sentiment classification tasks. Our results were compared with previous published methods. For compar-

Algorithm 2: Unsupervised Domain Adaptation using dataset registration.

Given : Source Labeled Data $\mathbf{X}^s$ and $\mathbf{Y}^s$, and Target Unlabeled Data $\mathbf{X}^t$

Parameters : $\lambda_s, \lambda_g, N_T$

Initialize : $t = 0$

Repeat

$\mathbf{C}^* = \text{argmin} f(\mathbf{C})$ such that $\mathbf{C} \in \mathcal{D}$ (Find Correspondence using CG method)

Regress $\mathbf{M}(\cdot)$ s.t. $\mathbf{X}^s \xrightarrow{\mathbf{M}} \mathbf{C}^*\mathbf{X}^t$ (Find Mapping)

Map $\mathbf{X}^s = \mathbf{M}(\mathbf{X}^s)$ and $t = t + 1$

Until $t = N_T$

Output : Adapted Source Data $\mathbf{X}^s$,$\mathbf{Y}^s$ to learn classifier.

isons, we used the reported accuracies or conduct experiments with the available source code. Since we are dealing with unsupervised domain adaptation, it is not possible to cross-validate our hyper-parameters λ_s, λ_g, and N_T. Unless explicitly mentioned, we reported the best results obtained over the hyper-parameter ranges λ_s and λ_g in $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$ and $N_T = 1$. In our simulations, we found that using $N_T > 1$ only provides a tiny bump in performance or no improvement in performance at all. This is because the source samples have already been transformed close to the target samples and further transformation does not affect recognition accuracies. After the correspondence was found, we considered mapping between the corresponding samples. For the mapping, we used a linear mapping $\mathbf{W} \in \mathbb{R}^{d \times d}$ with a regularization of 0.001. d is the dimension of the feature space in which the data lies.

Toy Dataset: Two Interleaving Moons

For the first experiment, we used the synthetic dataset of interleaving moons previously used in [49,165]. The dataset consists of 2 domains. The source domain consists

of 2 entangled moon's data. Each moon is associated with each class. The target domain consists of rotating the source domain. This can be considered as a domain-adaptation problem with increasing rotation angle implying increasing difficulty of the domain-adaptation problem. Since the problem is low dimensional, it allowed us to visualize the effect of our domain-adaptation method appropriately. Figures 2.5(a) and 2.5(b) show an example of the source-domain data and the target-domain data respectively, and Fig. 2.5(c) shows the adapted source-domain data using the proposed approach. The results showed that the transformed source domain becomes close to the target domain.

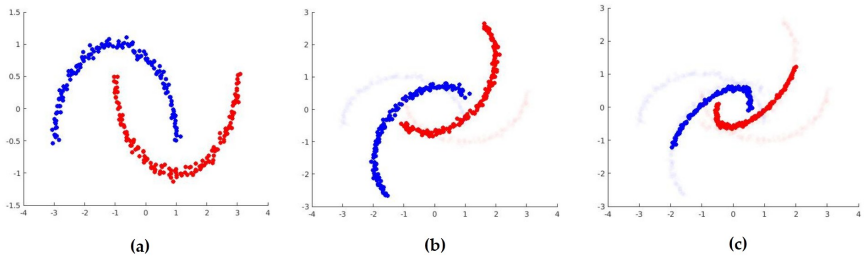


Fig. 2.5. (a) Source-domain data. (b) Target-domain data consists of a 50-degree rotation of the source-domain data. (c) Transformed source-domain data is now aligned with the target-domain data.

For testing on this toy dataset, we used the same experimental procedure as in [49, 165]. We sampled 150 instances from both the domains. The test data consisted of 1000 examples from the target domain distribution. For classification, we used a Gaussian kernel-based SVM, cross-validated using 5-fold method. The experiments were conducted over 10 trials and the mean accuracy was reported. At this juncture, it is important to note that choosing the classifier for domain adaptation is important. For example, the two classes in the interleaving moon dataset are not linearly separable at all. So, a linear kernel SVM would not classify the moons accurately and it would result in poor performance in the target domain as well. That is why we need a Gaussian Kernel SVM. So, we have to make sure that we choose a classifier that works well with the source dataset in the first place.

We compared our results with the DA-SVM [29]- a domain-adaptive support vector machine approach, PBDA [165]- which is a PAC-Bayesian based domain adaptation method, and different versions of the optimal transport approach [49]. OT-exact is the basic optimal transport approach. OT-IT is the information theoretic version with entropy regularization. OT-GL and OT-Laplace has additional group and graph based regularization, respectively. From our results in Table 2.1, we see that for low rotation angles, the OT-GL-based method dominates and our proposed method yields satisfactory results. But for higher angles ($\geq$ 50 °), our proposed method clearly dominates by a large margin. This is because we have taken into consideration second-order structural similarity information. For higher-rotation angles, the point-to-point sample distance is high. However, similar structures in the source and target domains can still correspond to each other. In other words, the adjacency matrices, which depend on relative distances between samples, can still be matched and do not depend on higher rotation angles between the source and target domains. That is why our proposed method out-performed other methods for large discrepancies between the source and target distributions.

Table 2.1. Accuracy results over 10 trials for the toy dataset domain-adaptation problem for varying degree of rotation between source and target domain.

Angle (°)	10	20	30	40	50	70	90
SVM-NA	100	89.6	76.0	68.8	60.0	23.6	17.2
DASVM	100	100	74.1	71.6	66.6	25.3	18.0
PBDA	100	90.6	89.7	77.5	59.8	37.4	31.3
OT-exact	100	97.2	93.5	89.1	79.4	61.6	49.3
OT-IT	100	99.3	94.6	89.8	87.9	60.2	49.2
OT-GL	100	100	**100**	**98.7**	81.4	62.2	49.2
OT-Laplace	100	100	99.6	93.8	79.9	59.8	47.6
Ours	100	100	96	87.4	**83.9**	**78.4**	**72.2**

We further provided the time comparison between the network simplex method (N-S) and MOSEK (M) for increasing number of samples of the toy dataset in Table 2.2. Results showed that the network simplex method is very fast compared to a general purpose linear programming solver like MOSEK.

Table 2.2. Time comparison (in seconds) of the two solvers for increasing sample size. The sample size is the number of samples per class per domain of the interleaving moon toy dataset. The target domain has a rotation of 50° with the source domain. We use $N_T = 1$. Implementation was in MATLAB in a workstation with Intel Xeon(R) CPU E5-2630 v2 and 40 GB RAM. Results are reported over 10 trials.

n	25	50	75	100	125	150	175	200
M	62.1	83.1	103.4	128.5	387.7	680.1	1028.3	1577.6
N-S	1.5	4	6.9	10.1	16.9	23.5	31.2	41.3

Real Dataset: Image Classification

We next evaluated the proposed method on image classification tasks. The image classification tasks that we considered were digit recognition and object recognition. The classifier used was 1-NN (Nearest Neighbor). 1-NN is used for experiments with images because it does not require cross-validating hyper-parameters and has been used in previous work as well [41,49] The 1-NN classifier is trained on the transformed source-domain data and tested on the target-domain data. Instances of the image dataset are shown in Fig. 2.6 (a),(b) and (e). Generally, we cannot directly cross-validate our hyper-parameters λ_s and λ_g on the unlabeled target domain data making it impractical for real-world applications. However, for practical transfer learning purposes, a reverse validation (RV) technique [166] was developed for tuning the hyper-parameters. We have carried out experiments with a variant of the method to tune λ_s and λ_g for our UDA approach..

For a particular hyper-parameter configuration, we divide the source domain data into K folds. We use one of the folds as the validation set. The remaining source

Fig. 2.6. Instances of the real dataset used. At the top left, we see that USPS has the worse resolution compared to MNIST handwriting dataset. At the bottom left, we have instances of the Amazon review dataset. There is a shift in textual domain when reviewing for different products. On the right, we have the Caltech-Office dataset and we see that there are differences in illumination, quality, pose, presence/absence of background across different domains.

data and the whole target data are used for domain adaptation. The classifier trained using the adapted source data is used to generate pseudo-labels for the target data. Another classifier is trained using the target domain data and its pseudo-labels. This classifier is then tested on the held-out source domain data after adaptation. The accuracy obtained is repeated and averaged over all the K folds. This reverse-validation approach is repeated over all hyper-parameter configurations. The optimal hyper-parameter configuration is the one with the best average validation accuracy. Using the obtained optimal hyper-parameter configuration, we then carry out domain adaptation over all the source and target domain data and report the accuracy over the target domain dataset. We used $K = 5$ folds for all the real-data experiments. Thus, we showed the results using this RV approach in addition to the best obtained results over the hyper-parameters. In majority of the cases we would see that the result obtained using the reverse validation approach matches the best obtained results suggesting that the hyper-parameters can be automatically tuned successfully.

Digit Recognition

For the source and target domains, we used 2 datasets – USPS (U) and MNIST (M). These datasets have 10 classes in common (0-9). The dataset consists of randomly sampling 1800 and 2000 images from USPS and MNIST, respectively. The MNIST digits have 28×28 resolution and the USPS 16×16. The MNIST images were resized to that of the USPS. The normalized grey levels were used to obtain a common 256-dimensional feature space for both domains.

Object Recognition

For object recognition, we used the popular Caltech-Office dataset [40, 41, 49, 167, 168]. This domain-adaptation dataset consists of images from 4 different domains: *Amazon* (A) (E-commerce), *Caltech-256* [169] (C) (a repository of images), *Webcam* (W) (webcam images), and *DSLR* (D) (images taken using DSLR camera). The differences between domains are due to the differences in quality, illumination, pose and also the presence and absence of backgrounds. The features used are the shallow SURF features [170] and deep-learning feature sets [171] – *decaf6* and *decaf7*. The SURF descriptors represent each image as a 800-bin histogram. The histogram is first normalized to represent a probability and then reduced to standard z-scores. On the other hand, the deep-learning feature sets, *decaf6* and *decaf7*, are extracted as the activations from the fully connected 6th and 7th layers of AlexNet. The features are 4096-dimensional.

For our experiments, we considered a random selection of 20 samples per class (with the exception of 8 samples per class for the DSLR domain) for the source domain. The target-domain data is split equally. One half of the target-domain data is used for domain adaptation and the other half is used for testing. This is in accordance with the protocol followed in [49]. The accuracy is reported on the test data over 10 trials of the experiment.

We compared our approach against (a) the no adaptation baseline (NA), which consists of carrying out classification without domain adaptation; (b) Geodesic Flow Kernel (GFK) [41]; (c) Transfer Subspace Learning (TSL) [172]; (d) Joint Distribution Adaptation (JDA) [173]; (e) Optimal Transport [49] with its variants - OT-IT and OT-GL. Here, TSL and JDA match distribution moments while OT-IT, OT-GL and our method match domain samples.

From Table 2.3, we see that in almost all the cases, the OT-GL and our proposed method dominated over other methods, suggesting that sample-matching methods perform better than moment-matching methods. For the handwritten digit recognition tasks (U $\rightarrow$ M and M $\rightarrow$ U), our proposed method clearly out-performs GFK, TSL and JDA, but is slightly out-performed by OT-GL. This might be because the handwritten digit datasets U and M do not contain enough structurally similar regions to exploit the second-order similarity cost term. For the Office-Caltech dataset, the only time our proposed method was beaten by a moment-matching method was W $\rightarrow$ D, though by a slight amount. This is because W and D are closest pair of domains and using sample-based matching does not have outright advantage over moment-matching. The fact that W and D have the closest pair of domains is evident form the NA accuracy of 53.62, which is the best among NA accuracies of the Office-Caltech domain-adaptation tasks.

We have performed a runtime comparison in terms of the CPU time in seconds of our method with other methods and have shown the results in Table 2.4. The experiments performed are over the same dataset as used in Table 2.3. From Table 2.4, we see that local methods like OT-GL and our method generally take more time than moment-matching method like JDA. Our method takes more time compared with OT-GL because of time taken in constructing adjacency matrices for the second order cost term. Overall, the time taken for domain adaptation between USPS and MNIST datasets is more because they contain relatively larger number of samples, compared to the Office-Caltech dataset.

Table 2.3. Domain-adaptation results for digit recognition using USPS and MNIST datasets and object recognition with the Office-Caltech dataset using SURF features.

Tasks	NA	GFK	TSL	JDA	OT-GL	Ours	Ours (RV)
U → M	39.00	44.16	40.66	54.52	**57.85**	56.90	56.90
M → U	58.33	60.96	53.79	60.09	**69.96**	68.44	66.24
C → A	20.54	35.29	45.25	40.73	44.17	**46.67**	**46.67**
C → W	18.94	31.72	37.35	33.44	38.94	**39.48**	**39.48**
C → D	19.62	35.62	39.25	39.75	**44.50**	42.88	40.12
A → C	22.25	32.87	38.46	33.99	34.57	**38.51**	**38.51**
A → W	23.51	32.05	35.70	36.03	37.02	**38.69**	**38.69**
A → D	20.38	30.12	32.62	32.62	**38.88**	36.12	36.12
W → C	19.29	27.75	29.02	31.81	**35.98**	33.81	32.83
W → A	23.19	33.35	34.94	31.48	**39.35**	37.69	37.69
W → D	53.62	79.25	80.50	**84.25**	84.00	84.10	84.10
D → C	23.97	29.50	31.03	29.84	32.38	**32.78**	**32.78**
D → A	27.10	32.98	36.67	32.85	37.17	**38.33**	37.61
D → W	51.26	69.67	77.48	80.00	81.06	**81.12**	**81.12**

We have also reported the results of Office-Caltech dataset using *decaf6* and *decaf7* features in Tables 2.5 and 2.6, respectively. The baseline performance of the deep-learning features are better than SURF features because they are more robust and contain higher-level representations. Expectedly, the *decaf7* features have better baseline performance than *decaf6* features. However, DA methods can further increase performance over the robust deep features. In Tables 2.5 and 2.6, we see that our proposed method dominates over JDA and OT-IT but is in close competition with OT-GL. We also noted that using *decaf7* instead of *decaf6* creates only a small incremental improvement in performance because most of the adaptation has already been performed by our proposed domain-adaptation method.

Table 2.4. CPU time (seconds) comparison of different domain adaptation algorithms.

Task	NA	GFK	TSL	JDA	OT-GL	Ours
U→M	1.24	2.62	567.8	82.34	171.84	201.23
M→U	1.13	2.43	522.37	81.13	168.23	196.15
C→A	0.46	2.6	382.98	41.6	85.95	99.9
C→W	0.24	1.45	157.52	37.89	78.73	101.1
C→D	0.36	1.35	117.81	37.33	61.17	63.38
A→C	0.54	2.69	462.12	40.11	105.87	126.18
A→W	0.39	1.47	153.95	37.63	86.12	100.21
A→D	0.42	1.31	115.87	36.82	69.29	82.1
W→C	0.33	2.92	461.1	42.39	98.26	111.2
W→A	0.61	2.52	388.23	41.64	94.38	101.45
W→D	0.34	1.37	117.47	37.9	76.5	79.25
D→C	0.45	2.36	364.13	39.75	106.21	118.12
D→A	0.43	2.14	310.18	41.24	98.41	115.35
D→W	0.24	1.05	93.73	34.62	76.23	88.69

As seen in Fig. 2.7, the source-domain samples are transformed to be near the target-domain samples using our proposed method. Therefore, we expect a classifier trained on the transformed source samples to perform better on the target-domain data.

We have also studied the effects of varying the regularization parameters on domain-adaptation performance. In Fig. 2.8, the blue line shows the accuracy when both $\lambda_s = \lambda_g = 0$. When $\lambda_s = 0$, best performance is obtained for $\lambda_g = 0.1$. When $\lambda_g = 0$, best performance is obtained for $\lambda_g = 1$. For $\lambda_s, \lambda_g > 1$, performance degrades (not shown) because we have put excess weight on the regularization terms of second-order structural similarity and group-lasso than on the first-order point-wise similarity cost term. Thus, the presence of second-order and regularization term,

Table 2.5. Domain-adaptation results for the Office-Caltech dataset using *decaf6* features.

Task	NA	JDA	OT-IT	OT-GL	Ours	Ours(RV)
C→A	79.25	88.04	88.69	**92.08**	91.92	89.91
C→W	48.61	79.60	75.17	**84.17**	83.58	81.23
C→D	62.75	84.12	83.38	87.25	**87.50**	**87.50**
A→C	64.66	81.28	81.65	85.51	**86.67**	85.63
A→W	51.39	80.33	78.94	**83.05**	81.39	81.39
A→D	60.38	86.25	85.88	85.00	**87.12**	**87.12**
W→C	58.17	81.97	74.80	81.45	**82.13**	81.64
W→A	61.15	90.19	80.96	**90.62**	88.87	88.87
W→D	97.50	98.88	95.62	96.25	**98.95**	**98.95**
D→C	52.13	81.13	77.71	**84.11**	83.72	83.72
D→A	60.71	91.31	87.15	92.31	**92.65**	**92.65**
D→W	85.70	**97.48**	93.77	96.29	96.69	96.13

weighted in the right amount is justified as it improves performance over when only the first-order term is present.

We have also studied the effect of group-lasso regularization parameter (λ_g) on the quality of the correspondence matrix $\mathbf{C}$ obtained for a domain-adaptation task. Visually the second plot from the left in Fig. 2.9 appears to discriminate the 10 classes best. Accordingly, this parameter configuration ($\lambda_s = 0, \lambda_g = 0.1, N_T = 1$) realizes the best performance as shown in the previous Fig. 2.8.

Sentiment Classification

We have also evaluated our proposed method on sentiment classification using the standard Amazon review dataset [175]. This dataset contains Amazon reviews on

Table 2.6. Domain-adaptation results for the Office-Caltech dataset using *decaf7* features.

Task	NA	JDA	OT-IT	OT-GL	Ours	Ours(RV)
C→A	85.27	89.63	91.56	**92.15**	91.85	91.85
C→W	65.23	79.80	82.19	83.84	**85.36**	**85.36**
C→D	75.38	85.00	85.00	85.38	**85.88**	**85.88**
A→C	72.80	82.59	84.22	**87.16**	86.67	85.39
A→W	63.64	83.05	81.52	84.50	**86.09**	85.36
A→D	75.25	85.50	86.62	85.25	**87.37**	**87.37**
W→C	69.17	79.84	81.74	**83.71**	82.80	82.80
W→A	72.96	90.94	88.31	**91.98**	90.15	89.31
W→D	98.50	98.88	98.38	91.38	**99.00**	**99.00**
D→C	65.23	81.21	82.02	**84.93**	82.20	82.20
D→A	75.46	91.92	92.15	**92.92**	92.60	92.15
D→W	92.25	97.02	96.62	94.17	**97.10**	**97.10**

4 domains: Kitchen items (K), DVD (D), Books (B) and Electronics (E). Instances of the dataset are shown in Fig. 2.6 (c),(d). The dimensionality of the bag-of-word features was reduced by keeping the top 400 features having maximum mutual information with class labels. This pre-processing was also carried out in [42, 44] without losing performance. For each domain, we used 1000 positive and 1000 negative reviews. For each domain-adaptation task, we used 1600 samples (800 positive and 800 negative) from each domain as the training dataset. The remaining 400 samples (200 positive and 200 negative) were used for testing. The classifier used is a 1-NN classifier since it is parameter free. The mean-accuracy was reported over 10 random training/test splits.

We compared our proposed approach to a recently proposed unsupervised domain-adaptation approach known as Correlation Alignment (CORAL) [44]. CORAL is a simple and efficient approach that aligns the input feature distributions of the source

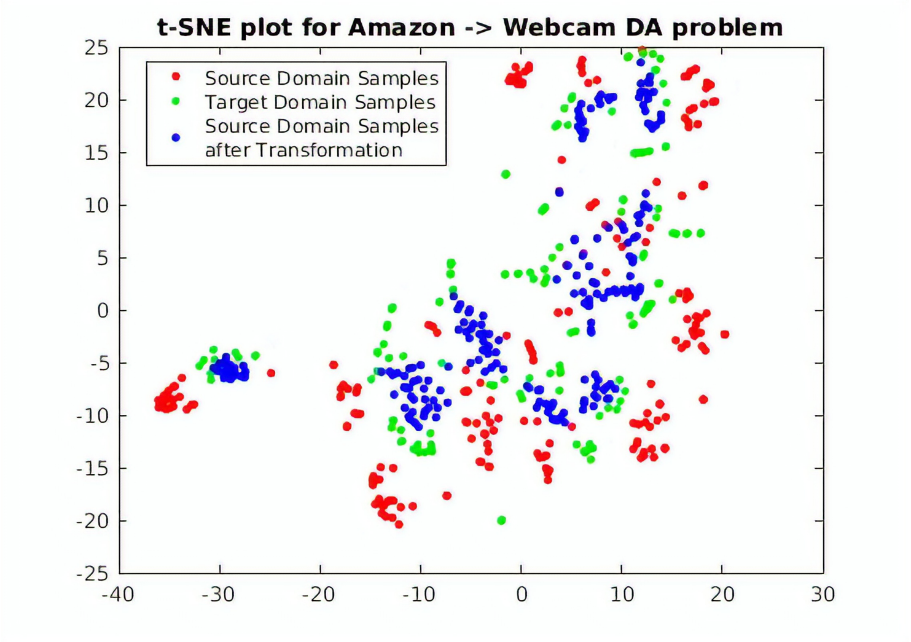


Fig. 2.7. t-SNE [174] visualization of a single trial of Amazon to Webcam DA problem using *decaf6* features.

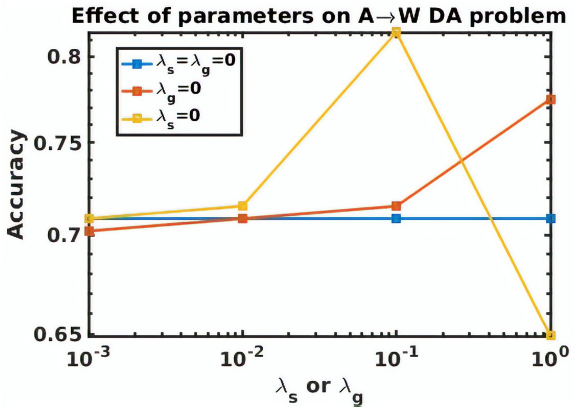


Fig. 2.8. Effect of varying regularization parameters λ_s and λ_g on the accuracy of Amazon (source domain) to Webcam (target domain) visual domain-adaptation problem for fixed $N_T = 1$.

and target domains by exploring their second-order statistics. Firstly, it computes the covariance statistics in each domain and then applies whitening and re-coloring linear transformation to the source features. Results in Table 2.7 showed that our proposed method outperforms CORAL in all the domain-adaptation tasks. Our pro-

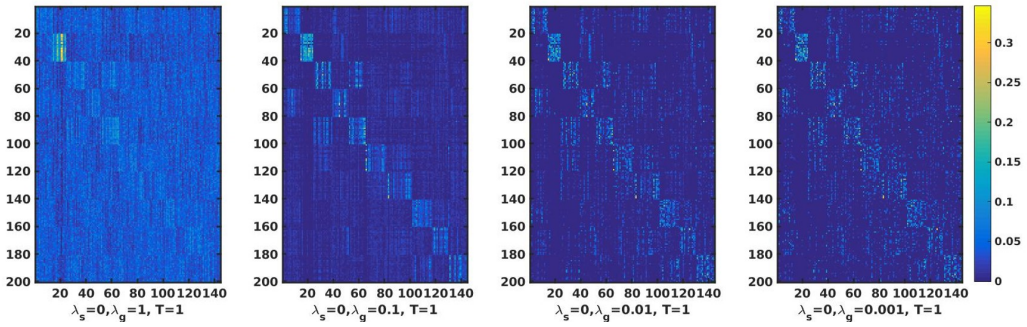


Fig. 2.9. The optimal correspondence matrix $\mathbf{C}$ for 4 different parameter settings visualized as a colormap, with $\lambda_s = 0, N_T = 1$. The task involved was the Amazon to Webcam domain adaptation.

posed method has better performance because CORAL matches covariances while our method matches samples explicitly through point-wise and pair-wise matching. Moreover, CORAL does not use source-domain label information. Our method uses source-domain label information through the group-lasso regularization. However, CORAL is quite fast in transforming the source samples compared to our method. For a single trial, CORAL took about a second while our proposed method took about a few minutes.

Table 2.7. Accuracy results of unsupervised domain-adaptation tasks for the Amazon reviews dataset.

Tasks	**K→D**	**D→B**	**B→E**	**E→K**	**K→B**	**D→E**
NA	58.6	63.4	58.5	66.5	59.3	57.9
CORAL	59.9	66.5	59.5	67.5	59.2	59.5
Ours	**63.5**	**69.5**	**62.0**	**69.5**	**64.5**	**61.2**
Ours (RV)	60.9	**69.5**	**62.0**	**69.5**	**64.5**	59.0

2.4 Proposed Regularized Hyper-graph Matching Method

Our proposed method consists of the following steps: (1) A mathematical framework using all the first-, second- and third-order relations to match the source- and target-domain samples along with a regularization using labels of the source-domain data. (2) Computationally efficient method of obtaining the solution of the optimization problem by solving a series of sub-problems using Alternating Direction Multiplier Method (ADMM). Moreover, we perform an initial clustering to select the most relevant instances and thus reduce the number of data points to be used in the optimization approach. (3) Experimental evaluation on an object recognition dataset with analysis of the effect of each cost term. Fig. 2.10 shows the intuition of our method in a two-dimensional setting.

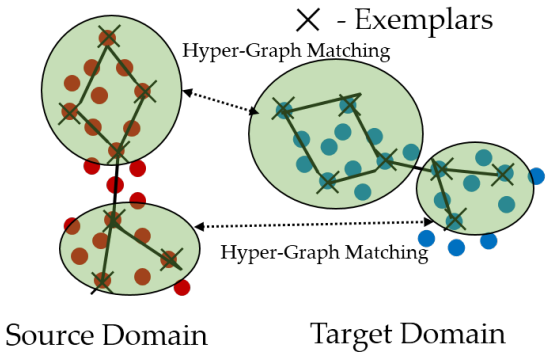


Fig. 2.10. Our approach involves extracting exemplars from both source and target domain. Hyper-graphs are constructed from those exemplars followed by matching.

2.4.1 Finding Exemplars

In our proposed method, we initially perform clustering to extract the set of exemplars, which is a representative subset of the original source and target domain dataset. This clustering is required to increase the computational efficiency of our hyper-graph matching method. We use Affinity Propagation (AP) [176] to extract the exemplars. AP is an efficient clustering algorithm that uses *message passing* between

data-points. The algorithm requires the similarity matrix $\mathbf{S}$ of a dataset as the input. Here, $[\mathbf{S}]_{ij} = -||\mathbf{x}_i - \mathbf{x}_j||^2, i \neq j$ and $[\mathbf{S}]_{ii}$ is the preference of sample $\mathbf{x}_i$ to be an exemplar and is set to the same value of p for all instances. p controls the number of exemplars obtained. For low values of p we obtain less exemplars while for large values of p we obtain more exemplars. To obtain a desired number of exemplars, a bisection method that adjusts p iteratively is used. For our purpose, we set η to be the desired fraction of exemplars for both the source and the target domain dataset. Thus, as an output of the AP algorithm, we would obtain the exemplar source and target domain matrix $\mathbf{X}'^s \in \mathbb{R}^{n'_s \times d}$ and $\mathbf{X}'^t \in \mathbb{R}^{n'_t \times d}$, respectively, where $n'_s \approx \eta n_s$, $n'_t \approx \eta n_t$. However, to keep up with the notation, from now onwards in this approach we would denote $\mathbf{X}^s$, $\mathbf{X}^t$ as the exemplar source and target datasets and n_s and n_t as the number of source and target exemplars.

2.4.2 Hyper-Graph Matching

To carry out hyper-graph matching between the source and target exemplars, we consider all first-, second- and third-order matching between the source and target exemplars. We do not use just the higher order matching because the source and target hyper-graphs will be far from isomorphic and using only the structural information will produce misleading results. On the contrary for registration between similarly shaped objects [158], using only higher order matching produces excellent results.

We seek to find a matching matrix $\mathbf{C} \in \mathbb{R}^{n_s \times n_t}$, where $[\mathbf{C}]_{ij}$ is the measure of correspondence between source exemplar i and target exemplar j. For the first-order matching, we would like exemplars in the source domain to be close to similar exemplars in the target domain. Since the operation $\mathbf{C}\mathbf{X}^t$ rearranges the target exemplars, we would like the re-arranged target exemplars $\mathbf{C}\mathbf{X}^t$ to be close to $\mathbf{X}^s$. Thus, to enable first-order matching, we would like to minimize the normalized term $f_1(\mathbf{C}) = ||\mathbf{C}\mathbf{X}^t - \mathbf{X}^s||_{\mathcal{F}}^2/(n_s d)$, where $||\cdot||_{\mathcal{F}}$ is the Frobenius norm.

For the second-order matching, we would like source exemplar pairs to match with similar target exemplar pairs. This is carried out by initially constructing a source and a target adjacency matrix with source and target exemplars as graph nodes. If $\mathbf{D}^s$ and $\mathbf{D}^t$ are source and target adjacency matrices, then $[\mathbf{D}^s]_{ij} = \exp(-\frac{||\mathbf{x}_i^s - \mathbf{x}_j^s||_2^2}{\sigma_s^2})$, $[\mathbf{D}^t]_{ij} = \exp(-\frac{||\mathbf{x}_i^t - \mathbf{x}_j^t||_2^2}{\sigma_t^2})$ for $i \neq j$ and $[\mathbf{D}^s]_{ii} = [\mathbf{D}^t]_{ii} = 0$. σ_s and σ_t can be found heuristically as the mean sample-to-sample pairwise distance in the source and target domains, respectively. For the second-order similarity, we want the row re-arranged target domain adjacency matrix $\mathbf{C}\mathbf{D}^t$ to be close to the column-rearranged source domain adjacency matrix $\mathbf{D}^s\mathbf{C}$. Taking into consideration the difference in the numbers of source and target exemplars n_s and n_t, second-order graph matching implies minimizing the term $f_2(\mathbf{C}) = ||\mathbf{C}\mathbf{D}^t - r\mathbf{D}^s\mathbf{C}||_{\mathcal{F}}^2$, where $r = \frac{n_t}{n_s}$ is a correction factor.

For the third-order matching problem, we use the tensor based cost term [158]. In that paper, they try to maximize the cost term $f_3(\mathbf{C}) = \mathbf{H} \otimes_1 \mathbf{c} \otimes_2 \mathbf{c} \otimes_3 \mathbf{c}$, where $\mathbf{H} \in \mathbb{R}^{n_s n_t \times n_s n_t \times n_s n_t}$ is a third-order tensor and the index k in $\otimes_k$ indicates tensor multiplication on the k^{th} dimension and $\mathbf{c} \in \mathbb{R}^{n_s n_t}$ is the vectorized matching matrix $\mathbf{C}$. Here, $[\mathbf{H}]_{ijk} = \exp(-\gamma||f_{i_s,j_s,k_s} - f_{i_t,j_t,k_t}||^2)$. If $\mathbf{c}_i$ is the matching variable for the samples $\mathbf{x}_i^s$ and $\mathbf{x}_i^t$, $\mathbf{c}_j$ for the samples $\mathbf{x}_j^s$ and $\mathbf{x}_j^t$ and $\mathbf{c}_k$ for the samples $\mathbf{x}_k^s$ and $\mathbf{x}_k^t$, then f_{i_s,j_s,k_s} is the feature consisting of the sine of the angles of the triangle formed by the data points $\mathbf{x}_i^s, \mathbf{x}_j^s$ and $\mathbf{x}_k^s$ and f_{i_t,j_t,k_t} consisting of the sine of the angles of the triangle formed by the data points $\mathbf{x}_i^t, \mathbf{x}_j^t$ and $\mathbf{x}_k^t$. γ is calculated from the mean pairwise squared distance between the features.

In addition to the graph-matching terms, we add a class-based regularization. The group-lasso regularizer [177] $\ell_2 - \ell_1$ norm term is equal to $f_g(\mathbf{C}) = \sum_j \sum_c ||[\mathbf{C}]_{\mathcal{I}_c j}||_2$, where $||\cdot||_2$ is the ℓ_2 norm and $\mathcal{I}_c$ contains the indices of rows of $\mathbf{C}$ corresponding to the source-domain samples of class c. In other words, $[\mathbf{C}]_{\mathcal{I}_c j}$ is a vector consisting of elements $[\mathbf{C}]_{ij}$, where i^{th} source sample belongs to class c and the j^{th} sample is in the target domain. Minimizing this group-lasso term ensures that a target-domain sample only corresponds to the source-domain samples of the same class.

In the case of $n_t = n_s$, we have one-to-one matching between each source sample and each target sample. However, for the case $n_t \neq n_s$, we must allow multiple correspondences. Accordingly, if the constraint $\mathbf{C}\mathbf{1}_{n_t} = \mathbf{1}_{n_s}$ ($\mathbf{1}_n$ is a $n \times 1$ vector of ones) implies that the sum of the correspondences of all the target samples to each source sample is one, then the second equality constraint $\mathbf{C}^T\mathbf{1}_{n_s} = \frac{n_s}{n_t}\mathbf{1}_{n_t}$ implies that the sum of correspondences of all the source samples to each target sample should increase proportionately by $\frac{n_s}{n_t}$ to allow for the multiple correspondences. Hence, the overall optimization problem becomes

Problem UDA-2

$$\begin{aligned}\min_{\mathbf{C}} f(\mathbf{C}) \;&=\; \frac{1}{(n_s d)}||\mathbf{C}\mathbf{X}^t - \mathbf{X}^s||_{\mathcal{F}}^2 + \lambda_2||\mathbf{C}\mathbf{D}^t - r\mathbf{D}^s\mathbf{C}||_{\mathcal{F}}^2 \\ &\quad - \lambda_3\mathbf{H} \otimes_1 \mathbf{c} \otimes_2 \mathbf{c} \otimes_3 \mathbf{c} + \lambda_g \sum_j \sum_c |||[\mathbf{C}]_{\mathcal{I}_c j}||_2 \qquad (2.4) \\ \text{such that } \mathbf{C} \geq \mathbf{0},\; &\mathbf{C}\mathbf{1}_{n_t} = \mathbf{1}_{n_s},\; \text{and } \mathbf{C}^T\mathbf{1}_{n_s} = (\frac{n_s}{n_t})\mathbf{1}_{n_t}.\end{aligned}$$

2.4.3 Problem Solution

Problem UDA-2 can be solved quickly using second-order methods but has memory and computational issues related to storing the Hessian ($O(n_s^2 n_t^2)$). Hence, first-order methods, specially conditional gradient method (CG) [161] can be used to solve the problem. The CG method maintains the desirable structure of the solution such as sparsity required of $\mathbf{C}$ by solving the successive linear minimization sub-problems over the constraint set [162]. The linear programming (LP) subproblem required to be solved is minimizing $\mathbf{Tr}(\mathbf{G}^T\mathbf{C})$,$\mathbf{C} \geq \mathbf{0}, \mathbf{C}\mathbf{1}_{n_t} = \mathbf{1}_{n_s}, \mathbf{C}^T\mathbf{1}_{n_s} = \frac{n_s}{n_t}\mathbf{1}_{n_t}$, where $\mathbf{G}$ is

the gradient of the function f in *Problem UDA-2* and $\mathbf{Tr}(\cdot)$ is the trace operation. The gradient of each cost term can be derived as

$$\begin{aligned}
\nabla f_1(\mathbf{C}) &= 2(\mathbf{C}\mathbf{X}^t - \mathbf{X}^s)\mathbf{X}^{tT}/(n_s d), \\
\nabla f_2(\mathbf{C}) &= \ 2(\mathbf{C}\mathbf{D}^t\mathbf{D}^{tT} - r\mathbf{D}^s\mathbf{C}\mathbf{D}^{tT} - r\mathbf{D}^{sT}\mathbf{C}\mathbf{D}^t + r^2\mathbf{D}^{sT}\mathbf{D}^s\mathbf{C}) \ , \\
\nabla f_3(\mathbf{C}) &= ([\mathbf{H} \otimes_1 \mathbf{c} \otimes_2 \mathbf{c} + \mathbf{H} \otimes_1 \mathbf{c} \otimes_3 \mathbf{c} + \mathbf{H} \otimes_2 \mathbf{c} \otimes_3 \mathbf{c}]) \\
\frac{\partial f_g}{\partial [\mathbf{C}]_{ij}} &= \frac{[\mathbf{C}]_{ij}}{|||[\mathbf{C}]_{\mathcal{I}_c(i)j}||_2} \cdot \delta(|||[\mathbf{C}]_{\mathcal{I}_c(i)j}||_2 \neq 0)
\end{aligned}$$

Here $[\cdot]$ operator on tensor term reshapes a vector into matrix of size $n_s \times n_t$. $c(i)$ is the class of the i^{th} source sample. $\delta(\cdot)$ is an indicator function which is 1 if the argument is true and 0 otherwise. Thus, after obtaining the gradient $\mathbf{G} = \nabla f(C) = \nabla f_1(C) + \lambda_2 \nabla f_2(C) - \lambda_3 \nabla f_3(C) + \lambda_g \nabla f_g(C)$, we solve the linear minimization problem LP mentioned before using the consensus form of ADMM [151]. We let $\mathbf{a} = \mathbf{1}_{n_s}$ and $\mathbf{b} = \frac{n_s}{n_t}\mathbf{1}_{n_t}$. Using the consensus ADMM form, we reformulate LP as

$$\begin{aligned}
&\min \{g_1(\mathbf{C}_1) + g_2(\mathbf{C}_2) + g_3(\mathbf{C}_3)\} \\
&\text{such that} \quad \mathbf{Z} = \mathbf{C}_1 = \mathbf{C}_2 = \mathbf{C}_3
\end{aligned}$$

where $g_1(\mathbf{C}_1) = 0.5\mathbf{Tr}(\mathbf{G}^T\mathbf{C}_1) + I(\mathbf{C}_1\mathbf{1}_{n_t} = \mathbf{a})$, $g_2(\mathbf{C}_2) = 0.5\mathbf{Tr}(\mathbf{G}^T\mathbf{C}_2) + I(\mathbf{C}_2^T\mathbf{1}_{n_s} = \mathbf{b})$ and $g_3(\mathbf{C}_3) = I(\mathbf{C}_3 \geq \mathbf{0})$. Here, $I(\cdot)$ is an indicator function which is 0 if argument is true and ∞ otherwise. $\mathbf{Z}$ is an intermediate variable to facilitate consensus ADMM. Accordingly, the ADMM updates will be as follows

$$\begin{aligned}
\mathbf{C}_1^{k+1} &= \underset{\mathbf{C}_1\mathbf{1}_{n_t}=\mathbf{a}}{\operatorname{argmin}}(\mathbf{Tr}((0.5\mathbf{G} + \mathbf{Y}_1^k)^T\mathbf{C}_1) + \frac{\rho}{2}||\mathbf{C}_1 - \mathbf{Z}^k||_F^2) \\
\mathbf{C}_2^{k+1} &= \underset{\mathbf{C}_2^T\mathbf{1}_{n_s}=\mathbf{b}}{\operatorname{argmin}}(\mathbf{Tr}((0.5\mathbf{G} + \mathbf{Y}_2^k)^T\mathbf{C}_2) + \frac{\rho}{2}||\mathbf{C}_2 - \mathbf{Z}^k||_F^2) \\
\mathbf{C}_3^{k+1} &= \Pi_{\{\mathbf{C}|\mathbf{C}\geq\mathbf{0}\}}(\mathbf{Z}^k - \mathbf{Y}_3^k), \quad \mathbf{Z}^{k+1} = \frac{1}{3}(\sum_i \mathbf{C}_i^{k+1}) \\
\mathbf{Y}_i^{k+1} &= \mathbf{Y}_i^k + \mathbf{C}_i^{k+1} - \mathbf{Z}^{k+1} \quad \forall i = 1, 2, 3
\end{aligned}$$

Here $\mathbf{Y}_i$'s are dual variables. Π is the projection operator. The penalty parameter is set $\rho = 1$ without loss of generality since scaling ρ is equivalent to scaling $\mathbf{G}$. Updates for $\mathbf{C}_1^{k+1}$ and $\mathbf{C}_2^{k+1}$ are solved using Lagrange multiplier to obtain

$$\mathbf{C}_1^{k+1} = \mathbf{Z}^k - \frac{\mathbf{G}}{2} - \mathbf{Y}_1^k - \frac{1}{n_t}((\mathbf{Z}^k - \frac{\mathbf{G}}{2} - \mathbf{Y}_1^k)\mathbf{1}_{n_t} - \mathbf{a})\mathbf{1}_{n_t}^T,$$
$$\mathbf{C}_2^{k+1} = \mathbf{Z}^k - \frac{\mathbf{G}}{2} - \mathbf{Y}_2^k - \frac{1}{n_s}\mathbf{1}_{n_s}(\mathbf{1}_{n_s}^T(\mathbf{Z}^k - \frac{\mathbf{G}}{2} - \mathbf{Y}_2^k) - \mathbf{b}^T),$$
$$\mathbf{C}_3^{k+1} = \max(\mathbf{Z}^k - \mathbf{Y}_3^k, \mathbf{0})$$

The ADMM updates are repeated for a fixed few-hundred iterations and the optimum value of LP is set to final value of $\mathbf{Z}$. This would complete one iteration of CG. After completing several such iterations of CG, we can obtain the optimal value of $\mathbf{C}^*$. Using that, we can map the source domain data close to the target domain data using regression with $\mathbf{X}^s$ and $\mathbf{C}^*\mathbf{X}^t$ as the input and output data respectively. The whole procedure can be repeated for a number of times at the end of which the adapted source data is used to train a classifier to be tested on target domain data. The overall algorithm is given in Algorithm 3.

2.4.4 Time and Space Complexity

The AP clustering algorithm has a time complexity of $(O(n_s^2 + n_t^2))$. The ADMM updates are linear in the number of variables $(O(\eta^2 n_s n_t))$ and they run for a fixed number of iterations. They are faster than interior point methods, which would result in cubic time-complexity in the number of variables. The overall time complexity will be multiplied by N_T. For the space complexity, the AP algorithm requires storing source and target similarity matrices of $O(n_s^2 + n_t^2)$. Graph adjacency matrices also require space complexity of $O(\eta^2 n_s^2 + \eta^2 n_t^2)$. For tensor storage, we use the sparse strategy [158] to obtain $O(m)$ space complexity, where m is the number of non-zero entries and $m \propto \max(\eta n_s, \eta n_t)$

Algorithm 3: UDA with Hyper-graph matching.

Given : Source Labeled Data $\mathbf{X}^s$ and $\mathbf{y}^s$, and Target Unlabeled Data $\mathbf{X}^t$

Parameters : $\eta, \lambda_2, \lambda_3, \lambda_g, N_T$

Initialize : $t_o = 0$

Repeat

$\mathbf{X}^s, \mathbf{X}^t \leftarrow \mathbf{AP}(\mathbf{X}^s, \mathbf{X}^t, \eta)$, (Affinity Propagation)

Initialize: $\mathbf{C}_0 \in \mathcal{D}$, $t_i = 1$

Repeat (Conditional Gradient)

$\mathbf{G} = \nabla_C f(\mathbf{C}_0)$, $\mathbf{C}_d \leftarrow \text{ADMM}(\mathbf{G}, \mathbf{a}.\mathbf{b})$

$\mathbf{C}_1 = \mathbf{C}_0 + \alpha(\mathbf{C}_d - \mathbf{C}_0), \quad \text{for} \quad \alpha = \frac{2}{t_i+2}$

$\mathbf{C}_0 = \mathbf{C}_1 \quad \text{and} \quad t_i = t_i + 1$

Until Fixed Number of Iterations

$\mathbf{C}^* = \mathbf{C}_0$ and Regress $\mathbf{M}(\cdot) \quad \text{s.t.} \quad \mathbf{X}^s \xrightarrow{\mathbf{M}} \mathbf{C}^*\mathbf{X}^t$

Map $\mathbf{X}^s \leftarrow \mathbf{M}(\mathbf{X}^s) \quad \text{and} \quad t_o = t_o + 1$

Until $t_o = N_T$

Output : Adapted Source Data $\mathbf{X}^s$

2.4.5 Experimental Results

Our proposed method is tested against a standard dataset, known as the **Office-Caltech** dataset, for domain adaptation of the object recognition task. It consists of a subset of the **Office** dataset [167]. This contains three different domains (Amazon, DSLR, Webcam) of images. The Amazon images are from the Amazon site, the DSLR images are captured with a high-resolution DSLR camera and the Webcam domain contains images captured with a low-resolution webcam. The fourth domain consists of a subset of the object recognition dataset **Caltech-256** [169]. These four domains have ten common classes (Bike, BackPack, Calculator, Headphone, Keyboard, Laptop, Monitor, Mouse, Mug, Projector). Accordingly, we can obtain 12 DA task pairs.

The image features are the normalized SURF [170] obtained as a 800-bin histogram. The classifier used was a 1-Nearest neighbor, as it is a hyper-parameter free. For our experiments, we considered a random selection of 20 samples per class (with the exception of 8 samples per class for the DSLR domain) for the source domain. One half of the target-domain data is used for domain adaptation. The accuracy is reported over 10 trials of the experiment. For our experiments, we used $\lambda_g = 0.01$. We reported the best results obtained over the range $\lambda_2, \lambda_3 \in \{10^{-3}, 10^{-2}...10^{0}\}$ with a maximum $N_T = 5$. For the transformation, we used a linear mapping with a ℓ_2 regularization coefficient of 10^{-3}.

We compared our approach against popular non-deep domain adaptation methods. Current methods involve deep architectures and would not compare fairly. So we include (a) The no adaptation baseline (NA); (b) Geodesic Flow Kernel (GFK) [41]; (c) Subspace Alignment (SA) [43]; (d) Joint Distribution Adaptation (JDA) [173], which jointly adapts both marginal and conditional distributions along with dimensionality reduction; (e) Correlation Alignment (CORAL) [44] and (f) Optimal Transport (OT) [49].

The comparison results are shown in Table 2.8. We see that in almost all the cases, local DA methods like OT and our proposed method dominated over other global methods. Moreover, our method dominates over OT for most of the tasks. This is because our method exploits higher order structural similarity between the source and target data points. Also, in situations where OT is slightly better than our methods, the source and target data do not have enough structurally similar regions.

We also performed experiments to see how initial clustering affects recognition performance of our proposed method. We see the results in Table 2.8 for $\eta = 0.75, 0.5$; that is, when the number of exemplars are around $75\%, 50\%$ of the total number of data-points, respectively. In general, the results showed decrease in performance with respect to $\eta = 1$ but are still competitive with respect to some of the previous methods. The only exception is in the tasks $C \rightarrow W$, $A \rightarrow C$ and $W \rightarrow A$, where

Table 2.8. Comparing different methods in terms of classification accuracy (%) of target data. Each task consists of *source* → *target*, where *source* and *target* represent any of the four domains: C(Caltech-256), A(Amazon), W(Webcam) and D(DSLR).

Method	C → A	C → W	C → D	A → C	A → W	A → D	W → C	W → A	W → D	D → C	D → A	D → W
NA	21.86	20.97	22.73	23.85	24.31	21.95	19.03	23.22	50.91	23.79	26.11	51.67
GFK	33.41	33.06	35.19	32.50	33.89	31.45	25.12	31.17	79.22	30.04	32.15	71.87
SA	34.12	30.06	33.11	32.18	32.56	32.98	30.15	34.95	68.31	31.57	34.25	73.40
CORAL	31.76	25.14	27.40	30.23	28.33	30.65	24.92	29.00	78.05	27.53	28.95	74.44
JDA	40.56	34.03	34.28	34.90	34.58	31.82	32.63	34.62	**85.19**	30.50	30.42	78.89
OT	**44.48**	36.02	36.88	**36.11**	37.12	39.35	33.44	37.36	84.02	31.82	36.25	79.23
Ours ($\eta = 1$)	42.30	**38.41**	**37.06**	33.10	**42.01**	**43.35**	**35.14**	**40.71**	83.14	**32.23**	**38.91**	**82.22**
Ours ($\eta = 0.75$)	37.45	30.56	36.66	34.20	41.67	42.86	30.23	33.05	77.97	27.15	33.81	75.87
Ours ($\eta = 0.5$)	34.73	34.81	33.77	33.63	35.42	37.66	29.74	35.24	70.13	24.61	31.42	66.67

for decreasing η, the accuracy increases. This is because decreasing the number of exemplars decreases the possibility of including outliers as unwanted nodes in constructing the graph.

Moreover, we analyzed the effect of regularization parameters $\lambda_2, \lambda_3, \lambda_g$ on the $A \rightarrow W$ task. Fig. 2.11 (a) and (b) showed the matching matrix $\mathbf{C}$ for $\lambda_g = 0.01$ and $\lambda_g = 0$, respectively. The rows and columns in the matrix $\mathbf{C}$ represent the source and target, respectively. In Fig. 2.11(a), for $\lambda_g = 0.01$, we see that 10 block-diagonal matrices representing all the 10 common categories appear as heatmap on the matrix $\mathbf{C}$. It suggests that the group lasso regularizer allows discrimination of the classes more accurately compared to Fig. 2.11(b) that is with $\lambda_g = 0$, where we have matching matrix $\mathbf{C}$ with more uniform entries.

In Fig. 2.11 (c), we show the effect of varying λ_2 and λ_3. The blue straight line shows the accuracy result when $\lambda_2 = \lambda_3 = 0$. The yellow line shows the accuracy result when $\lambda_2 = 0$ and $\log_{10} \lambda_3$ is varied. The red line shows the accuracy result when $\lambda_3 = 0$ and $\log_{10} \lambda_2$ is varied. The results suggest that including the higher order cost terms without weighing them heavily in the cost function improves performance over solely using the first-order matching.

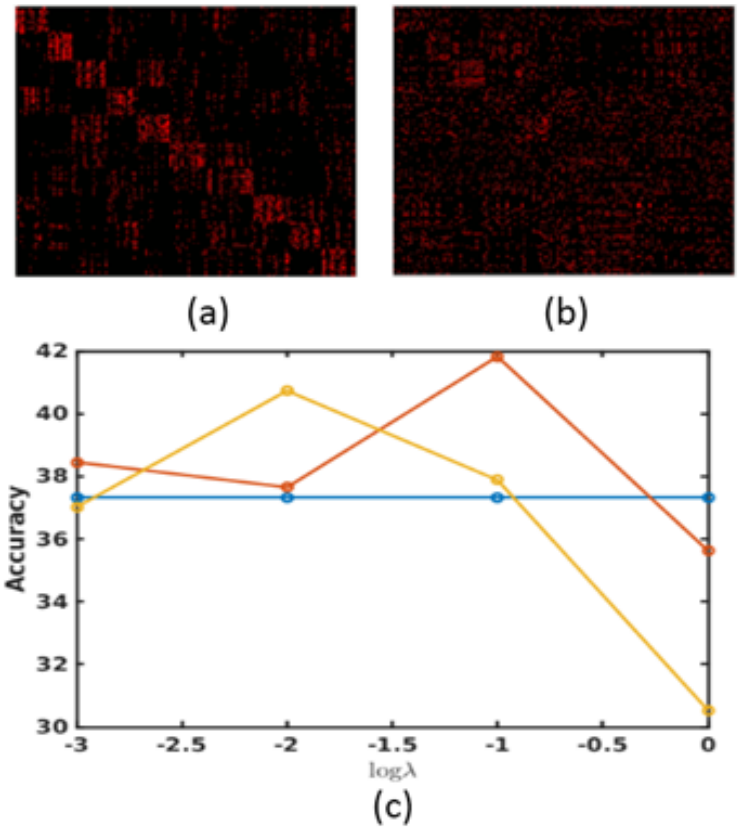


Fig. 2.11. Matching Matrix **C**. (a) With $\lambda_g = 0.01$. (b) With $\lambda_g = 0$. (c) Accuracy values for different λ_2, λ_3. This is for $A \rightarrow W$ task.

2.5 Unsupervised Domain Adaptation with Representation Learning

2.5.1 Problem Definition

We re-define the notation to account for the representations learned. We have n^s labeled samples, $\mathbf{X}^s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n^s}$ from the source domain $\mathcal{D}^s$. We also have n^t unlabeled samples $\mathbf{X}^t = \{\mathbf{x}_i^t\}_{i=1}^{n^t}$ from the target domain $\mathcal{D}^t$. We assume that the domains have the same feature and label space but have different marginal probability distributions; that is, $P(\mathbf{X}^s) \neq P(\mathbf{X}^t)$. The goal is to learn a classifier $\mathbf{K}(\cdot)$ and a representation $\phi(\cdot)$ to minimize the target risk $\epsilon_t = P_{(\mathbf{x},y)\sim D_t}[\mathbf{K}(\phi(\mathbf{x})) \neq y]$.

2.5.2 Minimizing Domain Discrepancy with Graph Matching

Our goal is to learn domain-invariant representations by minimizing a graph matching loss between the source and target representations. In our case, we realize feature extraction using a neural network. We force the feature extractor to learn domain-invariant representations. Given an input sample $\mathbf{x} \in \mathbb{R}^n$ from a domain, the feature extracting network learns a function $\phi : \mathbb{R}^n \to \mathbb{R}^d$ that maps an instance to a d-dimensional feature space. The parameters of the feature extracting network is represented as Θ_F. In order to minimize the discrepancy between the two domains, we minimize the graph matching loss between the domain representations. To encounter excess discrepancy between the source and target domains, we allow an additional affine transformation on the source domain representations. Thus, we have a modified source domain representation $\phi'(\cdot)$ such that $\phi'^T(\mathbf{x}^s) = \phi^T(\mathbf{x}^s)\mathbf{W}_{map} + \mathbf{b}_{map}^T$, where $\mathbf{W}_{map} \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_{map} \in \mathbb{R}^d$ are scaling matrix and bias, respectively. Superscript T indicates the transpose operation. The graph-matching loss considers minimizing a combination of first and second-order matching cost between graphs constructed from the two domains. So, if a mini-batch contains n_b^s, n_b^t source and target samples respectively, we represent the matching between the source and target representations through a matching matrix $\mathbf{C} \in \mathbb{R}^{n_b^s \times n_b^t}$. An element $[\mathbf{C}]_{ij}$ is a measure of matching between mini-batch source sample i and mini-batch target sample j. The source mini-batch features can be stacked to form a matrix $\Phi^s \in \mathbb{R}^{n_b^s \times d}$. Similarly, the target mini-batch features are stacked to form $\Phi^t \in \mathbb{R}^{n_b^t \times d}$. Accordingly, for the first-order matching we want the corresponding target representation to be close to the corresponding mapped source representation. Mathematically, this implies minimizing $||\mathbf{C}\Phi^t - \Phi'^s||_F^2$ where Φ'^s is the modified source domain feature matrix after affine transformation on Φ^s and $||\cdot||_F$ is the Frobenius norm. For the second-order matching, we try to minimize the discrepancy between the adjacency matrix of graphs constructed using the source and target mini-batches. Mathematically, this implies minimizing $||\mathbf{C}\mathbf{D}^t - r\mathbf{D}^s\mathbf{C}||_F^2$, where $\mathbf{D}^t \in \mathbb{R}^{n_b^t \times n_b^t}$ and $\mathbf{D}^s \in \mathbb{R}^{n_b^s \times n_b^s}$ are adjacency matrices con-

structed from Φ^t and Φ^s, respectively. We use the dot product for the similarity measure of the adjacency matrices and consequently $\mathbf{D}^t = \Phi^t \Phi^{tT}$ and $\mathbf{D}^s = \Phi^s \Phi^{sT}$, with diagonals set to 0. $r = \frac{n_b^t}{n_b^s}$ is a correction factor to account for the difference in the size of the source and target mini-batches. In addition, the constraints on $\mathbf{C}$ are as follows: $\mathbf{C} \geq \mathbf{0}$, $\mathbf{C}\mathbf{1}_{n_b^t} = \mathbf{1}_{n_b^s}$ and $\mathbf{C}^T\mathbf{1}_{n_b^s} = (\frac{n_b^s}{n_b^t})\mathbf{1}_{n_b^t}$. The equality constraint $\mathbf{C}\mathbf{1}_{n_b^t} = \mathbf{1}_{n_b^s}$ implies that the sum of the correspondences of all target samples to each source sample is one. The second equality constraint $\mathbf{C}^T\mathbf{1}_{n_b^s} = (\frac{n_b^s}{n_b^t})\mathbf{1}_{n_b^t}$ implies that the sum of correspondences of all source samples to each target sample should increase proportionately by $\frac{n_b^s}{n_b^t}$ to allow for multiple correspondences. Accordingly, the optimization problem becomes

$$\min_{\mathbf{C},\mathbf{W}_{map},\mathbf{b}_{map}} \mathcal{L}_{0GM} = \frac{1}{(n_s d)}||\mathbf{C}\Phi^t - \Phi'^s||_F^2 + \lambda_s||\mathbf{C}\mathbf{D}^t - r\mathbf{D}^s\mathbf{C}||_F^2$$
$$s.t. \quad \mathbf{C} \geq \mathbf{0}, \quad \mathbf{C}\mathbf{1}_{n_b^t} = \mathbf{1}_{n_b^s}, \quad \mathbf{C}^T\mathbf{1}_{n_b^s} = (\frac{1}{r})\mathbf{1}_{n_b^t} \tag{2.5}$$

In the context of training neural networks, the above optimization problem can be solved using the projected gradient descent, where each iterate is projected onto the constraint set. Training neural networks generally requires a lot of time and further projection might increase the time complexity. As a result, we propose to reformulate the equality constraints as penalties in addition to the cost function. Thus our optimization problem becomes

$$\min_{\mathbf{C},\mathbf{W}_{map},\mathbf{b}_{map}} \mathcal{L}_{GM} = \frac{1}{(n_s d)}||\mathbf{C}\Phi^t - \Phi'^s||_F^2 + \lambda_s||\mathbf{C}\mathbf{D}^t - r\mathbf{D}^s\mathbf{C}||_F^2$$
$$+\lambda_p(||\mathbf{C}\mathbf{1}_{n_b^t} - \mathbf{1}_{n_b^s}||_2^2 + ||\mathbf{C}^T\mathbf{1}_{n_b^s} - (\frac{1}{r})\mathbf{1}_{n_b^t}||_2^2) \quad s.t. \quad \mathbf{C} \geq \mathbf{0}, \tag{2.6}$$

where λ_p weighs the penalty terms. As a result, we can carry out gradient descent on $\mathcal{L}_{GM}$ and project it onto the set of positive matrices after each iteration.

In addition, we can exploit the labels of the source domain data to build a classifier on top of the feature extractor. We add the classifier on top of the feature extraction network. Since the graph-matching loss produces domain-invariant representations, the classifier can be applied on the target domain. The objective of the classifier $\mathbf{K}(\cdot) : \mathbb{R}^d \to \mathbb{R}^l$ is to compute softmax prediction for the l classes. Let the parameters

of the classifier be denoted as Θ_K. The classifier loss function is the cross-entropy between the predictions and the ground-truth class labels:

$$\mathcal{L}_c(\mathbf{x}^s, y^s) = -\frac{1}{n_b^s} \sum_{i=1}^{n_b^s} \sum_{k=1}^{l} 1(y_i^s = k) \log(\mathbf{K}(\phi'(\mathbf{x}_i^s))_k) \tag{2.7}$$

where $1(y_i^s = k)$ is a 0-1 indicator function and $\mathbf{K}(\phi'(\mathbf{x}_i^s))_k$ corresponds to the k^{th} dimension value of the softmax output. Thus, the classification loss is combined with the graph matching loss to obtain the following objective function

$$\min_{\Theta_F, \Theta_K} \{\mathcal{L}_c + \lambda \min_{\mathbf{C} \geq \mathbf{0}, \mathbf{W}_{map}, \mathbf{b}_{map}} [\mathcal{L}_{GM}]\} \tag{2.8}$$

where λ is the coefficient controlling the balance between classification and graph matching loss. Note that the minimization is carried out using mini-batch gradient descent. As described in Algorithm 4, using a mini-batch containing labeled source data and unlabeled target data, $\mathcal{L}_{GM}$ is optimized with respect to $\mathbf{C}$ and after that iteratively projecting onto positive matrices. After the optimized matching matrix $\mathbf{C}^*$ is obtained, we solve for $\mathbf{W}_{map}, \mathbf{b}_{map}$, for which a closed form solution exists. The solution for $\mathbf{W}_{map}$, $\mathbf{b}_{map}$ can be obtained as follows:

$$\begin{bmatrix} \mathbf{W}_{map} \\ \mathbf{b}_{map}^T \end{bmatrix} = \left[\frac{1}{n_b^s d} \begin{bmatrix} \Phi^{sT} \\ \mathbf{1}^T \end{bmatrix} \begin{bmatrix} \Phi^s & \mathbf{1} \end{bmatrix} + \frac{\lambda_w}{d^2} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \right]^{-1} \left[\frac{1}{n_b^s d} \begin{bmatrix} \Phi^{sT} \\ \mathbf{1}^T \end{bmatrix} \mathbf{C}^* \Phi^t + \frac{\lambda_w}{d^2} \begin{bmatrix} \mathbf{I} \\ \mathbf{0}^T \end{bmatrix} \right]. \tag{2.9}$$

Here λ_w regularizer is introduced to allow for a smooth mapping transformation. Subsequently, we optimize for the total loss as in Eq. (2.8) with respect to the parameters of the feature extracting network and the classifier. The learned representations are domain invariant as well as target discriminative since the feature extractor parameter Θ_F receives gradients from both the graph matching and classification loss. The overall framework of our method is given in Fig. 2.12(a). The detailed algorithm of the training procedure is illustrated in Algorithm 4.

Fig. 2.12. The overall neural network framework for training using (a) Graph Matching (GM) Loss and (b) Pseudo-label (PL) Loss. On the right of (a) and (b), we see the model we should use for inference.

2.5.3 Refinement with Pseudo-labels

This is the second stage of our proposed unsupervised domain adaptation approach. Till now, we have the mapped source domain representations in support of

the target domain representations. Since the domain discrepancy has been minimized, we can think of all the source and target representations belonging to a single domain. This single domain consists of labeled and unlabeled data. This is a semi-supervised learning setting that has been explored from a low-density separation, manifold regularization point of view [178]. In this method, we propose a novel approach to exploit the confident unlabeled target domain data to further refine the classification decision boundary.

Algorithm 4: Graph-Matching-Guided Deep Domain Adaptation.

Given : Source Labeled Data $\mathbf{X}^s$, $\mathbf{Y}^s$, Target Unlabeled Data $\mathbf{X}^t$

Parameters : $\lambda_s, \lambda_p, \lambda, m, T_i$ and learning rates

Randomly Initialize $\Theta_F, \Theta_K, \mathbf{C}, \mathbf{W}_{map}, \mathbf{b}_{map}$

Repeat

Sample mini-batch $\{\mathbf{x}_i^s, y_i^s\}_{i=1}^m$, $\{\mathbf{x}_i^t\}_{i=1}^m$ from $\mathbf{X}^s$ and $\mathbf{X}^t$

Use mini-batch to form Φ^t and Φ^s

for $t_i = 1, 2, ...T_i$

$\mathbf{C} \leftarrow \mathbf{C} - \alpha_1 \nabla_{\mathbf{C}} \mathcal{L}_{GM}(\Phi^t, \Phi^s)$

$\mathbf{C} \leftarrow \max(\mathbf{C}, \mathbf{0})$

end for

Use Eq. (2.9) to obtain $\mathbf{W}_{map}$, $\mathbf{b}_{map}$

$\Theta_K \leftarrow \Theta_K - \alpha_2 \nabla_{\Theta_K} \mathcal{L}_c(\mathbf{x}^s, y^s)$

$\Theta_F \leftarrow \Theta_F - \alpha_3 \nabla_{\Theta_F} [\mathcal{L}_c(\mathbf{x}^s, y^s) + \lambda \mathcal{L}_{GM}(\Phi^t, \Phi^s)]$

Until Convergence

Initially, we select a subset of a mini-batch of the unlabeled target data that provide highly-confident labels as output. In other words, we select those samples whose maximum softmax probability output is greater than a threshold (t_h). Mathematically, we select those $\mathbf{x}_i^t$ for which $\max\{\mathbf{K}(\phi(\mathbf{x}_i^t))_k\} \geq t_h$ over all classes $k \in \{1, 2, ...l\}$ and we repeat this for all unlabeled target domain samples in the mini-batch $i \in \{1, 2, ...m\}$. The pseudo-labels for those selected samples would be

$\underset{k}{\text{argmax}}\{\mathbf{K}(\phi(\mathbf{x}_i^t))_k\}$. After that, we use the original labeled data $\{\mathbf{x}_i^s, y_i^s\}_{i=1}^m$ and the selected unlabeled samples as $\{\mathbf{x}_i^t\}_{i=1}^{m'}$, where $m' \leq m$ to further refine our model. The intuition for our method is that we want the unlabeled samples to be as far as possible from the decision boundaries. This would make it possible for unseen examples in the target domain to not be misclassified easily. As a result, we expect performance in the target domain to increase significantly.

In our model, we have a softmax classifier that returns probabilities of each class that the sample belongs to. Also pairwise relations between the probabilities give a measure of how far a sample is from a decision boundary between the corresponding pair of classes. For example, if the softmax classifier returns $(p_1, p_2, ...p_l)$ as outputs to input sample $\mathbf{x}$, $|p_i - p_j|$ is a measure of how far the sample $\mathbf{x}$ is from the decision boundary between class i and class j. If $p_i = p_j$, then the sample lies on the decision boundary between class i and class j. The general expression for maximizing the distance to the decision boundaries for all selected unlabeled samples and all classes is as follows:

$$\mathcal{L}_p(\mathbf{x}^t, \hat{y}^s) = \frac{1}{m'} \sum_{i=1}^{m'} \sum_{j,k} 1(\hat{y}_i^t = j \ \ OR \ \ \hat{y}_i^t = k)(p_j - p_k)^2. \tag{2.10}$$

Here, $p_j = \mathbf{K}(\phi(\mathbf{x}_i^t))_j$, and $\hat{y}_i^t$ is the pseudo-label corresponding to the input sample $\mathbf{x}_i^t$ as obtained using thresholding. When $\hat{y}_i^t = j$ or $\hat{y}_i^t = k$ is true, we have $1(\hat{y}_i^t = j \ \ OR \ \ \hat{y}_i^t = k) = 1$, and 0 otherwise. We call $\mathcal{L}_p$ as the *Pseudo-Label* (PL) loss. We also use the classification loss $\mathcal{L}_c$ introduced in Eq. (2.7) to regularize $\mathcal{L}_p$. Hence, we need to solve the following optimization problem,

$$\min_{\Theta_F, \Theta_K} \{-\mathcal{L}_p + \gamma \mathcal{L}_c\}, \tag{2.11}$$

where γ weighs the classification cost term. In Fig. 2.12(b), we show the overall neural network framework for using the Pseudo-Label (PL) loss. Algorithm 5 outlines the detailed approach of the training procedure.

Algorithm 5: Pseudo-label-guided Deep Domain Adaptation.

Given : Source Labeled Data $\mathbf{X}^s$, $\mathbf{Y}^s$, Target Unlabeled Data $\mathbf{X}^t$

Parameters : γ, t_h, m and learning rates

Restart $\Theta_F, \Theta_K, \mathbf{W}_{map}, \mathbf{b}_{map}$ obtained from Algorithm 4

Repeat

Sample mini-batch $\{\mathbf{x}_i^s, y_i^s\}_{i=1}^m$, $\{\mathbf{x}_i^t\}_{i=1}^m$ from $\mathbf{X}^s$ and $\mathbf{X}^t$

Obtain high-confidence samples and pseudo-labels $\{\mathbf{x}_i^t, \hat{y}_i^t\}_{i=1}^{m'}$ using t_h criterion and use those samples for parameter update as follows

$\Theta_K \leftarrow \Theta_K - \alpha_2 \nabla_{\Theta_K}[-\mathcal{L}_p(\mathbf{x}^t, \hat{y}^t) + \gamma \mathcal{L}_c(\mathbf{x}^s, y^s)]$

$\Theta_F \leftarrow \Theta_F - \alpha_3 \nabla_{\Theta_F}[-\mathcal{L}_p(\mathbf{x}^t, \hat{y}^t) + \gamma \mathcal{L}_c(\mathbf{x}^s, y^s)]$

Until Convergence

2.5.4 Experiments and Results

To evaluate the effectiveness of our proposed approach on standard domain adaptation datasets for image classification,we utilized the Office-Caltech dataset, a small-scale domain adaptation benchmark dataset, initially released by [41]. The dataset is composed of 10 common categories across 4 domains - Amazon (A), Webcam (W), DSLR (D) and Caltech (C). Each of these domains varies in terms of image quality, viewpoints, presence/absence of backgrounds, etc. For domain adaptation, we would have 12 tasks, where each task consists of a source domain and a target domain picked from the 4 domains. For our experiments, we use *Decaf* features as the input. These deep features [171] are 4096-dimensional FC7 hidden activations of the deep convolutional neural network AlexNet [12].

We compared our method to recent approaches in learning domain-invariant representations. As a lower bound on recognition accuracy, we also compare against the no-adaptation (NA) baseline which includes training the model using only the source data and directly testing on the target data. The methods that we compared against include: (a) DANN [56], (b) MMD [35], (c) CORAL [54] and (d) WDGRL [57].

We have implemented our approach in Tensorflow and the training was carried out using *Adam* [179] optimizer. We followed the standard protocol used in previous method as in [57]. Since hyper-parameter selection is not possible using deep unsupervised domain-adaptation methods, we reported the best results of each approach after carrying out grid search on their respective hyper-parameters. For training, the batch consisted of 32 samples from each domain. The feature extractor is a 2-layer neural network with 500 and 100 nodes and a ReLU activation. We used this same feature extractor in all the methods for fair comparisons. For our method, we used the following values of the penalty parameter $\lambda_p = 10$, threshold $t_h = 0.8$, and mapping regularization $\lambda_w = 0.1$. We set λ_s, λ and γ as the tunable hyper-parameters over which we reported the best results averaged over 10 trials in Table 2.9.

Table 2.9. Domain-adaptation results for object recognition using Office-Caltech datasets using Decaf features for a pair of source → target domain.

Task	NA	MMD	DANN	CORAL	WDGRL	GM	GM+PL
A→C	83.93	86.72	87.12	86.24	87.84	87.99	**89.48**
A→D	82.23	89.96	83.27	90.36	91.67	92.82	**95.73**
A→W	76.69	90.68	80.13	89.61	89.34	92.63	**94.23**
W→A	80.23	89.34	81.36	83.42	92.34	90.64	**94.68**
W→D	96.49	**100**	**100**	**100**	**100**	**100**	**100**
W→C	78.65	88.64	80.11	86.27	89.42	88.78	**91.31**
D→A	82.91	90.24	84.72	84.1	91.34	89.24	**92.34**
D→W	96.86	97.68	98.34	96.93	97.24	97.84	**99.83**
D→C	78.61	86.58	83.69	80.49	**90.24**	85.68	88.87
C→A	89.97	91.6	90.84	92.49	93.57	93.68	**95.83**
C→W	86.47	90.36	88.74	91.62	91.23	92.68	**94.21**
C→D	87.79	90.64	89.41	88.71	92.68	92.83	**94.51**

From Table 2.9, we see that in almost all cases our proposed graph-matching method (GM) is close to the previous best method. However, with the additional

pseudo-labeling stage (PL), our proposed method produces better recognition accuracy in almost all the domain-adaptation tasks. Also, in almost all cases, the improvement of GM+PL over GM is 2-3%. This justifies the exploitation of labeled and unlabeled data after minimizing domain discrepancy, leading to an improvement in performance. For the task D→C, GM and eventually GM+PL do not produce the best result. This is possibly because the datasets D and C do not have enough structurally similar regions to be matched appropriately.

We chose a particular task A→W and studied the effect of varying hyper-parameters on recognition performance. In Fig. 2.13(a), we see that the performance reaches a peak at $\lambda_s = 10$. The red-dotted line is the base-line performance for $\lambda_s = 0$. So, the presence of the second-order matching term increases the performance over when it is not. Also, for $\lambda_s = 100$, the performance dips by a large amount, suggesting that putting excess weight on second-order term is not recommended. We saw a similar trend for the hyper-parameter λ in Fig. 2.13(b). λ weighs the graph-matching loss with respect to the classification loss. As expected, putting too much weight ($\lambda = 10$) ignores the classification loss in domain adaptation and produces a dip in performance. Recognition performance is comparatively less sensitive to γ as seen in Fig. 2.13(c). This is because domain discrepancy has already been minimized and the presence of classification loss on the source data does not affect target domain recognition rate much. Fig. 2.13(d) shows the convergence of source and target error. We used GM stage for the first 2000 iterations followed by the PL stage in the next 2000 iterations. We noticed the drop in error rate when the PL stage was introduced after 2000 iterations. We also visualized the learned features using t-SNE [174] in Fig 2.14. The clusters in the figure correspond to 10 classes. The blue and red points correspond to the source and target data respectively. For the unadapted data in Fig. 2.14(a), the target domain classes do not form compact clusters. Also, there is a lot of discrepancy between the corresponding source and target clusters, causing a lot of mis-classification. For UDA, using only the GM procedure as in Fig. 2.14(b), the target domain classes form clusters but there are still some divergence between

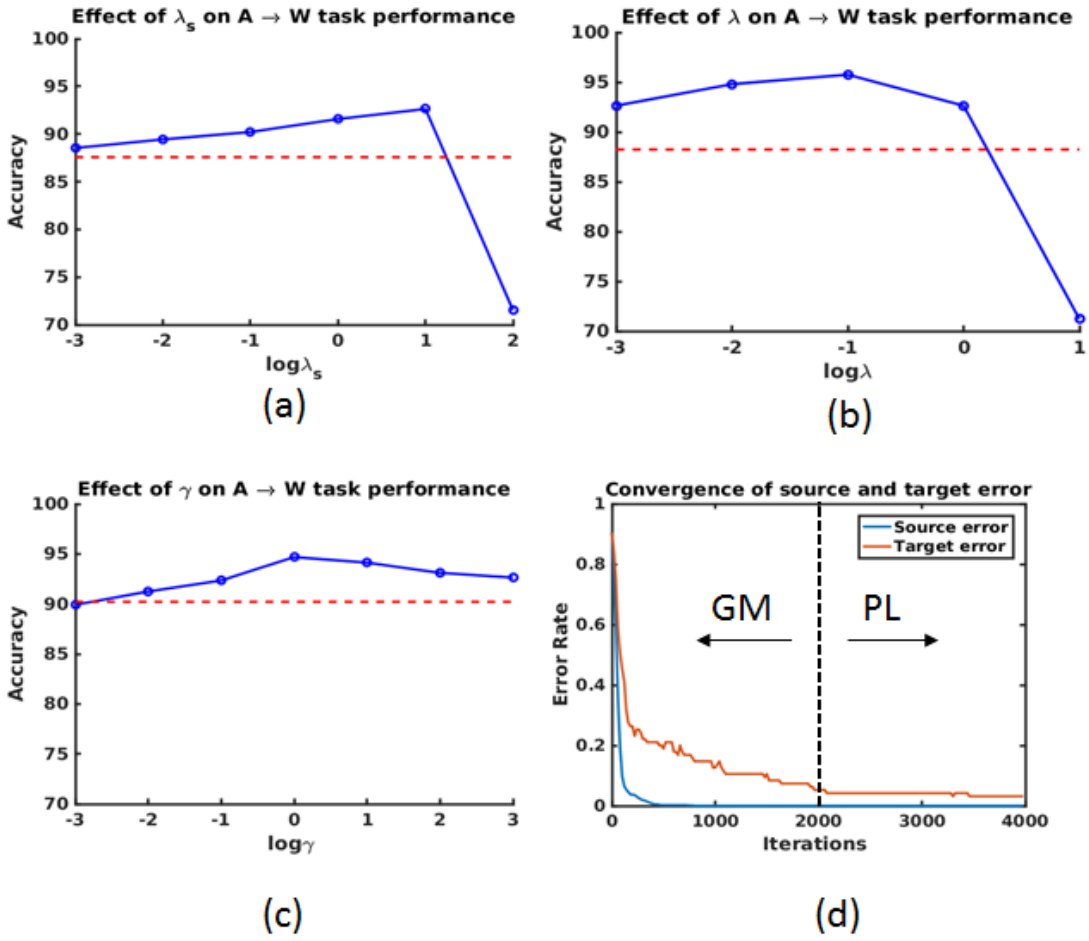


Fig. 2.13. Accuracy results on the A→W task due to change in (a) λ_s, (b) λ, (c) γ and (d) convergence results.

some of the corresponding source and target classes, which are reduced further using the PL stage as shown in Fig. 2.14(c).

2.6 Conclusions

In this Chapter, we proposed three methods for unsupervised domain adaptation. In the first method, we found correspondences between the source and target domain samples. The correspondences are found out by carrying out first and second order matching between the graphs constructed from the source and target domain

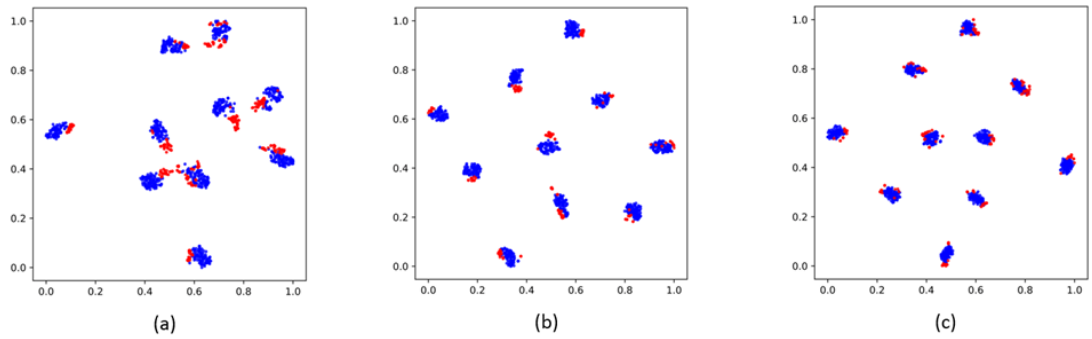


Fig. 2.14. Feature visualization for the A→W task for (a) no adaptation, (b) UDA with only Graph Matching and (c) UDA with Graph Matching and Pseudo-labeling.

samples. The second method is an extension of the first method in the sense that additional third-order hyper-graph matching is used. Also, to increase the computational efficiency, only a subset of the total number of samples are used for constructing the hyper-graph. The optimization subroutine used is ADMM instead of a network simplex one. In the third method, we proposed a two-stage approach to learning domain-invariant-feature representations for unsupervised domain adaptation. In the first stage, we considered minimizing graph matching (GM) loss to minimize the discrepancy between source and target domains. The graph matching loss includes a second-order structural similarity term that allows us to consider structural similarity between two domains. For the second stage, we refined the feature/classifier using the confident pseudo-labels (PL) of the target domain data. Experimental results suggest that each of these methods are competitive with respect to the state-of-the-art. Overall, the first method was found to have the fastest run-time but the worst recognition performance. On the other-hand, the third method had the slowest run-time but the best recognition performance. This is because this method had to learn a representation with a lot of parameters and it also had two-stage learning.

3. A TWO-STAGE APPROACH TO FEW-SHOT LEARNING FOR IMAGE RECOGNITION

3.1 Introduction

In this Chapter, we tackle the few-shot learning (FSL) problem, which is a sub-problem of transfer learning that assumes different sets of categories in the source and target domains. The source domain contains abundant labeled data while the target domain contains only a few-labeled data per category. The few-shot learning setting is depicted in Fig. 3.1 from a feature space perspective. Our goal is to learn a classification model for the novel categories. This problem setting is more difficult compared to the unsupervised domain adaptation setting described in the Chapter two. This is because the categories are different between the source domain and the target domain and hence the discrepancy between the domain distributions is larger. Besides that, the data distribution of the novel categories is not well defined because we only have access to a few labeled data from each of the novel categories. As a result, a classifier learned on the source-domain data will severely overfit on the target-domain data. This limitation motivates us to estimate the statistical properties of the novel classes in the form of means and variances. These means and the variances need to be estimated from the few-shot samples of the novel categories. This estimation function can be learned from the abundant labeled source-domain data and can therefore serve as prior knowledge for target-domain classification. Since the estimation function is expected to be complex, we choose neural network as a structural prior to learn the mapping from samples to statistics (mean and variance) . In addition we propose a new low-dimensional but discriminative feature space to prevent overfitting due to the curse of dimensionality. The results of this work have been published in [153].

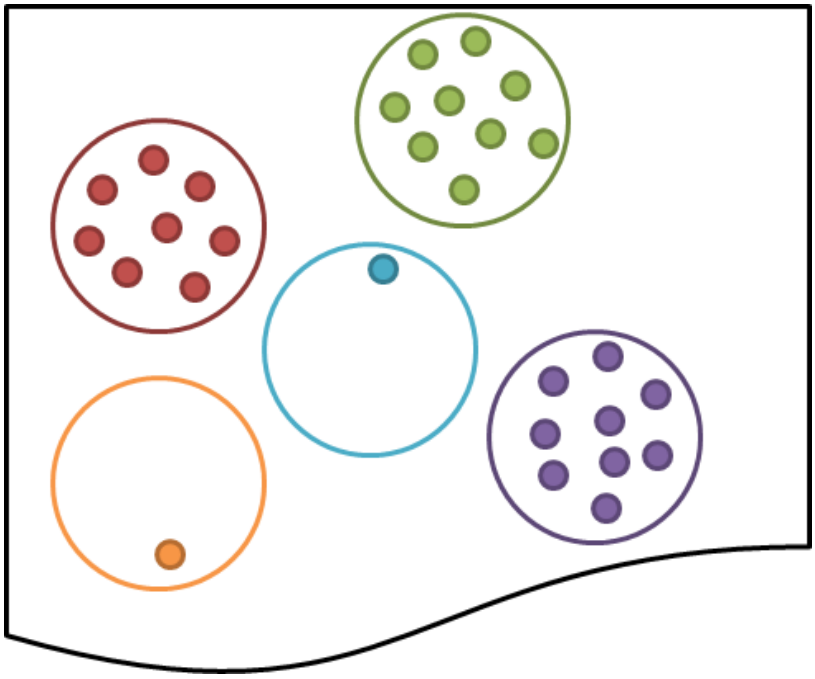


Fig. 3.1. In the few-shot learning setting, the base classes (red, green, violet) contain lots of labeled data while the novel classes (orange, blue) contain few-labeled data.

Previous methods closely related to our proposed approach include meta-learning and metric learning methods. Meta-learning methods use a learning-to-learn scheme where an auxiliary meta-model is used to predict or optimize the model hyper-parameters like the optimizer [91], the parameter initializations [93], etc. This idea of using a meta-model to guide the original model has been depicted in Fig. 3.2. On the other hand, metric-learning methods [86, 87, 89] learn a representation, where samples of the same class are constrained to be close to one another while samples of different classes are constrained to be kept far from each other. This concept is shown in Fig. 3.2. Our proposed approach consists of using both meta-learning and metric-learning components to improve recognition performance. Our proposed neural-network module that estimates novel-class mean and variance serves as the meta-model that learns to classify a given test sample as an input. Similarly, we

learn a new low-dimensional metric space that uses pair-wise distances between samples as features to produce better class discriminability.

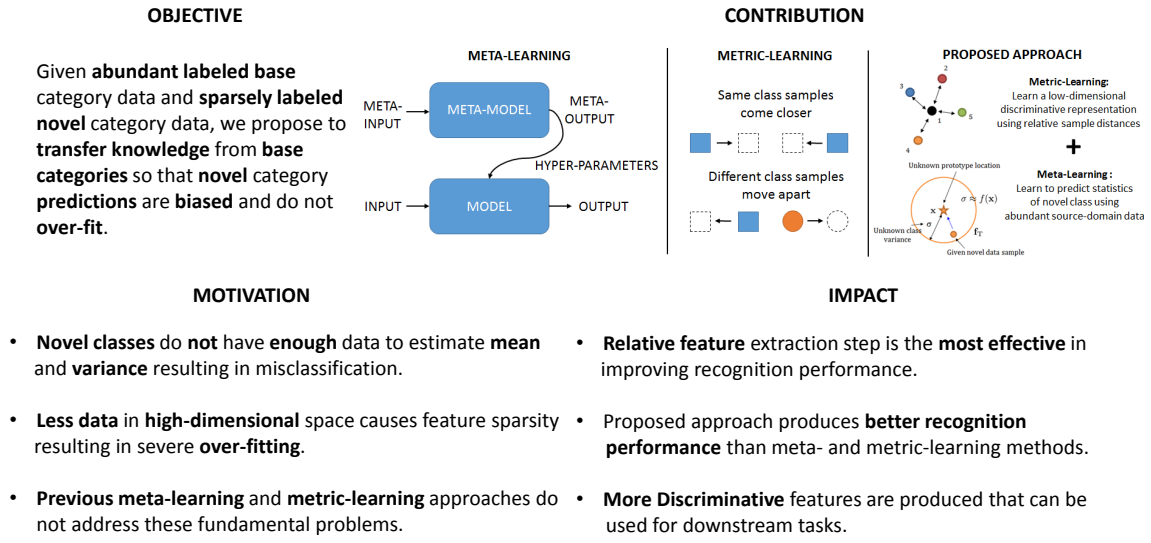


Fig. 3.2. Overall outline of our proposed framework along with motivations and impacts.

To be more specific, we propose a multi-layer neural network structure for few-shot image recognition of novel categories. The proposed multi-layer neural network architecture encodes transferable knowledge extracted from a large annotated dataset of base categories. This architecture is then applied to novel categories containing only a few samples. The transfer of knowledge is carried out at the feature-extraction and the classification levels distributed across the two training stages. In the first-training stage, we introduce the relative feature to capture the structure of the data as well as obtain a low-dimensional discriminative space. Secondly, we account for the variable variance of different categories by using a network to predict the variance of each class. Classification is then performed by computing the Mahalanobis distance to the mean-class representation in contrast to previous approaches that used the Euclidean distance. In the second-training stage, a category-agnostic mapping is

learned from the mean-sample representation to its corresponding class-prototype representation. This is because the mean-sample representation may not accurately represent the novel category prototype. Finally, we evaluate the proposed network structure on four standard few-shot image recognition datasets, where our proposed few-shot learning system produces competitive performance compared to previous work. We also extensively studied and analyzed the contribution of each component of our proposed framework.

3.2 Proposed Approach

3.2.1 Problem Definition and Formulation

Our proposed few-shot learning method has both metric-learning and meta-learning components, which are learned in two stages. The metric-learning stage learns both absolute and relative feature sets and then uses the Mahalanobis distance metric to compute class labels of the test sample. The idea of using relative features stemmed from our prior work in domain adaption [149, 150, 152]. Domain adaptation considers adaptation between labeled source-domain data and unlabeled target-domain data but with the same categories in both domains. The meta-learning stage learns auxiliary knowledge for classification, which is a transformation from a sample to its corresponding class prototype. This idea is related to the work of Wang et al. [101], where they learned to transform small-sample-model parameters to large-sample-model parameters. The work on few-shot learning without forgetting [102] also used a category-agnostic transformation but with a different distance metric and without any procedure to avoid negative transfer. The overall framework of our proposed few-shot learning approach is shown in Fig. 3.3.

Our proposed few-shot learning image recognition system is trained using a large database of N_{base} base categories, which consists of a large number of samples from each category. Each of these categories contains a large amount of data that we can use to learn some useful generalizable knowledge. This knowledge should help

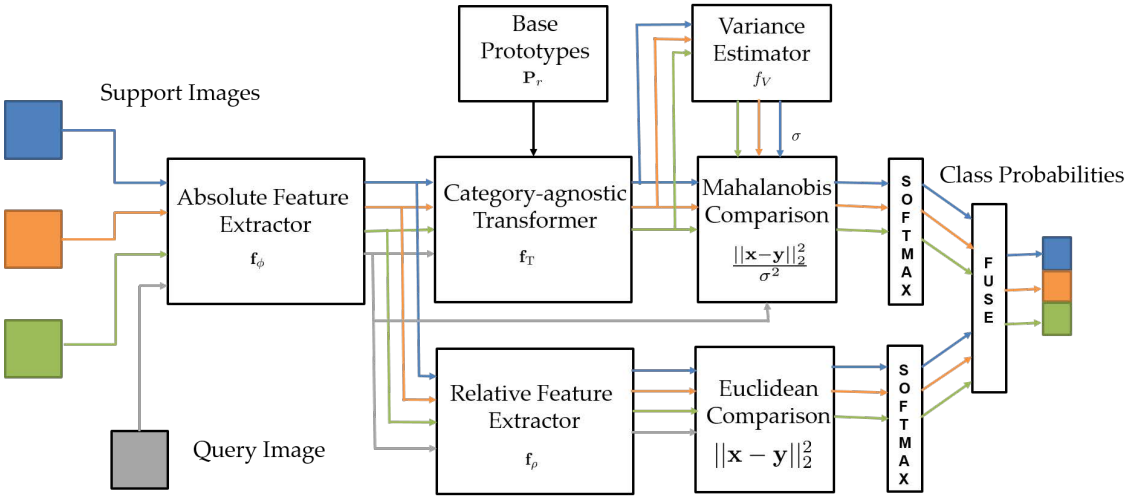


Fig. 3.3. Overall framework for the proposed approach for a 3-way 1-shot inference scenario. A single image from each of the 3 classes (classes are shown in different colors) are used as support examples while a single query image is used. The output is the probability of the query example belonging to each of the 3 classes.

the recognition of N_{novel} novel categories for which only a few labeled samples per category are available.

The knowledge can be learned using traditional supervised learning, where training is generally carried out by feeding instances from the base categories in the form of mini-batches and then optimizing some loss function. The model is generally tested on the same set of categories on which it is trained. If we want the trained model to work on novel categories, then the model can be fine-tuned on the new training dataset [180]. However, the procedure of fine-tuning might not work if the novel categories have very few samples in each category. In fact, the fine-tuning procedure might cause the model to overfit on the few training samples, causing it to underperform on novel category test samples. The main reason for overfitting is that the number of training samples per category is much less compared to the dimensionality of the feature space and therefore the variance of the few samples is inaccurate to capture the distribution of the class.

We address these shortcomings of high dimensionality and variable variance by proposing the use of relative features, variance estimator and category-agnostic trans-

formation. Still, the traditional training procedure involving mini-batches from a large dataset would not be able to produce a satisfactory model since it does not simulate the test condition well. Each test category contains only a few samples and extracting mini-batches for training is impossible. Hence, an *episodic* training strategy inspired from [86] needs to be deployed.

In episodic learning, the set of few labeled samples available from each of the novel categories is known as the *support set*. The set of unobserved testing samples of the novel categories is often called the *query set*. If the support set were large, we could have just trained the model on the support set. However, since the support set is small, traditional training of a model would result in over-fitting and consequently the model would produce unsatisfactory performance on the testing data. However, the episodic training strategy can avert poor performance by simulating the test conditions. In each training episode, we first select N classes randomly from among the N_{base} base categories. From each of those selected N classes, we randomly select K and Q disjoint samples from it. This sampling strategy is called the N-way K-shot sampling strategy. In general, K is same as the number of support samples present per novel category. Q is user-specified and is generally set in the range of 5 to 15 per category. Using this N-way K-shot sampling strategy, we form the training support set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_s}$, where $n_s = K \times N$, and also the training query set $\mathcal{Q} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{n_q}$, where $n_q = Q \times N$. In the training episode, the support set is used to represent the class while the query set is used for the evaluation.

3.2.2 Relative-Feature-Space Representation

The first step of our proposed few-shot learning framework requires feature extraction from the raw samples. This is done by feeding the support set samples $\mathbf{x}_i$ from $\mathcal{S}$ and the query set samples $\mathbf{x}_j$ from $\mathcal{Q}$ through the feature extraction module $\mathbf{f}_\phi$ to produce the embeddings $\mathbf{f}_\phi(\mathbf{x}_i)$ and $\mathbf{f}_\phi(\mathbf{x}_j)$, respectively. The dimensionality of this absolute feature map $\mathbf{f}_\phi$ is very large compared to the total number of support

and query samples. This sparsity in the number of samples compared to the dimension volume generally leads to over-fitting and poor generalization performance. To address this dimensionality problem, we propose the relative-feature-space representation, which has a dimensionality comparable to the total number of support and query samples in an episode. The dimensionality of this relative feature space will therefore be much less than the original absolute feature space.

The relative feature of a sample in an episode is computed by calculating the squared pairwise Euclidean distance with itself and to all other samples in the episode. Hence, if there are $r = n_s + n_q$ samples in an episode, counting all n_s support and n_q query samples regardless of the categories, then the d^{th} dimension of the relative feature $\mathbf{f}_\rho$ of a sample $\mathbf{x}_k$ is given as

$$[\mathbf{f}_\rho(\mathbf{x}_k)]_d = ||\mathbf{f}_\phi(\mathbf{x}_k) - \mathbf{f}_\phi(\mathbf{x}_d)||_2^2, \tag{3.1}$$

where $k, d \in \{1, 2, ..., r\}$ and $||\cdot||_2$ is the Euclidean norm. Note that $[\mathbf{f}_\rho(\mathbf{x}_k)]_d = 0$ for $k = d$. The dimensionality of this relative feature map is therefore r. Since this relative feature-space dimensionality is comparable to the number of samples and that these features contain important structural information about the data, we expect that the inclusion of this feature would increase few-shot testing performance.

In Fig. 3.4, we show a simple example on how to compute the relative-feature representation from the absolute-feature representation. Consider that there are three image samples – $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ in an episode whose absolute-feature representations are $\mathbf{p}_1 = \mathbf{f}_\phi(\mathbf{x}_1)$, $\mathbf{p}_2 = \mathbf{f}_\phi(\mathbf{x}_2)$, and $\mathbf{p}_3 = \mathbf{f}_\phi(\mathbf{x}_3)$, respectively. They are pairwise separated through Euclidean distances of 1, 2 and 3 as shown in the figure. From Eq. (3.1), the relative-feature representation is obtained by squaring the pairwise Euclidean distances. Since there are three points in the episode, these points will lie in a three-dimensional relative-feature representation space and they would be represented as $\mathbf{p}'_1 = \mathbf{f}_\rho(\mathbf{x}_1) = [0, 9, 1]^T$, $\mathbf{p}'_2 = \mathbf{f}_\rho(\mathbf{x}_2) = [9, 0, 4]^T$ and $\mathbf{p}'_3 = \mathbf{f}_\rho(\mathbf{x}_3) = [1, 4, 0]^T$.

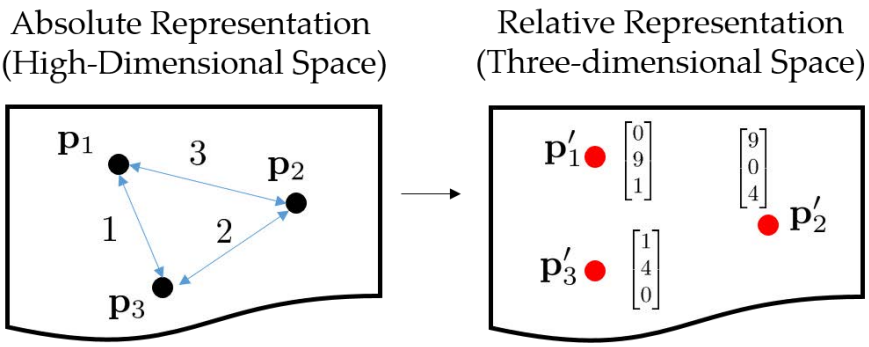


Fig. 3.4. This figure shows an example on how the low-dimensional relative-feature representation is computed from the original high-dimensional representation space. The original high dimensional feature space contains three data points. Accordingly, we would obtain a three-dimensional feature space if we compute pairwise distances of a data-point with itself and other points.

3.2.3 Variance Estimation

After embedding the support and query points in the absolute-feature space ($\mathbf{f}_\phi$) and the relative-feature space ($\mathbf{f}_\rho$), our goal is to use these features for classification. We do not want to tie our model to any category. We want to make our model generalizable to novel categories and therefore we do not use a classification layer that is commonly used for traditional neural networks. Instead, a nearest-class-mean approach is used [88], where the query point embeddings are compared to the prototype representation of each class. The prototypes of a class $\mathbf{p}_{r\phi_c}$ and $\mathbf{p}_{r\rho_c}$ can be found by averaging the embedded support points of its class for both the absolute and relative representations, respectively, as follows

$$\mathbf{p}_{r\phi_c} = \frac{1}{|\mathcal{S}_c|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_c} \mathbf{f}_\phi(\mathbf{x}_i), \tag{3.2}$$

$$\mathbf{p}_{r\rho_c} = \frac{1}{|\mathcal{S}_c|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_c} \mathbf{f}_\rho(\mathbf{x}_i), \tag{3.3}$$

where $\mathcal{S}_c$ is the set of samples of the support set $\mathcal{S}$, which belongs to class c. Using these prototypes, we can proceed to calculate the probability distribution over classes $p_\phi(y = c|\mathbf{x})$ and $p_\rho(y = c|\mathbf{x})$ for a query point $\mathbf{x}$. This is done using the

softmax operation with distance metrics $d_\phi(\cdot)$ and $d_\rho(\cdot)$ for the absolute and relative representations, respectively, as follows

$$p_\phi(y = c|\mathbf{x}) = \frac{\exp(-d_\phi(\mathbf{f}_\phi(\mathbf{x}), \mathbf{p}_{r\phi_c}))}{\sum_{c'} \exp(-d_\phi(\mathbf{f}_\phi(\mathbf{x}), \mathbf{p}_{r\phi_{c'}}))}, \tag{3.4}$$

$$p_\rho(y = c|\mathbf{x}) = \frac{\exp(-d_\rho(\mathbf{f}_\rho(\mathbf{x}), \mathbf{p}_{r\rho_c}))}{\sum_{c'} \exp(-d_\rho(\mathbf{f}_\rho(\mathbf{x}), \mathbf{p}_{r\rho_{c'}}))}, \tag{3.5}$$

where the summation $\sum_{c'}$ is over all the classes present in the episode. In Eqs. (3.4) and (3.5), the distance metrics $d_\phi(\cdot)$ and $d_\rho(\cdot)$ need to be defined in order to compute the probability distributions. Snell et. al [87] compared cosine and Euclidean distances and found Euclidean distance to perform better for few-shot testing. They argued that the Euclidean-distance metric is an example of Bregman Divergence. As a result, prototype computation and inference is considered mixture density estimation with exponential family distributions. However, if the Euclidean distance is used, we assume that all the classes have the same spread in the embedding space. This assumption may lead to poor classification performance because all the classes may not have the same variance. Thus, we propose to use the Mahalanobis distance to measure and include the spread of each class in the classification scheme.

The Mahalanobis metric measures the distance between a data point $\mathbf{x}$ and a distribution $\mathcal{D}$. If the distribution $\mathcal{D}$ has an associated mean μ and an invertible covariance matrix $\mathbf{S}$, then the Mahalanobis distance d_M is calculated as

$$d_M = \sqrt{(\mathbf{x} - \mu)^T \mathbf{S}^{-1} (\mathbf{x} - \mu)}, \tag{3.6}$$

where $\mathbf{S}^{-1}$ is the inverse of the covariance matrix $\mathbf{S}$. In case the distribution is spherically Gaussian with a variance σ^2 for all the feature dimensions, the Mahalanobis distance d_M is reduced to

$$\begin{aligned} d_M &= \sqrt{(\mathbf{x} - \mu)^T \mathbf{S}^{-1} (\mathbf{x} - \mu)} = \sqrt{(\mathbf{x} - \mu)^T (\sigma^2 \mathbf{I})^{-1} (\mathbf{x} - \mu)} \\ &= \sqrt{\frac{(\mathbf{x} - \mu)^T (\mathbf{x} - \mu)}{\sigma^2}} = \sqrt{\frac{||\mathbf{x} - \mu||_2^2}{\sigma^2}} = \frac{||\mathbf{x} - \mu||_2}{\sigma}, \end{aligned} \tag{3.7}$$

where $\mathbf{I}$ is an appropriate identity matrix.

The importance of using the Mahalanobis distance over the Euclidean distance is illustrated in Fig. 3.5 in which we have three classes with prototypes centered at $\mathbf{p}_{r\phi_1}$, $\mathbf{p}_{r\phi_2}$ and $\mathbf{p}_{r\phi_3}$. The spread of the classes is quantified through the standard deviations σ_1, σ_2 and σ_3. The goal is to classify the query points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ into one of the three classes. If we use the Euclidean distances for comparison, point $\mathbf{x}_1$ would yield equal probabilities for classes 1 and 2 since the point is equidistant from those classes. This classification does not take into consideration that the spread of class 1 is more than the spread of class 2; that is, $\sigma_1 > \sigma_2$. If we use the Mahalanobis distance, $\frac{||\mathbf{x}_1 - \mathbf{p}_{r\phi_1}||_2^2}{\sigma_1^2} < \frac{||\mathbf{x}_1 - \mathbf{p}_{r\phi_2}||_2^2}{\sigma_2^2}$, and accordingly the query point $\mathbf{x}_1$ will yield a higher probability for class 1. Similar treatment can also be applied to query points $\mathbf{x}_2$ and $\mathbf{x}_3$.

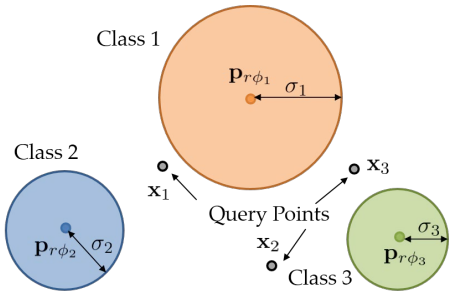


Fig. 3.5. This figure shows an example where different classes can have different variances. As a result, the Mahalanobis distance maybe preferred over Euclidean distance for classifying a test query point into one of these classes.

In our model, we expect each class to have its own covariance matrix $\mathbf{S}$. Therefore, there is a need to model the covariance $\mathbf{S}$ as a function of each class's prototype. However, the covariance matrix $\mathbf{S} \in \mathbb{R}^{D \times D}$ is very high-dimensional, requiring lots of parameters to model it. Furthermore, the covariance matrix $\mathbf{S}$ is required to be positive definite, the constraints of which need to be satisfied strictly. Hence, we settle with using a spherical Gaussian distribution with the same variance for all the feature dimensions. Since we let the class variance be a function of the class's prototype, we can write

$$\sigma_c^2 = f_V(\mathbf{p}_{r\phi_c}), \tag{3.8}$$

where σ_c^2 and $\mathbf{p}_{r\phi_c}$ are the variance and prototype of class c, respectively. This concept of predictable variance may be difficult to grasp initially. However, one can think of it as curve fitting of a function, where the input is the prototype and the output is the variance of the corresponding prototype. The corresponding function is fit using lots of data available from the base categories. Since we expect the function to be smooth, prototypes closer to each other should produce similar variances. After training is over, this function can then be used to predict the variance of novel-class prototypes. The variance estimating function f_V can therefore be implemented by a neural network. Hence, using Eqs. (3.7) and (3.8), the distance metric $d_\phi(\cdot)$ in Eq. (3.4) can be expressed as the square of the Mahalanobis distance as follows

$$d_\phi(\mathbf{f}_\phi(\mathbf{x}), \mathbf{p}_{r\phi_c}) = \frac{||\mathbf{f}_\phi(\mathbf{x}) - \mathbf{p}_{r\phi_c}||_2^2}{\sigma_c^2} = \frac{||\mathbf{f}_\phi(\mathbf{x}) - \mathbf{p}_{r\phi_c}||_2^2}{f_V(\mathbf{p}_{r\phi_c})}. \tag{3.9}$$

For the relative-feature space, the concept of having a variance does not have any physical meaning. As a result, we just use the square of the Euclidean distance metric for $d_\rho(\cdot)$ such that

$$d_\rho(\mathbf{f}_\rho(\mathbf{x}), \mathbf{p}_{r\rho_c}) = ||\mathbf{f}_\rho(\mathbf{x}) - \mathbf{p}_{r\rho_c}||_2^2. \tag{3.10}$$

The representation is learned by minimizing the negative log-probability averaged over all the query points. The negative log-probability of a query point is given as

$$L(\mathbf{\Phi}, \mathbf{V}) = -\log p_\phi(y = c|\mathbf{x}) - \lambda_\rho \log p_\rho(y = c|\mathbf{x}), \tag{3.11}$$

where $\mathbf{\Phi}$ and $\mathbf{V}$ are composed of all the trainable parameters of the feature extractor ($\mathbf{f}_\phi$) and the variance estimator (f_V), respectively, and λ_ρ is a hyper-parameter for the regularization in Eq. (3.11). The negative log-probability averaged over all the query points in the batch needs to be minimized.

3.2.4 Category-agnostic Transformation

After the feature-extraction model and the variance estimator are trained, we proceed to the next stage of training. In this training stage, we propose to find a

category-agnostic transformation from a mean-sample representation of a class to the prototype representation of the corresponding class. Learning this transformation is important because the novel categories have very few support samples and so the mean-sample representation will not accurately represent the prototype. The existence of this category-agnostic transformation may be questionable. However, previous work by Wang et al. [101] suggested the existence of a similar transformation. In that work, the authors proposed the existence of a transformation between model parameters trained using less number of samples to model parameters trained using large number of samples. Since model parameters and samples are dual of each other, we conjecture the existence of a transformation between the mean-sample representation and the prototypes. We next determine this category-agnostic transformation and the factors that this transformation depends on.

In addition to the mean-sample representation, the location of the novel-class prototype would also depend on the nearby base-class prototypes. This is illustrated through an example in Fig. 3.6 in which we have one support sample point for a novel class. But this support data-point may not always be able to represent a class prototype because it might be present on the edge of the distribution as in this example. The transformation function mapping the support point to the unknown class prototype should depend on the support point as well as on the nearby similar base categories. This is because the neighboring class prototypes condition the possible locations of the novel-class prototype. In this example, base classes 1 and 3 form the neighboring categories on which the location of the novel-class prototype should depend. Base class 2 is far from the novel class in the feature space and therefore it should have little effect on the location of the novel-class prototype. We next describe the construction of the transformation function $\mathbf{f}_{\mathrm{T}}$.

The prototype of a novel category c depends on the mean-sample representation and the base-category prototypes collected in $\mathbf{P}_r$, where $\mathbf{P}_r \in \mathbb{R}^{n_p \times D}$ consists of the n_p base-category prototypes stacked vertically in a matrix, and D is the dimension of the absolute-feature space in which the prototypes lie. Ideally, the prototype matrix

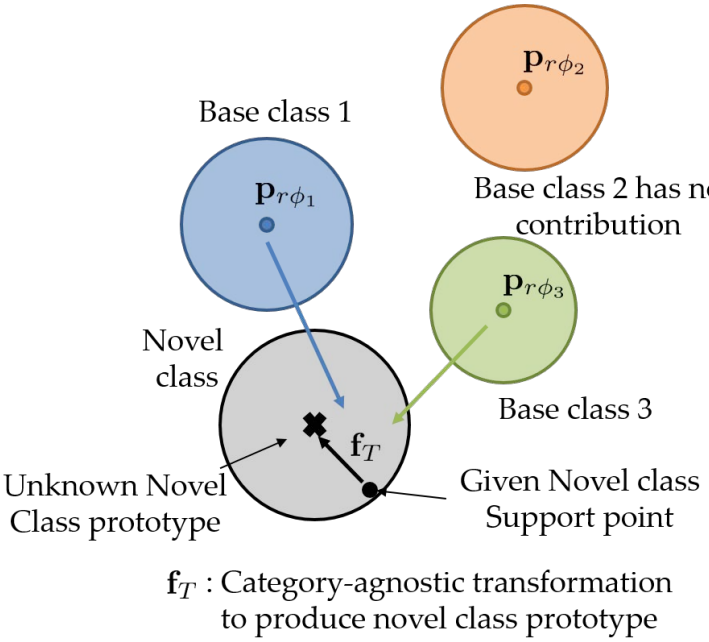


Fig. 3.6. Example depicting the choice of factors affecting the category-agnostic transformation from a support data-point to the corresponding prototype.

should be calculated using the base categories. Since each base category has a large number of samples, the mean representation will be used as an accurate estimate of the prototype. Thus, the prototype $\mathbf{p}'_{r\phi_c}$ of a novel class c can be represented as

$$\mathbf{p}'_{r\phi_c} = \mathbf{f}_{\mathrm{T}}(\mathbf{p}_{r\phi_c}, \mathbf{P}_r), \tag{3.12}$$

where $\mathbf{p}_{r\phi_c}$ is the mean-sample representation of the novel class c. We can decompose the function $\mathbf{f}_{\mathrm{T}}(\mathbf{p}_{r\phi_c}, \mathbf{P}_r)$ into two functions, $\mathbf{f}_{\mathrm{T}}(\mathbf{p}_{r\phi_c}, \mathbf{P}_r) = \mathbf{f}_{\mathrm{T}_1}(\mathbf{p}_{r\phi_c}) + \mathbf{f}_{\mathrm{T}_2}(\mathbf{p}_{r\phi_c}, \mathbf{P}_r)$, where $\mathbf{f}_{\mathrm{T}_1}$ is the contribution due to the mean-sample representation and $\mathbf{f}_{\mathrm{T}_2}$ is the contribution due to the base-class prototypes $\mathbf{P}_r$. Since the contribution of the base-class prototypes depends on the closeness of $\mathbf{p}_{r\phi_c}$ to the prototypes in $\mathbf{P}_r$, $\mathbf{f}_{\mathrm{T}_2}$ will also depend on $\mathbf{p}_{r\phi_c}$. We next discuss the construction of functions $\mathbf{f}_{\mathrm{T}_1}$ and $\mathbf{f}_{\mathrm{T}_2}$.

Contribution of novel-class samples using residual connection. The function $\mathbf{f}_{\mathrm{T}_1}$ is a complex non-linear function that transforms the mean-sample representation $\mathbf{p}_{r\phi_c}$ towards the prototype $\mathbf{p}'_{r\phi_c}$. In case the number of samples in the novel category is large, $\mathbf{p}_{r\phi_c}$ should identically map to $\mathbf{p}'_{r\phi_c}$. Hence, it is important for the

function $\mathbf{f}_{\mathrm{T}_1}$ to model identity mappings. Residual connections and networks have been shown to model identity functions smoothly [23]. In our case, the corresponding meaningful residual connection will be $\mathbf{f}_{\mathrm{T}_1}(\mathbf{p}_{r\phi_c}) = \mathbf{p}_{r\phi_c} + \mathbf{f}_{\mathrm{T}_{11}}(\mathbf{p}_{r\phi_c})$, where $\mathbf{f}_{\mathrm{T}_{11}}(\mathbf{p}_{r\phi_c})$ is a bias term and does not have a scaling effect on the mean-sample representation. Thus, if we include a scaled residual connection, then

$$\mathbf{f}_{\mathrm{T}_1}(\mathbf{p}_{r\phi_c}) = \mathbf{p}_{r\phi_c}\mathbf{W}_1 + \mathbf{f}_{\mathrm{T}_{11}}(\mathbf{p}_{r\phi_c}), \tag{3.13}$$

where $\mathbf{W}_1 \in \mathbb{R}^{D\times D}$ is the scaling matrix. Letting $\mathbf{f}_{\mathrm{T}_{12}}(\mathbf{p}_{r\phi_c}) = \mathbf{p}_{r\phi_c}\mathbf{W}_1$, the bias term $\mathbf{f}_{\mathrm{T}_{11}}(\mathbf{p}_{r\phi_c})$ will be a complex non-linear term and can be modeled using a multi-layer neural network.

Contribution of the base classes. The function $\mathbf{f}_{\mathrm{T}_2}$ models the contribution of base-class prototypes to the novel-class prototype. Base classes that are similar to the novel class will have more contribution. This similarity can be measured in terms of Euclidean distance between a novel class mean-sample representation and a base-class prototype. The contribution of a base class l to a novel class c is quantified through a probability distribution,

$$p_p(c,l) = \frac{\exp(-||[\mathbf{P}_r]_l - \mathbf{p}_{r\phi_c}||_2^2)}{\sum_{l'}\exp(-||[\mathbf{P}_r]_{l'} - \mathbf{p}_{r\phi_c}||_2^2)}, \tag{3.14}$$

where $[\mathbf{P}_r]_l$ is the prototype belonging to the l^{th} base class. The computation of probability is carried out for all the base classes $l = 1, 2, ..., n_p$. These are stacked together to form a probability vector $\mathbf{p}_c$ for the novel class c. After that, we use a threshold t_h on the probability vector $\mathbf{p}_c \in \mathbb{R}^{1\times n_p}$. Only the elements above the threshold t_h are kept while other elements are set to zero. This thresholding step is important as it ignores the effect of base classes that have very little contribution to the novel classes. From the feature-space perspective, novel classes that are distant from the base classes are ignored. This step is our attempt to prevent negative transfer [181], where irrelevant base classes contributing to learning novel-class recognition will reduce the recognition performance. The thresholded probability vector is set as $\mathbf{p}_c^{t_h}$. This is used to combine the base-class prototypes such that

$$\mathbf{f}_{\mathrm{T}_2}(\mathbf{p}_{r\phi_c}, \mathbf{P}_r) = \mathbf{p}_c^{t_h}\mathbf{P}_r\mathbf{W}_2, \tag{3.15}$$

where $\mathbf{W}_2 \in \mathbb{R}^{D\times D}$ is the scaling matrix. The factor $\mathbf{p}_c^{t_h}\mathbf{P}_r$ linearly combines the contributing base-class prototypes. The presence of $\mathbf{W}_2$ is important in scaling the effect of this term to the whole transformation function $\mathbf{f}_{\mathrm{T}}$. Next, we discuss the procedure to learn this category-agnostic transformation $\mathbf{f}_{\mathrm{T}}$, using the large labeled dataset available from the base categories.

Training Strategy. In the second stage of training, we follow the *episodic* training strategy similar to the first stage. In each training episode, we randomly sample N_{pn} categories from among the N_{base} categories. We call these N_{pn} categories as pseudo-novel categories. We refer to the remaining $N_{base} - N_{pn}$ categories as pseudo-base categories. The goal of this training strategy is to simulate the testing scenario where we have novel classes as well as already known base classes.

In a training episode, the prototypes of the pseudo-base categories are calculated using the mean-sample representation. These prototypes can be stacked together to form the prototype matrix $\mathbf{P}_r$. For each pseudo-novel category, we randomly select K_{pn} and Q_{pn} disjoint samples. From this, we form the training support set $\mathcal{S}_{pn} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m_{pn}}$, where $m_{pn} = K_{pn} \times N_{pn}$ and also the training query set $\mathcal{Q}_{pn} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{n_{pn}}$, where $n_{pn} = Q_{pn} \times N_{pn}$. For a category c belonging to one of the N_{pn} categories, we calculate the corresponding class prototype $\mathbf{p}'_{r\phi_c}$ using Eqs. (3.12)-(3.15). Using this modified prototype $\mathbf{p}'_{r\phi_c}$, we can proceed to calculate the class probability distribution for a query point $\mathbf{x}$. This is done using the softmax operation with the Mahalanobis distance metric as described previously

$$p'_\phi(y = c|\mathbf{x}) = \frac{\exp(-d_\phi(\mathbf{f}_\phi(\mathbf{x}), \mathbf{p}'_{r\phi_c}))}{\sum_{c'} \exp(-d_\phi(\mathbf{f}_\phi(\mathbf{x}), \mathbf{p}'_{r\phi_{c'}}))}, \tag{3.16}$$

where the summation $\sum_{c'}$ is over all the N_{pn} pseudo-novel classes present in the episode.

The training is carried out by minimizing the negative log-probability averaged over all the query points. The negative log-probability of a query point is given as $L(\mathbf{\Theta}) = -\log p'_\phi(y = c|\mathbf{x})$, where $\mathbf{\Theta}$ consists of the scaling matrices $\mathbf{W}_1$, $\mathbf{W}_2$ and all the trainable parameters of the residual network $\mathbf{f}_{\mathrm{T}_{11}}$. We also include a

regression-based regularization involving the ground truth and predicted prototypes of these N_{pn} pseudo-novel classes. If the ground truth prototype of class c is $\mathbf{p}^{gt}_{r\phi_c}$ and the predicted prototype is $\mathbf{p}'_{r\phi_c}$, then the corresponding regularization becomes $L_r(\mathbf{\Theta}) = ||\mathbf{p}'_{r\phi_c} - \mathbf{p}^{gt}_{r\phi_c}||_2^2$. This regularization is averaged over all the prototypes of pseudo-novel classes. The regularization coefficient is set as λ_r.

After the training is done, testing is also carried out in an episodic fashion. For each episode, we randomly sample N_{test} classes from the novel test classes. From each novel class, K_{test} support samples and Q_{test} query samples are drawn randomly. The class prediction for a query point $\mathbf{x}$ is given as the class c which minimizes $-\log p'_\phi(y = c|\mathbf{x}) - \lambda_\rho \log p_\rho(y = c|\mathbf{x})$. The overall training procedure of the proposed two-stage few-shot learning method is provided in Algorithm 6.

3.3 Experimental Results

3.3.1 Datasets

To evaluate our proposed few-shot learning approach, we performed experiments on four datasets – the Omniglot [73], the miniImagenet, the CUB-200 [182] and the CIFAR-100 [183] datasets. These datasets provide a large variety of category-level granularity, image resolution and categories to test upon. The Omniglot dataset consists of 1623 handwritten characters taken from 50 alphabets. Each character has 20 examples associated with it. Each example is written by a different person, resulting in sufficient intra-class variation. According to the procedure of Vinyals et al. [86], the images are resized to 28×28. Each character class is augmented with more samples by having rotations in multiples of 90 degrees. So around 1200 character classes (total of 4800 including rotations) are chosen as the training (i.e., base) categories and the remaining classes are chosen as the testing (i.e., novel) categories. The miniImagenet dataset is a subset of the ILSVRC-12 dataset [11]. It consists of RGB color images of size 84×84, consisting of 100 classes with 600 examples in each class. The 100 classes are divided into 64 for training (base), 16 for validation and 20 for testing (novel).

Algorithm 6: Proposed two-stage few-shot learning procedure.

Given: Base category training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where each $y_i \in \{1, 2, \ldots, C\}$. $\mathcal{D}_c$ is a subset of $\mathcal{D}$ containing elements from class c

Parameters: λ_ρ, λ_r

Randomly initialize parameters of feature extraction ($\mathbf{\Phi}$) and variance estimation ($\mathbf{V}$)

for each episode

$\mathcal{N} \leftarrow \text{Sample}(\{1, 2, \ldots, C\}, N)$

for $c \in \{1, 2, \ldots, N\}$

$\mathcal{S}_c \leftarrow \text{Sample}(\mathcal{D}_{\mathcal{N}_c}, K)$, $\mathcal{Q}_c \leftarrow \text{Sample}(\mathcal{D}_{\mathcal{N}_c} \setminus \mathcal{S}_c, Q)$

$\mathbf{p}_{r\phi_c} = \frac{1}{|\mathcal{S}_c|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_c} \mathbf{f}_\phi(\mathbf{x}_i)$, $\mathbf{p}_{r\rho_c} = \frac{1}{|\mathcal{S}_c|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_c} \mathbf{f}_\rho(\mathbf{x}_i)$, $\sigma_c^2 = f_V(\mathbf{p}_{r\phi_c})$

end for

$L_1 \leftarrow 0$

for $c \in \{1, 2, \ldots, N\}$

for $(\mathbf{x}, y) \in \mathcal{Q}_c$

$L_1 \leftarrow L_1 + \frac{1}{NQ}[(d_\phi(\mathbf{f}_\phi(\mathbf{x}), \mathbf{p}_{r\phi_c})) + \log(\sum_{c'} \exp(-d_\phi(\mathbf{f}_\phi(\mathbf{x}), \mathbf{p}_{r\phi_{c'}})))) +$
$\lambda_\rho(d_\rho(\mathbf{f}_\rho(\mathbf{x}), \mathbf{p}_{r\rho_c})) + \log(\sum_{c'} \exp(-d_\rho(\mathbf{f}_\rho(\mathbf{x}), \mathbf{p}_{r\rho_{c'}}))))]$

end for

end for

Take gradient step of L_1 with respect to $\mathbf{\Phi}, \mathbf{V}$

end for

First training stage ends and second training stage starts.

Randomly initialize parameters of category-agnostic transformer ($\mathbf{\Theta}$)

for each episode

$\mathcal{N}_{nov} \leftarrow \text{Sample}(\{1, 2, \ldots, C\}, N_{pn})$

$\mathcal{N}_{base} \leftarrow \{1, 2, \ldots, C\} \setminus \mathcal{N}_{nov}$

$\mathbf{P}_r \leftarrow \mathcal{N}_{base}$ [Form pseudo-base prototypes]

$\mathbf{P}_{rn} \leftarrow \mathcal{N}_{nov}$ [Form pseudo-novel prototypes]

for $c \in \{1, 2, \ldots, N_{pn}\}$

$\mathcal{S}_{pn_c} \leftarrow \text{Sample}(\mathcal{D}_{\mathcal{N}_{nov_c}}, K_{pn})$, $\mathcal{Q}_{pn_c} \leftarrow \text{Sample}(\mathcal{D}_{\mathcal{N}_{nov_c}} \setminus \mathcal{S}_{pn_c}, Q_{pn})$

$\mathbf{p}'_{r\phi_c} = \mathbf{f}_{\text{T}}(\mathbf{p}_{r\phi_c}, \mathbf{P}_r)$

end for

$L_2 \leftarrow 0$

for $c \in \{1, 2, \ldots, N_{pn}\}$

for $(\mathbf{x}, y) \in \mathcal{Q}_{pn_c}$

$L_2 \leftarrow L_2 + \frac{1}{N_{pn} Q_{pn}}[(d_\phi(\mathbf{f}_\phi(\mathbf{x}), \mathbf{p}'_{r\phi_c})) + \log(\sum_{c'} \exp(-d_\phi(\mathbf{f}_\phi(\mathbf{x}), \mathbf{p}'_{r\phi_{c'}})))) +$
$\lambda_r ||\mathbf{p}'_{r\phi_c} - \mathbf{p}^{gt}_{r\phi_c}||_2^2]$, where $\mathbf{p}^{gt}_{r\phi_c} = [\mathbf{P}_{rn}]_c$

end for

end for

Take gradient step of L_2 with respect to $\mathbf{\Theta}$

end for

The CUB-200 and CIFAR-100 datasets have been introduced long before but have only recently been used as a benchmark for few-shot learning algorithms. The CUB-200 dataset is a fine-grained dataset consisting of 11,788 images of size $84 \times 84 \times 3$, distributed across 200 categories of bird species. Using the class splits in [184], we have 100, 50 and 50 categories used for training, validation and testing, respectively. The CIFAR-100 dataset consists of 60000 low-resolution images of size $32 \times 32 \times 3$. These images are distributed across 100 fine-grained categories or 20 coarse-grained categories. Using the class splits in [185], we have 64, 16 and 20 categories used for training, validation and testing, respectively. Figures 3.7(a), (b), (c) and (d) show some of the examples from the Omniglot, the miniImagenet, the CUB-200 and the CIFAR-100 datasets, respectively.

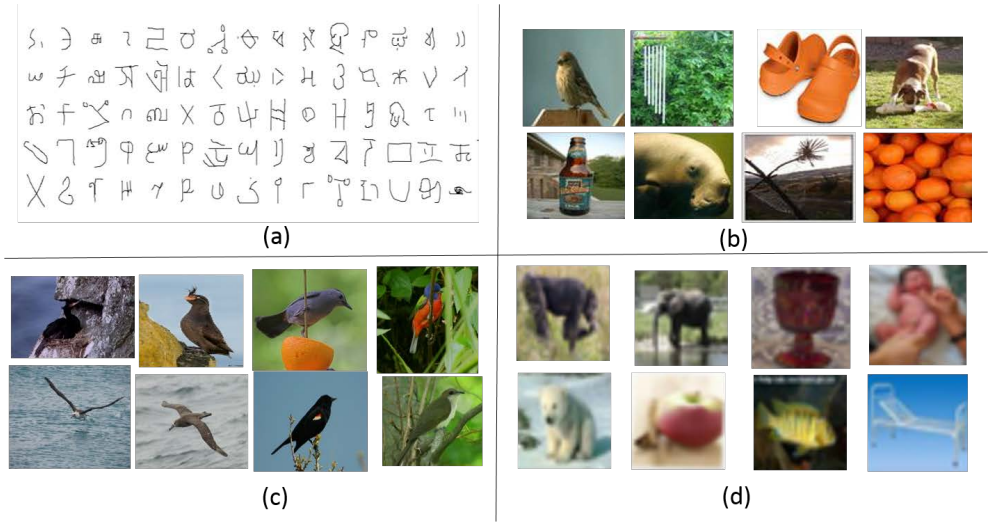


Fig. 3.7. Instances of the dataset used in our experiment for (a) Omniglot, (b) miniImagenet, (c) CUB-200, and (d) CIFAR-100.

3.3.2 Implementation

In this sub-section, we discuss the details of our neural network architecture and the training procedure. For the feature extractor module ($\mathbf{f}_\phi$) of our trainable neural network architecture, we use four convolutional blocks. This feature extractor

architecture is the same as used in previous works [86, 87]. This is done for the sake of fair comparison. Most of these previous works selected the feature-extraction architecture empirically. For shallow convolutional architecture and therefore more high-dimensional feature space, the performance is poor because the features extracted are not robust and not class-discriminative enough. But, as the depth of the convolutional architectures increases to a certain limit, we obtain a more informative low-dimensional feature space and therefore better recognition performance. The authors of [86,87] experimented and found that the presented four-convolutional-blocks-based architecture is lightweight and optimal. Each of these blocks consists of a 64-filter 3×3 convolution layer with SAME padding, batch normalization layer, and an ReLU activation followed by a 2×2 max-pooling layer all stacked upon another. The batch normalization [186] results in better recognition performance because it prevents internal covariate shift. When a 28×28 Omniglot image is applied as an input to these four convolutional blocks, its output results in a 64-dimensional feature space.

The variance estimator f_V consists of two convolutional blocks. Each convolutional block consists of 1×1 convolution layer with SAME padding, batch normalization layer and an ReLU activation layer. The first and the second convolutional blocks consist of 32 and 1 filters, respectively. The last layer producing the variance has softplus operation as the activation function. This is selected to produce only positive outputs.

The transformation layer $\mathbf{f}_{\mathrm{T}_{11}}$ consists of three fully connected layers of 128, 96 and 64 dimensions. Except the last layer, all the layers contain batch-normalization and ReLU activation functions. The last layer does not have an ReLU activation so that it can provide both negative and positive transformation shifts as output. The overall architecture of all the modules used for the Omniglot dataset is shown in Fig. 3.8(a).

The neural-network structure was trained using the stochastic gradient descent variant *Adam* [179] with an initial learning rate of 10^{-3}. The first-stage training was carried out using 60-way 5-shot with 5 query points per episode. The higher way is

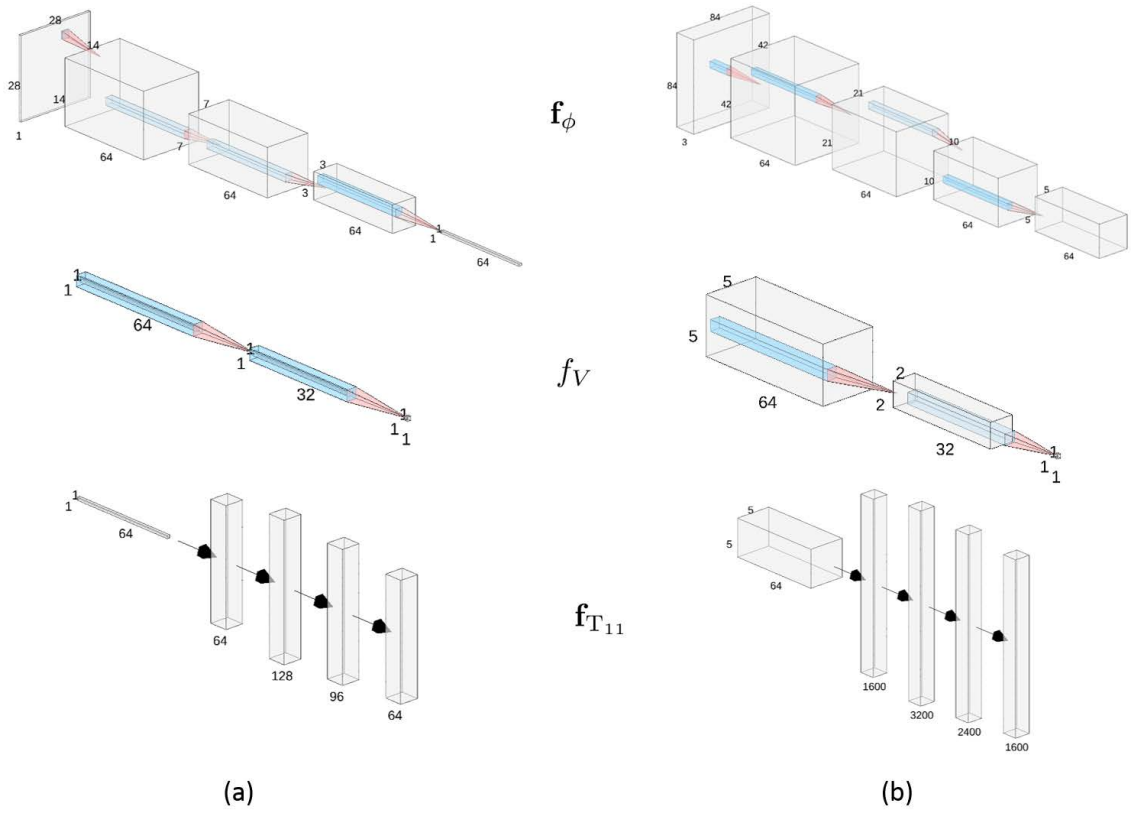


Fig. 3.8. Network architecture used for different modules $\mathbf{f}_\phi$, f_V and $\mathbf{f}_{\mathrm{T}_{11}}$. (a) For the Omniglot dataset, $\mathbf{f}_\phi$ produces a 1×64 dimensional feature map from a $28 \times 28 \times 1$ dimensional input image. The f_V module produces a scalar variance from the feature map. The $\mathbf{f}_{\mathrm{T}_{11}}$ regresses a 64-dimensional output from the feature map. (b) For the miniImagenet dataset, $\mathbf{f}_\phi$ produces a 5×5×64 dimensional feature map from a 84×84×3 dimensional input image. The f_V module produces a scalar variance from the feature map. The $\mathbf{f}_{\mathrm{T}_{11}}$ regresses a 1600-dimensional output from the feature map.

chosen in training so that the model can learn a more difficult task of distinguishing more classes and therefore produce a more discriminative feature space. In this Chapter, the second-stage training episodic setup is always kept the same as the testing episodic setup for all the experiments; that is, if the testing setup is N-way K-shot, so is the second-stage training setup.

The hyper-parameters λ_ρ, λ_r, and t_h were set to 0.1, 10^{-4}, and 0.02, respectively. It is important to note that these hyper-parameters are kept fixed for a particular dataset. This is mainly because cross-validation is not always feasible for the few-shot

learning setting, which contains only a few samples from the target category. Also, the validation classes are not representative of the test classes.

For reporting the recognition performance, 1000 random test episodes were selected and accuracy was obtained by averaging over all the test episodes. Each episode contained the corresponding N-way K-shot support samples and 5 query samples per way for testing.

For the miniImagenet dataset, we used the same feature extraction network architecture as the Omniglot dataset. However, since the miniImagenet dataset has images of size 84×84×3, the convolution module produces a 1600-dimensional feature vector. The variance estimator is also the same as that of the Omniglot dataset except that this estimator contains a 2×2 max-pooling stage before the non-linearity. This is required to reduce the $5 \times 5 \times 64$ (1600-dimensional) feature map to a scalar variance value. The transformation layer $\mathbf{f}_{\mathrm{T}_{11}}$ consists of three fully connected layers of 3200, 2400 and 1600 dimensions. The overall architecture of all the modules used for the miniImagenet dataset is shown in Fig. 3.8(b).

The hyper-parameters λ_ρ, λ_r and t_h were set to 0.1, 10^{-4} and 0.02, respectively. For testing on the 5-way 1-shot and 5-way 5-shot episodic strategy, we used a 20-way 1-shot and 20-way 5-shot sampling strategy, respectively, in the first-stage training. Each episode contained the corresponding N-way K-shot support samples and 15 query samples per way for testing. Results were reported by computing the average accuracy over 600 such randomly sampled episodes with 95% confidence interval.

For the CUB-200 and CIFAR-100 datasets, we used the same four-convolutional-blocks-based architecture as the feature extractor that has been previously used on the miniImagenet and Omniglot datasets. This embedding results in 1600 and 256 dimensional feature spaces for the CUB-200 and the CIFAR-100 datasets, respectively. The transformation layer $\mathbf{f}_{\mathrm{T}_{11}}$ for the CUB-200 dataset consists of three fully connected layers of 3200, 2400 and 1600 dimensions. The architecture of f_V for the CUB-200 dataset is the same as that of the miniImagenet dataset. The transformation layer $\mathbf{f}_{\mathrm{T}_{11}}$ for the CIFAR-100 dataset consists of three fully connected layers of

512, 384 and 256 dimensions. The architecture of f_V for the CIFAR-100 dataset is similar to that of the miniImagenet dataset except that the 2×2 max-pooling step is applied only on the second convolutional block. The hyper-parameters λ_ρ, λ_r and t_h were set to 1.0, 10^{-3} and 0.5 on both the CUB-200 and the CIFAR-100 datasets. It is important to note that for a fair comparison, we only report previous work that used the simple four-convolutional-block-based embedding instead of the more sophisticated ResNet [23] architecture.

3.3.3 Comparison against Related Approaches

Since our proposed few-shot learning method has both meta-learning and metric-learning components, we compared our proposed method against recent meta-learning [91, 93, 99] and metric-learning [85–87, 89] methods. We also compared against recent memory-based models [98, 187] and the Neural Statistician method [188] that learns how to represent statistics of the data. The results of the comparisons on the Omniglot dataset are shown in Table 3.1.

Table 3.1. Results of few-shot classification on the Omniglot dataset. Accuracies in % are reported as averaged over 1000 test episodes. Some of the studies report 95% confidence interval while some do not report results as shown by '–'.

Method	5-way 1-shot	5-way 5-shot	20-way 1-shot	20-way 5-shot
SIAMESE [85]	97.3	98.4	88.1	97.0
MANN [98]	82.8	94.9	-	-
MATCHING NET [86]	98.1	98.9	93.8	98.5
SIAMESE MEMORY [187]	98.4	99.6	95.0	98.6
NEURAL STATISTICIAN [188]	98.1	99.5	93.2	98.1
MAML [93]	98.7±0.4	**99.9±0.1**	95.8±0.3	98.9±0.2
META NET [99]	99.0	-	97.0	-
PROTO NET [87]	98.8	99.7	96.0	98.9
RELATION NET [89]	**99.6±0.2**	99.8±0.1	**97.6±0.2**	**99.1±0.1**
OUR PROPOSED METHOD	99.2±0.3	99.5±0.2	97.2±0.3	98.9±0.3

As seen from Table 3.1, most of the recent methods achieved almost perfect recognition performance on the Omniglot dataset (8 out of 10 methods obtained an average accuracy of more than 98% for the 5-way 1-shot task). Our proposed method obtained an average accuracy of 99.2% and 97.2% for the 5-way 1-shot and 20-way 1-shot tasks, respectively, which are better than most of the previous approaches. However, Relational Network [89] produced the best result; that is, 99.6% and 97.6% for the 5-way 1-shot and 20-way 1-shot tasks, respectively, because it learned a distance metric while our proposed method used a predefined Mahalanobis distance metric. The confidence interval of our proposed method (98.9%-99.5%) also overlapped with that of the Relational Network approach (99.4%-99.8%) for the 5-way 1-shot task. The confidence interval overlapped for the 20-way 1-shot task as well. As expected, higher shots during the testing produced better results (98.9%>97.2% for the 20-way task) for our proposed method because they represented the class statistics better than by just using one shot. Also, higher ways produced worse result (97.2%<99.2% for the 1-shot task) because there were more potential classes to choose from and the chances of misclassification were higher.

For the miniImagenet dataset, the comparison is more challenging and there is more room for improvement towards perfect performance. The results of the comparison are shown in Table 3.2. From Table 3.2, we can see that our proposed method produced an average accuracy of 52.68% and 70.91% on the 5-way 1-shot and 5-way 5-shot tasks, respectively, which are better than most of the previous methods. This can be mainly attributed to our two-stage training procedure, where the model learns to both represent and classify in a low-shot regime. However, the methods – Predicting Parameters from Activation [103] (PPA) and Transductive Propagation Networks [95] (TPN) produced better results than our proposed method in the 1-shot setting. Upon inspection, we realized that the PPA method used pre-trained embedding while most other few-shot learning methods and our training method of the embedding/feature extractor were done from scratch. Using a pre-trained embedding implies that datasets beyond the base and novel categories have been used in training

Table 3.2. Results of few-shot classification on the miniImagenet dataset. Accuracies are reported as averaged over 600 test episodes. Most of these studies report 95% confidence interval while unreported results are shown as '–'.

Method	5-way 1-shot	5-way 5-shot
META-LSTM [91]	43.44±0.77	60.60±0.71
MAML [93]	48.70±1.84	63.11±0.92
MATCHING NET [86]	43.56±0.84	55.31±0.73
META NET [99]	49.21±0.96	–
PROTO NET [87]	49.42±0.78	68.20±0.66
RELATION NET [89]	51.38±0.82	67.07±0.69
GNN [106]	50.33±0.36	66.41±0.63
REPTILE [189]	49.97	65.99
TPN [95]	53.75	69.43
PPA [103]	**54.53±0.40**	67.07±0.20
R2D2 [105]	51.8±0.2	68.4±0.2
OUR PROPOSED METHOD	52.68±0.51	**70.91±0.85**

the model and therefore the model would not be suitable for comparison. However, we still included the results for PPA in Table 3.2 for the sake of completeness. Also, the TPN method uses a transductive approach which assumes all the test/query data are available as a batch. The improvement in performance of this method is mainly due to the fact that the authors used the manifold of the unlabeled test data as well as support data to do inference. However, the method might not work if the number of query points is less or the query points arrive in a streaming fashion as in a real-world situation.

The results of our proposed method in comparison with previous work for the CUB-200 and CIFAR-100 datasets are shown in Table 3.3 and Table 3.4, respectively. In Table 3.3, on the CUB-200 dataset, our proposed method produced about 6 points

improvement over the second best method. Similarly, in Table 3.4, on the CIFAR-100 dataset, our proposed method produced around 2 points improvement over the second best method. This suggests that our proposed method can provide competitive performance on fine-grained and low-resolution datasets as well. Also, the average performance on the CUB-200 dataset is less than that on the CIFAR-100 dataset. This is because the CUB-200 dataset contains more fine-grained categories compared to the CIFAR-100 dataset and therefore classes overlap more in the CUB-200 dataset.

From these comparative studies, it is not clear how all the modules in our trainable neural-network architecture contributed to the performance. Therefore, we resort to further analyzing each component of our proposed method in the following sub-sections.

Table 3.3. Results of few-shot classification on the CUB-200 dataset where our accuracy is reported as averaged over 600 test episodes.

Method	5-way 1-shot	5-way 5-shot
META-LSTM [91]	40.43	49.65
MAML [93]	38.43	59.15
MATCHING NET [86]	49.34	59.31
PROTO NET [87]	45.27	56.35
OUR PROPOSED METHOD	**55.85**	**66.73**

3.3.4 Ablation Study with Varying Training and Testing Conditions

The contribution of this Chapter consists of the following modules on top of the Prototypical Network (PN) – a variance estimator (V), the relative features (R), and the category-agnostic transformer (T). We thus performed an ablative study, where we added all combinations of the modules on the PN and observed the change in

Table 3.4. Results of few-shot classification on the CIFAR-100 dataset where the accuracy is reported as averaged over 10000 test episodes. Most of these studies report 95% confidence interval.

Method	5-way 1-shot	5-way 5-shot
MAML [93]	58.9±1.9	71.5±1.0
PROTO NET [87]	55.5±0.7	72.0±0.6
RELATION NET [89]	55.0±1.0	69.3±0.8
GNN [106]	61.9	75.3
R2D2 [105]	65.4±0.2	79.4±0.2
OUR PROPOSED METHOD	**67.15±0.3**	**81.65±0.3**

performance. Results of this experiment are reported in Table 3.5 as the training way is varied for the 5-way 1-shot and 5-way 5-shot testing conditions.

Table 3.5. Ablative study of our approach on the miniImagenet dataset. Averaged accuracy is reported as the training way is varied. Ablations include the Variance estimator (V), Relative features (R), and Category-agnostic Transformer (T). The baseline is the Prototypical Network (PN).

	5-way 1-shot Testing						5-way 5-shot Testing					
Training way	5	10	15	20	25	30	5	10	15	20	25	30
PN	43.987	46.956	46.589	46.122	47.253	47.3	62.693	64.742	64.524	63.578	62.416	61.9
PN+V	44.411	47.067	47.936	48.304	47.778	48.067	64.813	65.033	66.158	65.37	64.318	64.82
PN+R	47.849	50.309	52.631	52.607	52.14	51.996	66.758	70.831	70.771	70.447	71.147	62.733
PN+T	43.942	45.944	47.263	48.022	48.122	48.011	62.396	63.316	64.342	63.024	63.531	64.86
PN+V+R	49.322	51.057	51.031	52.782	52.716	51.773	69.1	70.936	71.496	71.36	70.36	68.23
PN+V+T	45.689	47.927	48.422	48.002	47.693	47.947	61.667	63.484	63.736	62.431	61.978	63.48
PN+R+T	46.913	51.224	52.338	53.036	53.789	53.66	68.76	71.38	72.34	72.151	72.584	67.34

We provided our own implementation of PN in this experiment and future experiments. From Table 3.5, it reveals that the addition of the relative features (R) has the most significant effect on the performance followed by the variance estimator (V) and the category-agnostic transformer (T). This is because relative features try

to diminish the difference between feature dimensionality and the number of samples, and thus try to alleviate overfitting. On the other hand, PN+T has negligible improvement or slightly worse performance compared to the PN baseline. This is because prototypical networks tend to cluster same-class samples very close to one another and therefore additional transformation stage (T) to map samples to prototype might be redundant. In certain cases, the complex non-linear transformation might over-fit to produce worse performance. It should be noted that higher ways in training do not always produce better performance. For example, in a 5-way 1-shot testing, PN+R produced a peak in performance for the 15-way training strategy with a dip in performance on either side. Similar pattern can be observed for the 5-way 5-shot testing results. The effect of relative features is also significant in case pairs of modules are added to the PN baseline. In Table 3.5, we can see that PN+V+R and PN+R+T reached accuracy levels over 50% and 70% for the 5-way 1-shot and 5-way 5-shot testing cases, respectively, but PN+V+T failed to do so. An interesting observation is that the combined effects of R+T mostly provided better performance than V+R even though V provided better performance than T. This suggests that adding modules upon the PN baseline did not always produce additive effects but they also produced interactive effects between the two modules.

3.3.5 Parameter Sensitivity Studies

We also performed experiments to find how the performance of PN+R varied with changing λ_ρ. The results are shown in Fig. 3.9 for both 5-way 1-shot and 5-way 5-shot testing conditions. The training condition for 5-way 1-shot testing is 20-way 1-shot and that for the 5-way 5-shot testing is 20-way 5-shot. The PN baseline is shown using the dotted line. From the plot, it is shown that the accuracy followed a bell-curve with the maximum accuracy observed at $\lambda_\rho = 1$. It is recommended not to use $\lambda_\rho > 1$ as it caused degradation in performance, which was sometimes worse than

the PN baseline. This is because putting excess weight on relative features diminishes the effect of absolute features that are crucial for recognition.

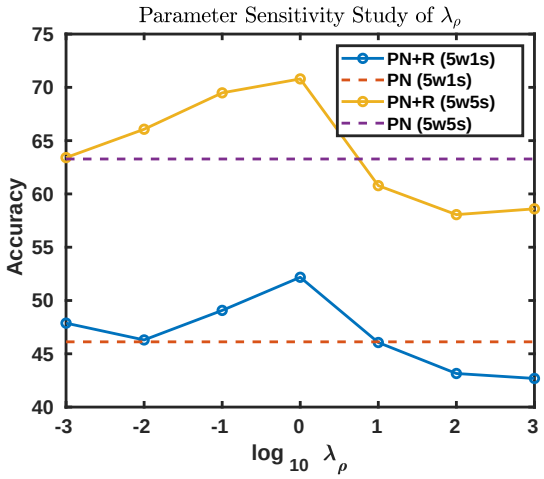


Fig. 3.9. Plot of accuracy with respect to λ_ρ for 5-way 1-shot (5w1s) and 5-way 5-shot (5w5s) testing conditions with the prototypical network baseline. The dataset used is miniImagenet.

We also studied the effect of changing t_h and λ_r on the recognition performance for different testing shots. In Fig. 3.10, we see that the performance varied for different thresholds with a peak performance obtained for a value of t_h between 0 and 1. In fact, for the higher shot configuration, the peak performance was obtained at a higher threshold. This is because for higher shots, the contribution of the few-shot sample mean was much more compared to the contribution of the base categories. As a result, a higher threshold t_h was required to reduce the contribution of the base classes.

In Fig. 3.11, we observed how the recognition performance changed as λ_r was varied for different shots. As expected, the peak performance was better than the baseline $\lambda_r = 0$ shown in dashed lines. However, the sensitivity at the 5-shot configuration was less compared to that in the 1-shot configuration. This is because, for higher shots, the constraint corresponding to λ_r - that the sample mean should be close to the prototype is automatically satisfied and therefore changing the value of λ_r did not change the performance much.

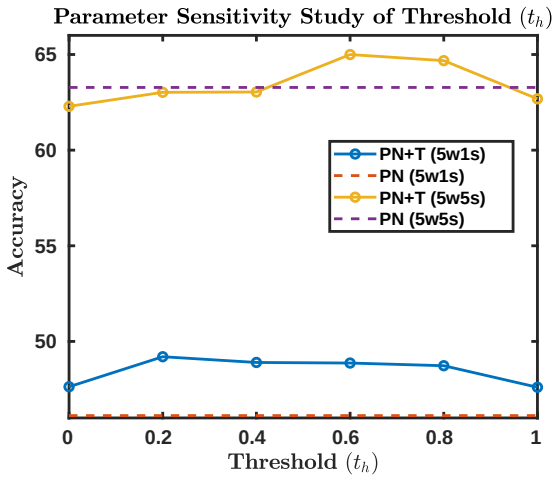


Fig. 3.10. Plot of accuracy with respect to t_h for 5-way 1-shot (5w1s) and 5-way 5-shot (5w5s) testing conditions with the prototypical network baseline. The dataset used is miniImagenet.

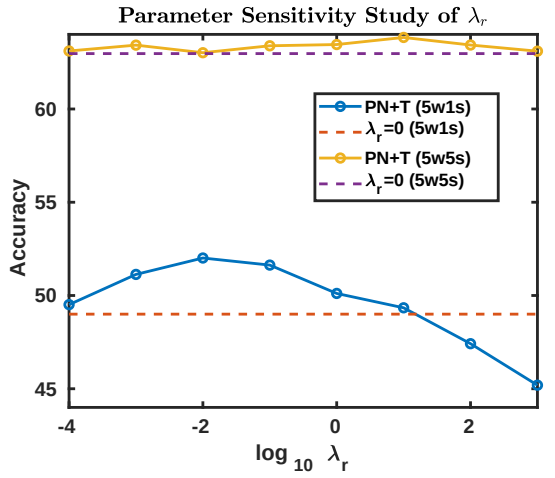


Fig. 3.11. Plot of accuracy with respect to λ_r for 5-way 1-shot (5w1s) and 5-way 5-shot (5w5s) testing conditions. The dataset used was miniImagenet.

We did additional sensitivity studies of t_h and λ_r over a smaller range of values. The results are reported in Tables 3.6 and 3.7 for t_h and λ_r, respectively. From the results, it showed that there was very little change when the parameters were varied over such a small range. However, the response was oscillatory probably because of the non-convexity of the loss functions used in our framework.

Table 3.6. Performance sensitivity with respect to threshold t_h over a small range. The dataset used is miniImagenet.

t_h	0.02	0.04	0.06	0.08	0.1
5-way 1-shot	48.01	48.23	48.11	48.34	48.66
5-way 5-shot	62.39	62.31	62.54	62.51	62.73

Table 3.7. Performance sensitivity with respect to λ_r over a small range. The dataset used is miniImagenet.

λ_r	1e-4	2e-4	4e-4	8e-4
5-way 1-shot	49.51	50.64	50.50	50.78
5-way 5-shot	63.11	63.26	63.36	63.34

3.3.6 Feature Visualization

We also visualized the features in two dimensions using t-SNE [174] as shown in Fig. 3.12. From Fig. 3.12(a), it is clear that PN produced a very compact feature space, where the classes were very difficult to distinguish. On the other hand, the features obtained using PN+R+V as shown in Fig. 3.12(b) were more distinguishable class-wise. This resulted in better recognition performance.

It is important to note that removing the outlier from Fig. 3.12(a) and rescaling the figure would make the image similar to Fig. 3.12(b). This is the point of difference between using Prototypical Network (PN) and our (PN+R+V) method. Using PN, we obtained more scaled-down features. Thus, these features were closer to one another, resulting in more difficult classification compared to our (PN+R+V) method. However, distinguishing classes in both cases was complicated and that is why we used the Euclidean-distance-based differential nearest-neighbor classifier.

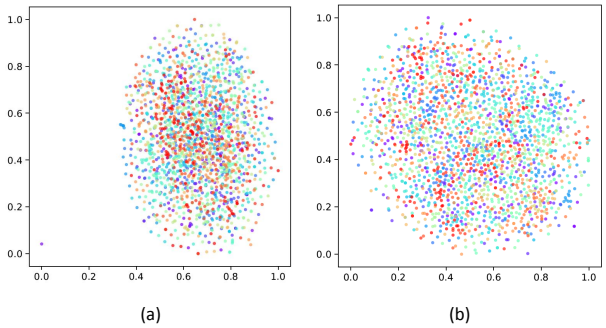


Fig. 3.12. t-SNE plot for (a) PN and (b) PN+R+V ($\lambda_\rho = 1$). The dataset used was miniImagenet. Same color corresponds to different samples of the same category.

3.3.7 Convergence Results

We also reported the training and testing performance with increasing training episodes in Fig. 3.13. We used the 5-way 5-shot and 20-way 5-shot settings for testing and training, respectively. As shown in Fig. 3.13, the test accuracy for PN+V+R rose fast compared to that of PN. Also, the training accuracy was quite noisy. This is because each training episode produced a newer set of categories and therefore there was a high variance in the training accuracy.

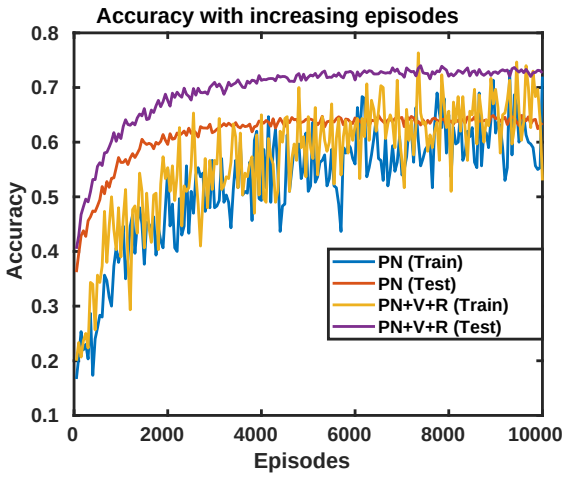


Fig. 3.13. Training and test accuracy with increasing number of episodes for the prototypical network (PN) baseline and our proposed approach using relative features and variance estimator (PN+V+R).

3.3.8 Effect of Number of Samples

Since the relative features are constructed using both the support and query points, it is worthwhile to note the effect on recognition performance by changing the number of query points per class in the training and testing stages. We performed two experiments for the PN+R case. The first experiment considered the situation when the number of training query points per class was fixed at 15 and the number of test query points was varied. The second experiment considered the situation when the number of test query points per class was fixed at 15 and the number of training points was varied. In Fig. 3.14, it is shown that as the number of query points increased, the recognition performance increased and it became saturated after a while. This is because query points beyond a certain quantity did not provide additional second-order structural information. Also, from the poor performance in the case of one test query sample, it is evident that having sufficient query samples in the testing stage was more important than having sufficient quantity of query samples in the training stage.

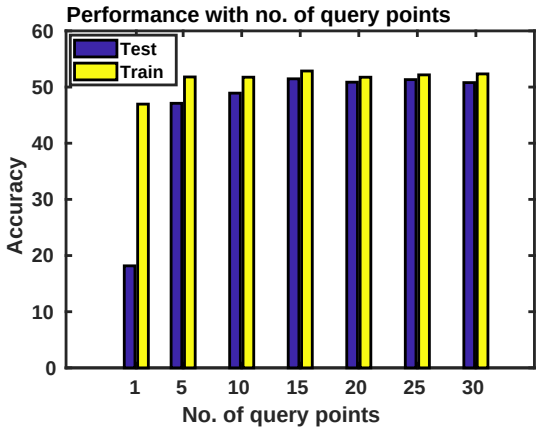


Fig. 3.14. Plot of accuracy when the number of training query points is fixed and the number of test query points is varied and vice-versa. The dataset used was miniImagenet.

3.3.9 Effect of Base Categories

We also evaluated how the performance of PN+T varied as the number of source base categories changed. Results are shown in Table 3.8. The recognition performance increased with the increasing number of source categories. This is because the increasing number of source categories trained a robust feature space. Also, the probability of finding relevant categories became more for the category-agnostic transformation stage. The performance of the category-agnostic transformation became poorer at higher shots compared to PN. This is because the transformation became closer to identity and its significance became less.

Table 3.8. Performance analysis as the number of base categories is varied for the PN+T case. The dataset used is miniImagenet.

Source No.	20	30	40	50	60
5-way 1-shot (PN)	40.10	42.196	44.14	45.66	45.74
5-way 1-shot (PN+T)	41.61	43.74	45.48	46.82	47.20
5-way 5-shot (PN)	45.89	51.93	55.95	59.796	60.96
5-way 5-shot (PN+T)	43.89	50.93	55.85	59.70	61.14

Till now, we have tested our proposed approach on the novel categories. It is also important to test our proposed approach on base categories since they are more common and are likely to be observed more frequently compared to novel categories. The results of applying our proposed approach to the base categories are shown in Table 3.9 for different testing settings. As expected, the performance on base categories was better compared to that of novel categories. Furthermore, our proposed approach (PN+V+R) produced better results as compared to PN.

Table 3.9. Performance comparison of testing on the base training classes.

5-way 5-shot (PN)	5-way 1-shot (PN)	5-way 5-shot (PN+V+R)	5-way 1-shot (PN+V+R)
82.236	58.409	85.293	64.111

3.3.10 Analysis of Category-agnostic Transformation

We also carried out the ablation analysis of PN+T; that is, the addition of the category-agnostic transformer (T) on top of the prototypical network baseline (PN). As described previously, the category-agnostic transformer (T) consists of three modules - the neural-network-based transformer (T_{11}), the residual connection (T_{12}), and the contribution of the base prototypes (T_2). From Table 3.10, we can see that the addition of these modules gradually improved the recognition performance, suggesting that the addition of all these modules was important. The method PN+T_{11}+T_{12}+T_2 used a threshold $t_h = 0.02$. It is important to note that using PN+T_{11}+T_{12} was equivalent to PN+T_{11}+T_{12}+T_2 with threshold $t_h = 1$. We also performed an additional experiment using the method PN+T_{11}+T_{12}+T_2 with threshold $t_h = 0$. Using $t_h = 0$, we obtained an accuracy of 47.63% and 62.29% on the 5-way 1-shot and 5-way 5-shot classification tasks, respectively. The recognition performance was worse compared to using $t_h = 0.02$ because $t_h = 0$ caused all the base classes and therefore irrelevant classes to contribute to the category-agnostic transformation thus causing a negative transfer.

The category agnostic transformer consisted of contribution of the base categories as described mathematically through $\mathbf{f}_{T_2}$. Using the threshold mechanism, only relevant base categories were selected for contribution because these categories were closer to the novel category in the feature space compared to the irrelevant base categories. Using the thresholded probability vector $\mathbf{p}_c^{th}$, we selected the top three relevant base categories for a few novel categories. The results are shown in Table 3.11. As an example, all the top relevant categories for the African hunting dog have canine fea-

Table 3.10. Ablation analysis of each component of the category-agnostic transformer. The dataset used was miniImagenet.

Method	5-way 1-shot	5-way 5-shot
PN+T_{11}	47.393	62.411
PN+T_{11}+T_{12}	48.604	62.683
PN+T_{11}+T_{12}+T_2	49.002	63.024

tures. The relevant categories for the mixing bowl seem to fit in context. Pictures of Consomme and Hotdog are generally shown in plates or bowls. Also, the relevant categories of nematode, a worm-like organism involved insects and snakes. There could be erroneous selections like harvestman spider being the most relevant category for the Golden-retriever dog. This suggested that an additional class relevance criterion based on WordNet [190] might be more appropriate.

Table 3.11. Novel categories and top three relevant base categories.

Novel /Relevant Class	**Rank 1**	**Rank 2**	**Rank 3**
African hunting dog	Saluki	Arctic Fox	Komondor
Mixing bowl	Consomme	Hotdog	Ear
Golden-retriever	Harvestman	Miniature poodle	Bolete
Nematode	Green-mamba	Lady-bug	Spider-web

3.4 Conclusions

In this Chapter, we have proposed a two-stage framework for few-shot learning of image recognition. The framework has contributions at both the feature extraction stage and the classification stage of image recognition. At the feature extraction stage, we proposed the use of relative-feature representation as well as the Mahalanobis

distance metric with predictable variance. For the classification stage, we proposed a category-agnostic transformation that produces class prototypes from class samples. Results on standard few-shot learning datasets showed our approach to be comparable or even better than previous approaches. We also provided further analysis on our model and concluded that the relative-feature component was mostly responsible for the improvement of the performance of our proposed approach. In the future, we would like to extend our work to zero-shot classification, where we do not have any support samples from the novel class but only high-level semantic information for each of these classes.

4. PARAMETRIC AND NON-PARAMETRIC APPROACHES TO FEW-SHOT LEARNING

4.1 Introduction

In this Chapter, we extend the traditional few-shot learning (FSL) problem to the situation when the source-domain data is not accessible but only high-level information in the form of class prototypes is available. This limited information setup for the FSL problem deserves much attention due to its implication of privacy-preserving inaccessibility to the source-domain data. This limited information FSL (LI-FSL) problem has rarely been addressed before and is depicted in Fig. 4.1.

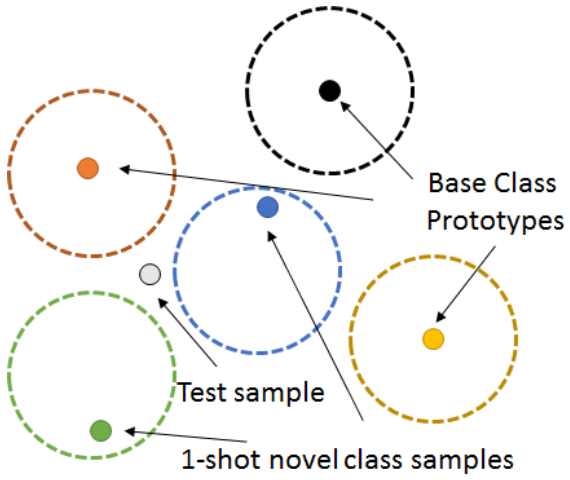


Fig. 4.1. In this limited-information FSL problem setting, the base-class prototypes are known but not the novel-class prototypes. The spreads of the classes (dashed boundaries) are also unknown.

Previous works that are closely related to our LI-FSL problem setting include [110, 112, 191]. These methods assumed access to source-class models, where the target-class models are constrained as a linear [110] or non-linear [112] combination

of source-class models as shown in Fig. 4.2. As a result, the recognition performance would not only depend on their proposed transfer-learning mechanism but also on the choice of their source-class model family. Hence, this setting does not allow for fair comparison as results would vary with changing source-model families. On the other hand, our proposed setting uses transferable information in the form of source-class prototypes as shown in Fig. 4.2. This setting allows for fair comparison, where recognition performance depends only on the data and the proposed transfer learning approach. Thus, we redefine the LI-FSL problem setting where the results of our proposed approach can serve as benchmark for future research.

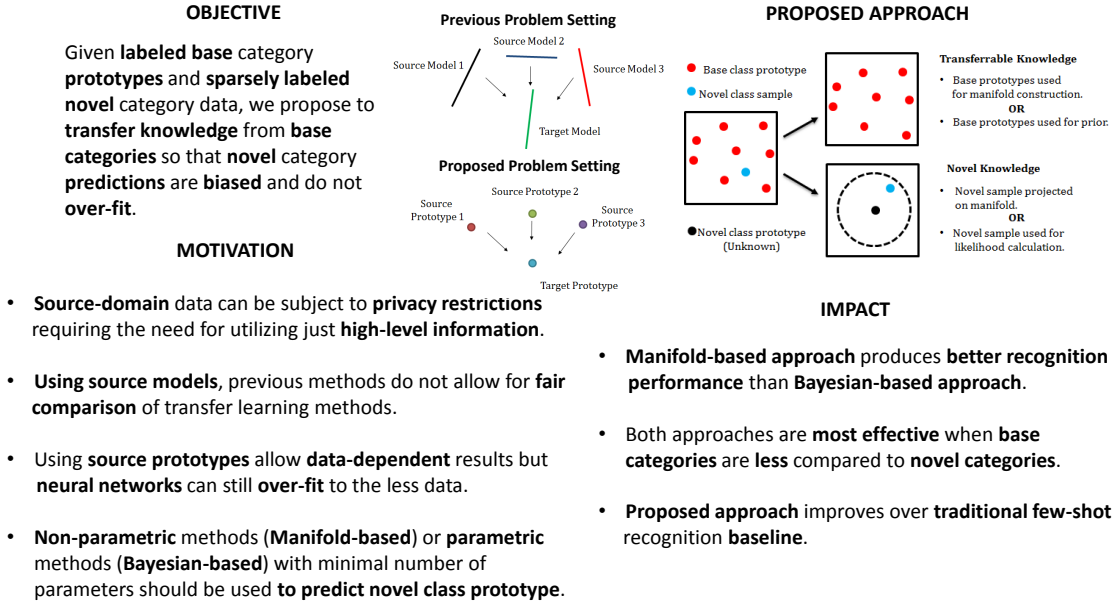


Fig. 4.2. Overall outline of our proposed approach including motivations and impacts.

Most previous methods in FSL encoded the transferable prior knowledge from the source domain using a parametric model. Since these methods assumed access to lots of labeled data from the source domain, neural-network-based models are used to encode the transferable knowledge. However, the neural-network-based parametric models might severely overfit if we only have limited data in the form of class proto-

types available from the source domain. Hence, in order to solve this LI-FSL problem, it is natural to seek a non-parametric approach or a simple parametric approach with minimal number of parameters. In this Chapter, we address this LI-FSL setting by formulating it as a case of *ill-sampling*. As depicted in Fig. 4.1, the correct locations of the base category prototypes are known but that of the novel categories are unknown. This is because the few-shot data from a novel category might be sampled from the periphery of the class distribution. These non-representative samples when used for classification will result in poor recognition performance. Therefore, we propose to use the parametric- or the non-parametric-based prior to produce a biased estimate of the novel-class prototype location.

For the non-parametric-based prior, we find inspiration from the idea [154] that data samples from one class lie on a low-dimensional subspace. Therefore, we can consider all the classes as a collection of piece-wise linear subspaces. This set of subspaces can be considered as an approximation of a non-linear manifold close to which the class-prototypes lie. This manifold serves as a structural prior to estimate the location of the novel-class prototype. The subspace near the novel-class prototype is found by calculating the mean of the subspaces on which the nearby base classes lie. The subspace on which the nearby base classes lie is again found using their nearest neighbors as shown in Fig. 4.3. Finally, the novel-class sample can be projected onto the mean subspace to obtain the corresponding novel-class prototype.

Once the novel-class prototypes are estimated, one can use the nearest-neighbor (NN) approach to assign a test sample to a class based on the Euclidean distance to all the prototypes. However, directly using NN in such a high-dimensional feature space might lead to the phenomenon of *hubness* [192], where the NN prediction distribution is skewed towards only a few class prototypes and rest of the classes are not considered. Moreover, the estimation procedure for the novel-class prototype might still be error prone. Hence, there is a need to exploit the structural arrangement of the manifold containing all the classes to assign a class to a test sample. This can be achieved by constructing a graph using all the class prototypes and then using

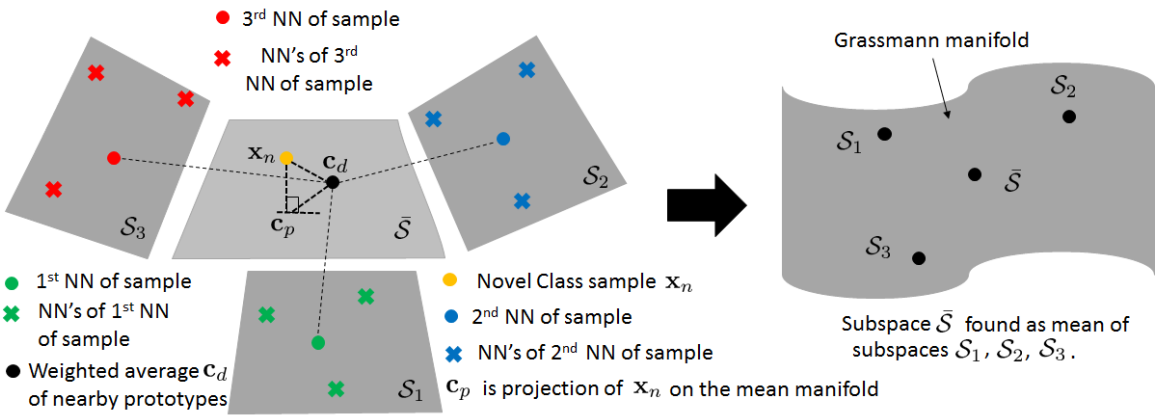


Fig. 4.3. The surrounding subspaces $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ for a novel-class sample $\mathbf{x}_n$ are found by using its Nearest Neighbors (NN) and the NNs of its NNs. These subspaces when orthonormalized are represented as points on the Grassmann manifold. Their mean can be calculated to obtain the subspace $\bar{\mathcal{S}}$ on which the novel-class sample $\mathbf{x}_n$ is projected to obtain $\mathbf{c}_p$. The weighted average $\mathbf{c}_d$ of the nearby prototypes are also used to obtain the novel-class prototype.

equilibrium probability of an induced absorbing Markov-chain process to output the most probable class.

The performance of a manifold-based, non-parametric approach may not be as competitive as a parametric approach. This is because the manifold is constructed directly using the data samples and without any assumption about the model family the manifold belongs to. Alternatively, we could use the Bayesian method as a parametric approach where the class distribution is assumed to belong to the Gaussian family. The Bayesian approach uses a minimal number of trainable parameters and is less likely to overfit compared to that of a neural network. The parameters of the Gaussian distribution is then estimated using a maximum-a-posterior method. The prior combined with the likelihood information obtained from the novel-class samples is used to obtain the posterior estimate of the novel-class prototype. The estimated prototype will be closer to the true prototype compared to the few-shot sample mean thus facilitating classification. The concept is visually described in Fig. 4.4.

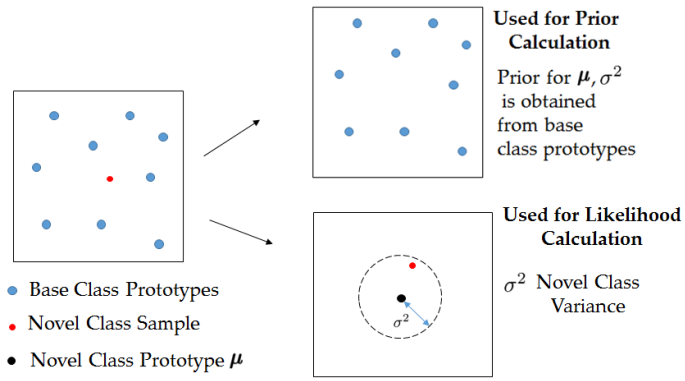


Fig. 4.4. The proposed Bayesian framework. The mean ($\boldsymbol{\mu}$) and the variance (σ^2) of the novel class are unknown. A Gaussian likelihood is formulated using the novel-class data-samples with $\boldsymbol{\mu}$ and σ^2 as parameters. The priors for the $\boldsymbol{\mu}$ and σ^2 are obtained from the base-class prototype locations.

To summarize, our major contributions in this Chapter are as follows: (a) We introduce the limited-information setting of few-shot learning where only base category prototypes are available; (b) We propose both non-parametric and parametric approaches to solve this limited-information setting. The non-parametric- and parametric-based model uses a manifold and Bayesian prior, respectively. All these priors are constructed from the base category prototypes; (c) We then study different models using different priors and hyper-parameters and consequently perform experiments and analyses on two image datasets – the large-scale ImageNet and the small-scale but fine-grained CUB-200.

4.2 Proposed Approach

4.2.1 Notation

Consider that we have access to the base category prototypes collected in the form of a matrix $\mathbf{C} \in \mathbb{R}^{n_b \times d}$, where n_b is the number of base prototypes and d is the dimensionality of the feature space. We also have access to k examples $\{\mathbf{x}_{n1}, \mathbf{x}_{n2}, ..., \mathbf{x}_{nk}\}$ from the n^{th} novel class, where $\mathbf{x}_{ni} \in \mathbb{R}^d$ and k is very small. Our goal is to estimate

the location of the novel-class prototype $\mathbf{c}_n \in \mathbb{R}^d$. In our case, a prototype of a base class is the arithmetic mean of all the samples in the class. Using these notations, we will discuss our proposed parametric (Bayesian-based) and non-parametric (manifold-based) approaches in the next two sub-sections. $\mathbf{C}$ is used to construct the manifold or the prior in case of the Bayesian approach. Then, the novel-class samples $\mathbf{x}_{ni}$ are used to estimate the novel-class prototype.

4.2.2 Proposed Manifold-based approach

Estimating Novel-Class Prototypes

We assume that all the base and novel-class prototypes lie close to a non-linear manifold; that is, all rows of $\mathbf{C}$ and $\mathbf{c}_n$ lie close to a non-linear manifold. Since the mathematical expression of the manifold is unknown, we express it as a collection of piece-wise linear subspaces. First, we find the surrounding classes of the novel class by finding r nearest-neighboring prototypes of the novel-class sample $\mathbf{x}_n$. Let these nearest-neighboring prototypes be denoted as $\mathbf{c}_{ni} \in \mathbb{R}^d$ for $i \in \{1, 2, ..., r\}$. For each of the r neighboring prototypes, we find q neighboring prototypes. These new prototypes can be expressed as $\mathbf{c}_{nij} \in \mathbb{R}^d$ for $j \in \{1, 2, ..., q\}$ and $i \in \{1, 2, ..., r\}$. Hence, $\mathbf{c}_{nij}$ represents the j^{th} nearest neighbor of the i^{th} nearest neighbor of the novel-class sample $\mathbf{x}_n$. To represent the non-linear manifold, we form r linear subspaces using the r nearest neighbors of the novel sample as well as the q nearest neighbors of each of the r prototypes. The linear subspace $\mathcal{S}_i$ corresponding to the i^{th} nearest neighbor of $\mathbf{x}_n$ is represented as a column space such that $\mathcal{S}_i \equiv [\mathbf{c}_{ni} \vdots \mathbf{c}_{ni1} \vdots \mathbf{c}_{ni2} \vdots .... \vdots \mathbf{c}_{niq}]$. The linear subspace can be orthonormalized to obtain $\mathcal{S}_i^{\perp}$ and the operation can be repeated for all the r nearest neighbors. The net result is r linear subspaces with dimensionality $(q + 1)$ surrounding the novel-class sample $\mathbf{x}_n$. In the example in Fig. 4.3, we chose $r = 3$ and $q = 3$. These linear subspaces represent linearized localized versions of the non-linear manifold on which the class prototypes lie. The subspace on which the novel-class prototype lies close to can be found by averaging these surrounding

r subspaces $\mathcal{S}_i^{\perp}$ for $i \in \{1, 2, ..., r\}$. For finding the average of these orthonormal subspaces, we use the concept of Grassmann manifold.

Grassmann Manifold. A Grassmann manifold $\mathcal{G}(n, l)$ for $n, l > 0$ is the topological space composed of all l-dimensional linear subspaces embedded in an n-dimensional Euclidean space. A point on the Grassmann manifold is represented as an $n \times l$ orthonormal matrix $\mathbf{S}$ whose columns span the corresponding linear subspace $\mathcal{S}$. It is represented as: $\mathcal{G}(n, l) = \{\text{span}(\mathbf{S}):\ \mathbf{S} \in \mathbb{R}^{n \times l}, \mathbf{S}^T\mathbf{S} = \mathbf{I}_l\}$, where $\mathbf{I}_l$ is a $l \times l$-dimensional identity matrix and superscript T indicates matrix transpose.

Following this definition, the r orthonormal subspaces $\mathcal{S}_i^{\perp}$ for $i \in \{1, 2, ..., r\}$ are points lying on a $\mathcal{G}(d, q+1)$ Grassmann manifold. The average of these points on the Grassmann manifold will represent the linear subspace to which the novel-class prototype lies close to. The average of these points is found using the *extrinsic mean*. For a set of points on the Grassmann manifold $\mathcal{G}(d, q+1)$, the extrinsic mean is the point that minimizes the Frobenius-norm-squared difference of the projections of the points onto the space of $(q+1)$ ranked $d \times d$ matrices. Therefore, the optimization problem for finding the extrinsic mean $\bar{\mathbf{S}}$ is

$$\begin{aligned} &\underset{\mathbf{S}^T\mathbf{S}=\mathbf{I}}{\text{argmin}} \sum_{i=1}^{r} d(\mathbf{S}_i, \mathbf{S})^2, \\ &\text{where} \quad d(\mathbf{S}_i, \mathbf{S}) = \frac{||\mathbf{S}\mathbf{S}^T - \mathbf{S}_i\mathbf{S}_i^T||_{\mathcal{F}}}{\sqrt{2}}. \end{aligned} \tag{4.1}$$

Here $||\cdot||_{\mathcal{F}}$ is the Frobenius norm. Let $\mathbf{S}^*$ be the solution to the optimization problem (4.1), which can be found using eigenvalue decomposition as discussed in Appendix A. $\mathbf{S}^*$ is the spanning matrix of the extrinsic mean of the surrounding subspaces $\mathcal{S}_i^{\perp}$'s. Setting the extrinsic mean $\bar{\mathbf{S}} = \mathbf{S}^*$, we project the novel-class sample $\mathbf{x}_n$ onto the subspace spanned by the matrix $\bar{\mathbf{S}}$. The projected point $\mathbf{c}_p$ can be obtained as $\mathbf{c}_p = \bar{\mathbf{S}}\bar{\mathbf{S}}^{+}\mathbf{x}_n$, where $\bar{\mathbf{S}}^{+} = (\bar{\mathbf{S}}^T\bar{\mathbf{S}})^{-1}\bar{\mathbf{S}}^T$. Superscripts -1 and $+$ indicate matrix inverse and matrix pseudo-inverse, respectively.

We also consider the direct contribution of the surrounding class prototypes into estimating the novel-class prototype. If $\mathbf{C}_r \in \mathbb{R}^{r \times d}$ consists of the r nearest neighbors of the novel-class sample $\mathbf{x}_n$, then their contribution $\mathbf{c}_d$ to the novel-class prototype

location can be found using the equation $\mathbf{c}_d = \mathbf{C}_r^T \mathbf{p}_d$, where $\mathbf{p}_d \in \mathbb{R}^r$ is the probability vector formed by carrying out the exponential mapping of the Euclidean distances of the class prototypes to $\mathbf{x}_n$, followed by normalization. Hence, the contributions $\mathbf{x}_n$, $\mathbf{c}_p$ and $\mathbf{c}_d$ can be used to estimate the novel-class prototype location $\mathbf{c}_n$ as

$$\mathbf{c}_n = \alpha_2[\alpha_1 \mathbf{x}_n + (1 - \alpha_1)\mathbf{c}_p] + (1 - \alpha_2)\mathbf{c}_d, \tag{4.2}$$

where α_1, $\alpha_2 \in [0, 1]$ are scalar weights. These scalar weights are manually set and depend on how close $\mathbf{x}_n$ is to the true novel-class prototype. In case $\mathbf{x}_n$ is very close to the novel-class prototype, $\alpha_1 = \alpha_2 \approx 1$ will produce optimal classification performance.

Classification using Absorbing Markov Chain

Once the class prototype locations of the novel classes are known, the structural arrangement of the prototypes of both the base and novel classes are again used to recognize a test sample. The motivation behind using the structural arrangement of the classes is to obtain a more informed decision about the classification. This is useful if we do not have access to all the samples of the base classes and we do not know how spread out each class is. Nearest-neighbor classification will just compare with each and every class individually without regard to the global arrangement of the other classes.

The structural arrangement of the classes is represented using a k'-nearest-neighbor (k'-NN) graph, where each node represents a class prototype. The k'-NN graph formulation allows nodes to only be connected to its k'-NN nodes. The weights between the nodes are defined using the exponential of the negative Euclidean distances. Upon defining this graph, an absorbing Markov-chain process is induced on it. Each state of the Markov chain corresponds to a node in the graph and therefore a category. The transition probability from a state i to a state j is found as $p_{ij} = w_{ij} / \sum_l w_{il}$, where w_{il} is the weight connecting nodes i and l. In the absorbing Markov chain process, there are two kinds of states - *transient* and *absorbing*. The transient state

and the absorbing state have self-transition probabilities p_{ii} as 0 and 1, respectively. This suggests that a random walker on a graph cannot stay on the transient node for the next step but for the absorbing node it will stay there forever. An example of an absorbing Markov chain is given in Fig. 4.5, where the arrows represent the possible transitions from one node to another.

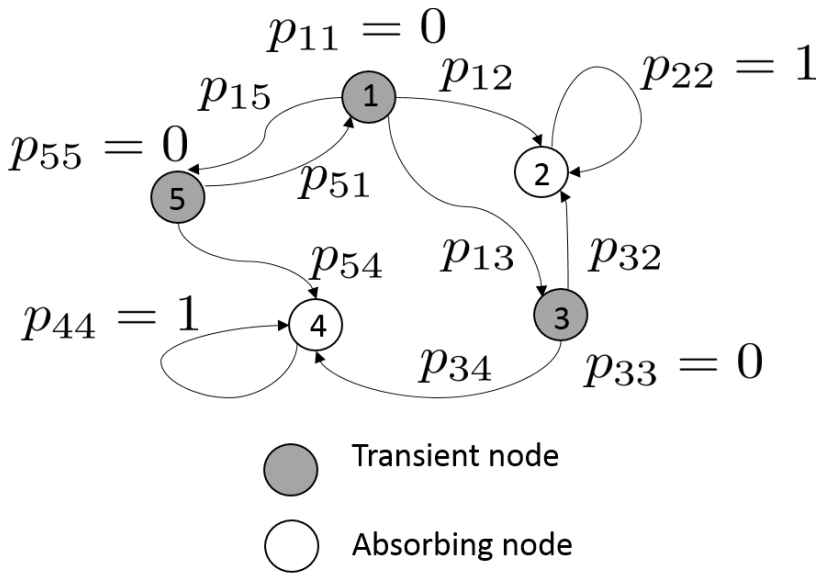


Fig. 4.5. A random walker transitions from a transient node while it terminates on an absorbing node. Possible transitions are shown with directed arrows. The transition probability from a state i to a state j is p_{ij}. The transient state and the absorbing state have self-transition probabilities p_{ii} as 0 and 1, respectively.

Overall, the Markov chain is represented using the transition matrix $\mathbf{P}$, which models the dynamics of the process. Using $\mathbf{P}$, the Markov-chain equations are described as follows:

$$\mathbf{u}^{t+1} = \mathbf{u}^t\mathbf{P}, \quad \text{where} \quad \mathbf{P} = \begin{bmatrix} \mathbf{T}_{n_t \times n_t} & \mathbf{A}_{n_t \times n_a} \\ \mathbf{0}_{n_a \times n_t} & \mathbf{I}_{n_a \times n_a} \end{bmatrix}, \tag{4.3}$$

$\mathbf{u}^t$ and $\mathbf{u}^{t+1}$ are the states of the process at instants t and $t+1$, respectively, and they are represented as a probability vector over all the states. $\mathbf{T}$ describes the transition probabilities from one transient state to another. $\mathbf{A}$ describes the transition

probabilities from transient states to absorbing states. n_t and n_a are the number of transient and absorbing states, respectively, and the zero and identity matrices $\mathbf{0}_{n_a \times n_t}$ and $\mathbf{I}_{n_a \times n_a}$ imply that the process cannot leave the absorbing state. Our goal is to find the equilibrium state $\mathbf{u}^t$ as $t \to \infty$ for a given initial state $\mathbf{u}^0$. Accordingly, $\mathbf{u}^\infty = \mathbf{u}^0 \mathbf{P}^m$ as $m \to \infty$. The closed-form solution of $\mathbf{P}^m$ as $m \to \infty$ is treated as $\mathbf{P}^\infty$, where

$$\mathbf{P}^\infty = \begin{bmatrix} \mathbf{0}_{n_t \times n_t} & (\mathbf{I} - \mathbf{T})^{-1}\mathbf{A} \\ \mathbf{0}_{n_a \times n_t} & \mathbf{I}_{n_a \times n_a} \end{bmatrix}. \tag{4.4}$$

Using this formulation, the equilibrium state probabilities can only be distributed among the absorbing states with zero probabilities on the transient states.

The initial state $\mathbf{u}^0$ is calculated using the Euclidean distances of the test sample to all the base- and novel-class prototypes and normalizing it to obtain a probability vector. However, we need to decide how to split all the classes into transient and absorbing states so that correct class predictions of the test sample can be obtained. We propose to use a three-step approach to decide the class prediction of a test sample. In the first step, we choose the novel categories and base categories as transient and absorbing states, respectively. Using the absorbing Markov-chain formulation in Eq. (4.3), we obtain the most probable base category from $\mathbf{u}^\infty$. Similarly, for the second step, we choose the novel categories and base categories as absorbing and transient states, respectively, and then obtain the most probable novel category. In the final step, we apply one nearest neighbor on the test sample to choose the most probable class among the most probable base and novel categories obtained in the previous steps. The overall procedure from the novel-class prototype estimation to the Markov-chain-based prediction for a test sample is given in *Algorithm 7*. In case we have multiple samples for a novel class, $\mathbf{x}_n$ is set as the mean of these samples. The time and space complexity of this manifold-based approach is analyzed in Appendix B.

Till now, we have described our non-parametric approach to tackle the limited information few-shot learning setting. However, the performance of a non-parametric

Algorithm 7: Proposed two-step few-shot learning procedure using manifolds.

Given: Base category prototypes $\mathbf{C} \in \mathbb{R}^{n_b \times d}$, Novel class one-shot samples $\mathbf{x}_n, n \in \{1, 2, ..., n_{nov}\}$ where n_{nov} is the number of novel categories. Test sample $\mathbf{x}_{te}$.

Parameters: $r, q, k', \alpha_1, \alpha_2$

Goal: Classify $\mathbf{x}_{\text{te}}$ into one of the $n_b + n_{nov}$ categories

Step 1 *Estimate class prototype for each novel class*

for each novel class $n \in \{1, 2, ..., n_{nov}\}$

Obtain r nearest base prototypes for $\mathbf{x}_n$ to form $\mathbf{C}_r \in \mathbb{R}^{r \times d}$

Obtain q nearest base prototypes for each of the r base prototypes

Obtain orthonormal subspaces $\mathcal{S}_i^{\perp}$ for $i \in \{1, 2, ..., r\}$ using the q neighbors

$\bar{\mathcal{S}} \leftarrow \texttt{ExtrinsicManifoldMean}(\mathcal{S}_1^{\perp}, \mathcal{S}_2^{\perp}, ..., \mathcal{S}_r^{\perp})$

Project $\mathbf{x}_n$ onto $\bar{\mathcal{S}}$ to obtain $\mathbf{c}_p$ followed by distance averaging prototypes in $\mathbf{C}_r$ to obtain $\mathbf{c}_d$

Obtain novel-class prototypes as $\mathbf{c}_n \leftarrow \alpha_2(\alpha_1 \mathbf{x}_n + (1 - \alpha_1)\mathbf{c}_p) + (1 - \alpha_2)\mathbf{c}_d$

end for

Result *Novel class prototypes estimated*

Step 2 *Predict class of test sample* $\mathbf{x}_{te}$

Construct k'-nearest-neighbor graph with n_b base prototypes and n_{nov} novel prototypes as nodes.

Find initial probability vector $\mathbf{u}_0$ using distance of $\mathbf{x}_{te}$ to all the class prototypes.

Construct Markov chain and obtain most probable base class.

Construct Markov chain and obtain most probable novel class.

Use nearest neighbor to obtain the most probable class among the most probable base and novel class.

Result *Class prediction of test sample obtained*

approach may not be as competitive as a parametric approach. Therefore, it is worthwhile to compare our non-parametric approach against a parametric baseline of which the Bayesian framework seems to be a reasonable choice. This is because the Bayesian

approach involves less number of parameters compared to a neural-network-based approach and it is less likely to overfit in the limited-information setting. Accordingly, we describe the Bayesian approach in the next subsection.

4.2.3 Proposed Bayesian Approach

In this section, we describe our proposed Bayesian-based approach. For this Bayesian-based framework, we fix the likelihood as Gaussian because of the assumption that class-data distributions can be safely fitted using a Gaussian-model family. We vary the prior distribution of the novel-class statistics (mean and variance) and accordingly obtain different posterior estimates of the novel-class mean. The mean is chosen to have a normal prior because the class prototypes are assumed to be sampled from a normal distribution. For the variance, we keep it fixed or experiment with different priors that produce known posterior distributions. Accordingly, we obtain four model-variants of the Bayesian formulation. Each of these models are described below.

Normal prior on Mean but fixed Variance

Let us consider that the dataset $\mathbf{X} = \{\mathbf{x}_{n1}, \mathbf{x}_{n2}, ..., \mathbf{x}_{nk}\}$ is generated from a normally distributed novel class with unknown mean $\boldsymbol{\mu}$ and unknown variance σ^2. We want to find the maximum-a-posterior estimate of the class given a normally distributed prior. The description of the Bayesian model is as follows

$$\mathbf{x}_{n1}, \mathbf{x}_{n2}, ..., \mathbf{x}_{nk} \sim \mathcal{N}(\cdot|\boldsymbol{\mu}, \sigma^2), \quad \boldsymbol{\mu} \sim \mathcal{N}(\cdot|\boldsymbol{\mu}_0, \sigma_0^2) \tag{4.5}$$

where $\boldsymbol{\mu}_0$ is set as the empirical mean of the base-class prototypes such that $\boldsymbol{\mu}_0 = \frac{1}{n_b}\sum_{j=1}^{n_b} \mathbf{c}_j$ and $\mathbf{c}_j$'s are the base-class prototypes. Similarly, σ_0^2 is also found from the variance of the base-class prototypes with $\sigma_0^2 = \frac{1}{n_b}\sum_{j=1}^{n_b} ||\mathbf{c}_j - \boldsymbol{\mu}_0||_2^2$. The variance σ^2 of the novel class is unknown but can be set heuristically using the pairwise distances of the base-class prototypes. Let $\mathbf{p} = (p_1, p_2, ..., p_l)$ such that $l = \binom{n_b}{2}$. $\binom{n_b}{2}$ is the

number of combinations in choosing pairs from n_b items. $\mathbf{p}$ is the list of the squared half-pairwise distances of the base-class prototypes. Accordingly, σ^2 can be set as the minimum, maximum, median or mean of the list $\mathbf{p}$. The posterior density can be described as

$$\underbrace{p(\boldsymbol{\mu}|\mathbf{X})}_{\text{Posterior}} \propto \underbrace{p(\mathbf{X}|\boldsymbol{\mu},\sigma^2)}_{\text{Likelihood}} \underbrace{p(\boldsymbol{\mu}|\boldsymbol{\mu}_0,\sigma_0^2)}_{\text{Prior}} = \tag{4.6}$$

$$\prod_{i=1}^{k} p(\mathbf{x}_{ni}|\boldsymbol{\mu},\sigma^2)p(\boldsymbol{\mu}|\boldsymbol{\mu}_0,\sigma_0^2) = \prod_{i=1}^{k} \mathcal{N}(\mathbf{x}_{ni}|\boldsymbol{\mu},\sigma^2)\mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_0,\sigma_0^2).$$

The posterior density has a closed-form solution of a normal distribution with mean $\boldsymbol{\mu}_{post}$ as

$$\boldsymbol{\mu}_{post} = \frac{\sum_{i=1}^{k} \mathbf{x}_{ni}\sigma_0^2 + \boldsymbol{\mu}_0\sigma^2}{k\sigma_0^2 + \sigma^2}. \tag{4.7}$$

Since the mode of a normal distribution is the same as its mean, we can set the maximum-a-posterior estimate of $\boldsymbol{\mu}$ as $\boldsymbol{\mu}_{post}$. Consequently, the novel-class prototype estimate $\mathbf{c}_n$ can be set as $\boldsymbol{\mu}_{post}$ and the steps can be repeated for all the novel classes. We call this Bayesian model as B1 and set the maximum heuristic as the default heuristic unless explicitly mentioned.

Normal prior on Mean, Gamma/Uniform prior on Precision

In this model, we go a step forward and assume gamma prior on the precision $\lambda = \sigma^{-2}$. Accordingly, the Bayesian model can be described as

$$\mathbf{x}_{n1}, \mathbf{x}_{n2}, ..., \mathbf{x}_{nk} \sim \mathcal{N}(\cdot|\boldsymbol{\mu},\lambda^{-1}), \quad \boldsymbol{\mu} \sim \mathcal{N}(\cdot|\boldsymbol{\mu}_0,\sigma_0^2) \tag{4.8}$$

$$\lambda \sim Ga(\cdot|\alpha,\beta)$$

and $\boldsymbol{\mu}_0$ and σ_0^2 are the same as in B1. $Ga(\cdot)$ is the gamma distribution. Hence, the posterior distribution is

$$\underbrace{p(\boldsymbol{\mu},\lambda|\mathbf{X})}_{\text{Posterior}} \propto \underbrace{p(\mathbf{X}|\boldsymbol{\mu},\lambda^{-1})}_{\text{Likelihood}} \underbrace{p(\boldsymbol{\mu}|\boldsymbol{\mu}_0,\sigma_0^2)}_{\text{Prior}} \underbrace{p(\lambda)}_{\text{Prior}} \tag{4.9}$$

$$= \prod_{i=1}^{k} \mathcal{N}(\mathbf{x}_{ni}|\boldsymbol{\mu},\lambda^{-1})\mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_0,\sigma_0^2)Ga(\lambda|\alpha,\beta).$$

Since we do not have a closed-form solution of the posterior distribution, we could use the Gibbs-sampling approach to generate samples ($\boldsymbol{\mu}$, λ).

However, the Gibbs-sampling approach is computationally demanding and therefore not scalable to higher dimensions or larger number of categories. Therefore, we experiment with variational Bayes approximation, where we assume that the posterior distribution of the novel-class mean and variance has a factorization of the form $p(\boldsymbol{\mu}, \lambda) = p_1(\boldsymbol{\mu})p_2(\lambda)$. We set $p_1(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_V, \lambda_V)$ and $p_2(\lambda) = Ga(\lambda|\alpha_V, \beta_V)$ and using alternating optimization technique, we obtain the following expressions

$$\boldsymbol{\mu}_V = \frac{\sum_{i=1}^{k} \mathbf{x}_{ni} \frac{\alpha_V}{\beta_V} + \boldsymbol{\mu}_0 \lambda_0}{k\frac{\alpha_V}{\beta_V} + \lambda_0}, \quad \lambda_V = k\frac{\alpha_V}{\beta_V} + \lambda_0 \tag{4.10}$$

$$\alpha_V = \alpha + \frac{dk}{2} \tag{4.11}$$

$$\beta_V = \beta + \frac{1}{2}(\sum_{i=1}^{k} \mathbf{x}_{ni}^T(\mathbf{x}_{ni} - 2\boldsymbol{\mu}_V) + k(\frac{d}{\lambda_V} + \boldsymbol{\mu}_V^T\boldsymbol{\mu}_V)). \tag{4.12}$$

The derivation of these circularly dependent equations is given in Appendix C. Because of the circular dependency among the variables $\boldsymbol{\mu}_V, \lambda_V, \alpha_V$ and β_V, we need to solve the expressions alternately for a fixed number of iterations $t = 5$. After that, $\boldsymbol{\mu}_V$ is set as the prototype location of the new class. In case we use a uniform prior for λ, we just need to set $\alpha = 1$ and $\beta = 0$. We call this model with uniform prior as B2 and the one with gamma prior as B3.

Normal-Gamma prior on Mean and Precision

In this model, the precision $\lambda_0 = \sigma_0^{-2}$ of the normal prior of the mean is a scaled factor of the novel-class distribution precision $\lambda = \sigma^{-2}$. If this scaled factor is s, then the Bayesian model can be described as

$$\mathbf{x}_{n1}, ..., \mathbf{x}_{nk} \sim \mathcal{N}(\cdot|\boldsymbol{\mu}, \sigma^2), \quad (\boldsymbol{\mu}, \lambda) \sim NG(\cdot|\boldsymbol{\mu}_0, s, \alpha, \beta), \tag{4.13}$$

where $NG(\cdot|\boldsymbol{\mu}_0, s, \alpha, \beta)$ is a normal-gamma distribution with parameters $\boldsymbol{\mu}_0, s, \alpha$ and β. The probability density function of a normal-gamma distribution is the product of the normal ($\mathcal{N}(\cdot)$) and the gamma ($Ga(\cdot)$) density function. It is described as

$$NG(\boldsymbol{\mu}, \lambda|\boldsymbol{\mu}_0, s, \alpha, \beta) \equiv \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_0, (s\lambda)^{-1})Ga(\lambda|\alpha, \beta). \tag{4.14}$$

We are concerned about the mode of the posterior distribution. The posterior distribution can be written as

$$\begin{aligned} \underbrace{p(\boldsymbol{\mu}, \lambda|\mathbf{X})}_{\text{Posterior}} &\propto \underbrace{p(\mathbf{X}|\boldsymbol{\mu}, \lambda^{-1})}_{\text{Likelihood}} \underbrace{p(\boldsymbol{\mu}, \lambda|\boldsymbol{\mu}_0, s, \alpha, \beta)}_{\text{Prior}} \\ &= \prod_{i=1}^{k} \mathcal{N}(\mathbf{x}_{ni}|\boldsymbol{\mu}, \lambda^{-1})\mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_0, (s\lambda)^{-1})Ga(\lambda|\alpha, \beta). \end{aligned} \tag{4.15}$$

The posterior distribution can be shown to be following a normal-gamma distribution having a mode $\boldsymbol{\mu}_{post}$ as

$$\boldsymbol{\mu}_{post} = \frac{\sum_{i=1}^{k} \mathbf{x}_{ni} + \boldsymbol{\mu}_0 s}{k + s}. \tag{4.16}$$

After that, the novel-class prototype estimate $\mathbf{c}_n$ is set as $\boldsymbol{\mu}_{post}$ and the steps are repeated for all the novel classes. We call this model B4 and set $s = 1$ unless explicitly mentioned.

4.3 Experiments and Discussions

4.3.1 Implementation Details

To evaluate our proposed non-parametric and parametric approaches, we used two image recognition datasets – ImageNet and CUB-200. Originally, the ImageNet dataset consists of 21K categories of which we used 1000 for our experiments. These 1000 categories are accordingly split into the base and novel classes. The CUB-200 is a fine-grained dataset that consists of 200 categories of different bird species. Of these 200 classes, we used a total of 150 of which 100 are the base classes and 50 are the novel classes. For both datasets, the image features used were the 2048-dimensional ResNet-101 [23] because they have been found to be class-discriminative.

The recognition performance metric used is class-wise averaged accuracy over test samples taken from both base and novel categories. This metric ensures that major classes do not dominate the performance and minor classes containing less number of samples are not ignored. It is noted that performing cross-validation is impossible since we do not have access to data from the base categories to be held out as a validation set. Therefore, results were reported by fixing the hyper-parameters.

For evaluation and comparison purposes, we used the following models:

- **NA** – The No-adaptation baseline, which consists of using just the nearest-neighbor classification on the few-shot sample mean.
- **M1** – The model uses the nearest-neighbor classification on the estimated novel-class prototypes.
- **M2** – The model performs classification on the few-shot sample mean using the manifold distance.
- **M1+M2** – The model uses the manifold distance on the estimated prototypes for classification.
- **B1** – The model uses normal prior on mean but fixed variance.
- **B2** – The model uses normal prior on mean but uniform prior on precision.
- **B3** – The model uses normal prior on mean but gamma prior on precision.
- **B4** – The model uses normal-gamma prior on mean and precision.
- **Oracle** – This model assumes access to novel-class prototypes and it uses nearest-neighbor classification for prediction.

The Bayesian baselines B1, B2, B3 and B4 used the nearest-neighbor approach for classification. For the B2 and B3 implementation, we used the variational Bayes (VB) approximation instead of the Gibbs-sampling approach because empirically we found out that the sampling approach showed similar recognition performance as VB but it

was about 160 times slower. The Oracle baseline is the maximum possible recognition performance that any of these models can achieve. In the next few subsections, we discuss and analyze all the experiments that we have performed.

4.3.2 Effects of varying the number of classes and samples

In this subsection, we study how the recognition performance is affected by the number of categories and the number of samples per category in the base and novel datasets. This experiment is important to understand the sensitivity of our methods to changes in the number of training classes and samples. For training purposes, we used the prototypes of the base categories and the few-shot samples from the novel categories. If we have k training samples in each novel category, we call the setting as k-shot.

For the first set of experiments, we used the ImageNet dataset with 800 base and 200 novel categories and studied the effect of changing the number of shots per novel category. The results were taken over 10 trials and reported in Table 4.1. Here, each trial corresponds to a randomly sampled set of k training samples from each novel category.

From the results, it is seen that M1 improves the recognition performance over the no-adaptation baseline but the difference diminishes as the number of shots increases. This is because for the novel categories, the few-shot mean becomes closer to the prototype location as the number of shots increases. Also, the contribution of M2 over the baseline or over M1 is incremental. This can be attributed to the fact that the ResNet-101 features are not trained using the manifold-based distance and there is a mis-match between the training and testing evaluation measures. The standard error reduces with the increasing number of shots because of reduced variance in the few-shot mean and eventually reduced variance of the estimated prototype over the trials. Also, different Bayesian models performed better than the NA baseline suggesting that including prior information for mean $\boldsymbol{\mu}$ of the novel class would be

Table 4.1. Accuracy results averaged over 10 trials on the ImageNet dataset with 800 base and 200 novel categories as the number of shots per novel category is changed. Standard error is shown in the parentheses. The hyper-parameter setting is $r = 20, q = 20, k' = 3, \alpha_1 = 0.9, \alpha_2 = 0.7$.

	1 shot	2 shot	5 shot	10 shot	20 shot
NA	64.31 (0.05)	67.60 (0.05)	71.09 (0.03)	72.24 (0.02)	72.89 (0.01)
M1	66.58 (0.05)	69.67 (0.05)	71.62 (0.03)	72.31 (0.02)	72.91 (0.01)
M1+M2	**66.91** (0.05)	**69.88** (0.05)	**72.05** (0.03)	**72.72** (0.02)	**72.97** (0.02)
M2	65.21 (0.05)	67.98 (0.05)	71.33 (0.02)	72.07 (0.02)	72.60 (0.01)
B1	66.69 (0.05)	69.15 (0.04)	71.48 (0.03)	72.44 (0.02)	72.91 (0.01)
B2	65.66 (0.05)	68.89 (0.05)	71.19 (0.03)	72.31 (0.02)	72.89 (0.01)
B3	65.64 (0.05)	67.74 (0.05)	71.19 (0.03)	72.31 (0.02)	72.89 (0.01)
B4	66.59 (0.05)	69.12 (0.04)	71.47 (0.03)	72.44 (0.02)	72.92 (0.01)
Oracle	73.30	73.30	73.30	73.30	73.30

helpful. Surprisingly, B1 performed better than other Bayesian model variants even though it used a fixed value of class variance σ^2. Still, the Bayesian models did not perform as good as the M1+M2 method. We repeated the same experiment for the CUB-200 dataset, the results of which are reported in Table 4.2. In this case, we have 100 base and 50 novel categories, all of which are fine-grained. As a result, the recognition performance is poorer compared to ImageNet, even though CUB-200 has lesser number of categories. Still, the observed recognition performance has a pattern similar to that of the ImageNet dataset. However, there is no reduction in the standard error with increasing shots. This can be attributed to larger overlap between the fine-grained classes of CUB-200. Moreover, B4 showed poor performance at higher shots because the contribution of the few-shot samples is not increased as compared to $s = 1$.

Table 4.2. Accuracy results over 10 trials on the CUB-200 dataset with 100 base and 50 novel categories as the number of shots per novel category is changed. The hyper-parameter setting is $r = 20, q = 20, k' = 5, \alpha_1 = 0.5, \alpha_2 = 0.5$.

	1 shot	2 shot	5 shot	10 shot	20 shot
NA	43.40 (0.12)	45.28 (0.13)	51.19 (0.18)	55.70 (0.20)	58.16 (0.16)
M1	45.91 (0.23)	48.80 (0.20)	51.90 (0.16)	55.94 (0.18)	57.63 (0.14)
M1+M2	**46.13** (0.28)	**49.01** (0.22)	52.13 (0.11)	55.57 (0.17)	**58.67** (0.12)
M2	43.81 (0.11)	45.86 (0.15)	51.45 (0.15)	55.91 (0.18)	58.31 (0.14)
B1	44.53 (0.19)	46.38 (0.17)	**52.54** (0.11)	55.86 (0.02)	57.26 (0.11)
B2	44.36 (0.23)	47.33 (0.20)	52.34 (0.17)	55.85 (0.15)	57.35 (0.10)
B3	44.08 (0.21)	46.60 (0.16)	52.27 (0.13)	**56.19** (0.21)	58.17 (0.15)
B4	44.76 (0.21)	46.93 (0.20)	49.89 (0.15)	51.91 (0.18)	52.56 (0.16)
Oracle	60.51	60.51	60.51	60.51	60.51

For the next set of experiments, we considered the performance change on the ImageNet dataset for the 1-shot setting as the numbers of base and novel categories are varied. We considered two such scenarios. In the first case, the total number of categories was fixed at 1000 while the proportion of base categories was changed. This setting considers less number of base categories compared to novel categories and it has rarely been studied in previous work. The results of this setting are reported in Table 4.3.

From the results, it can be seen that M1 improved over NA by a large margin (9 points) especially when the number of base categories was very less (ratio of 0.1). This is alluded to our assumption that all the class prototypes have a structural arrangement on a manifold. Therefore, the use of this structure is especially beneficial in the few-class regime. However, the difference between M1 and NA decreases mainly due to the presence of more base categories and lesser amount of the difficult novel categories for evaluation. Also, the Bayesian models produced a maximum of 10 points improvement over the NA baseline. This suggested that the Bayesian model

Table 4.3. Accuracy results on the ImageNet dataset for the 1-shot setting as the ratio of number of base categories to the total number of categories is changed. (x-b, y-n) implies x base and y novel categories.

	0.1 (100-b, 900-n)	0.2 (200-b,800-n)	0.4 (400-b, 600-n)	0.6 (600-b, 400-n)
NA	38.29 (0.30)	39.59 (0.29)	46.50 (0.18)	55.28 (0.11)
M1	47.70 (0.28)	49.65 (0.30)	54.65 (0.23)	60.71 (0.12)
M1+M2	47.89 (0.28)	**50.33** (0.29)	**55.13** (0.23)	**61.34** (0.11)
M2	38.36 (0.30)	39.68 (0.29)	46.57 (0.19)	55.31 (0.11)
B1	48.41 (0.28)	50.13 (0.28)	54.68 (0.22)	60.51 (0.11)
B2	45.23 (0.31)	47.81 (0.31)	51.78 (0.21)	58.65 (0.12)
B3	45.10 (0.31)	46.46 (0.31)	51.72 (0.22)	58.60 (0.12)
B4	**48.44** (0.26)	50.08 (0.26)	54.55 (0.22)	60.35 (0.11)
Oracle	73.30	73.30	73.30	73.30

was also significant when the number of base categories was less. On average, B1 and B4 produced better results than B2 and B3. This is probably because B2 and B3 provided an approximation of the posterior distribution.

For the second experiment, we considered the setting where the total number of categories was varied but the proportion of the base and novel categories was kept the same at 4:1. The results of this experiment are reported in Table 4.4.

From the results, it can be seen that the upper bound of the recognition performance; that is, the Oracle performance decreases with an increase in the number of categories. This is mainly because classification becomes more difficult as the number of categories increases. As expected, M1 performed better compared to NA and the contribution of M2 was incremental. B1 and B4 also performed better than B2 and B3.

Our proposed few-shot learning setting is new and therefore we do not have previous work to compare with and benchmark against. However, we can study

Table 4.4. Accuracy results on the ImageNet dataset for the 1-shot setting as the total number of categories is changed but keeping the ratio of base categories to novel categories as 4:1.

	50 (40-b, 10-n)	100 (80-b, 20-n)	200 (160-b, 40-n)	500 (400-b, 100-n)
NA	81.76 (0.54)	75.98 (0.34)	70.25 (0.16)	67.43 (0.10)
M1	86.10 (0.45)	79.92 (0.36)	73.50 (0.16)	69.59 (0.08)
M1+M2	**86.43** (0.47)	**80.61** (0.51)	**74.09** (0.34)	70.41 (0.08)
M2	82.52 (0.53)	75.48 (0.39)	70.88 (0.23)	67.44 (0.10)
B1	86.05 (0.48)	79.75 (0.35)	72.98 (0.15)	**74.32** (0.08)
B2	84.47 (0.51)	78.34 (0.36)	72.10 (0.17)	73.37 (0.08)
B3	84.40 (0.52)	78.31 (0.36)	72.09 (0.17)	73.36 (0.09)
B4	85.97 (0.46)	79.62 (0.33)	72.73 (0.15)	74.17 (0.08)
Oracle	92.20	86.76	80.85	76.87

whether our approach can improve existing relevant work. Prototypical networks (ProtoNets) [87] consider the mean of the few-shot samples to represent class prototypes without using any prior information. Therefore, there is a possibility of obtaining the class prototypes using our manifold-based approach and further improving the performance. Accordingly, we tested the contribution of M1 and M2 over prototypical networks on *miniImageNet*, which is a subset of the ImageNet dataset. The results are shown in Table 4.5. In the table, K-way N-shot implies that K novel categories are sampled per testing episode with N samples per category. From the results, it is clear that M1 improved the performance, however M2 declined it. This is mainly because of the discrepancy between the Euclidean distance metric used during training ProtoNets and the manifold-based distance metric used during testing. The Bayesian models also provided an incremental improvement in performance because ProtoNets already created a discriminative space and therefore discrepancy between the sam-

Table 4.5. Few-shot classification accuracies on the miniImageNet dataset averaged over 600 test episodes for different ways and shots. 95% confidence intervals are shown in the parentheses.

	5-way 1-shot	5-way 5-shot	20-way 1-shot	20-way 5-shot
ProtoNet	47.21 (0.69)	63.62 (0.61)	20.51 (0.46)	35.20 (0.59)
ProtoNet+M1	**48.79** (0.51)	**65.67** (0.56)	**21.93** (0.62)	35.66 (0.53)
ProtoNet+M2	41.36 (0.47)	57.48 (0.43)	16.94 (0.57)	31.52 (0.64)
ProtoNet+B1	48.21 (0.49)	65.28 (0.51)	21.23 (0.52)	**36.84** (0.63)
ProtoNet+B2	47.66 (0.54)	64.67 (0.55)	20.94 (0.63)	35.73 (0.51)
ProtoNet+B3	47.45 (0.50)	64.28 (0.53)	21.55 (0.59)	35.49 (0.52)
ProtoNet+B4	48.08 (0.48)	64.56 (0.44)	21.08 (0.58)	36.09 (0.65)

ple mean and the predicted prototype did not change the classification performance much.

To summarize, in most of the cases where we have less number of shots and less number of base categories, the manifold-based approach performed better than the Bayesian approach. This is because non-parametric methods perform better than parametric methods when there is less training data.

4.3.3 Parameter Sensitivity Studies

In this section, we study the effect of hyper-parameters on the recognition performance. We only report results of sensitivity with respect to r, α_1 and α_2 in Fig. 4.6. We found our recognition performance to be negligibly sensitive to q and k'. This suggests that the location of the novel-class prototype estimate only depends on the number of subspaces (r) rather than its dimensionality ($q+1$). Similarly, the Markov-chain-based prediction does not depend on the number of nearest neighbors k' used for connecting the graph. In Fig. 4.6(a), the number of subspaces (r) is varied, keeping

rest of the hyper-parameters the same. This is done for both the ImageNet (denoted as (I)) and the CUB-200 dataset (denoted as (C)). From the plot, it is seen that the

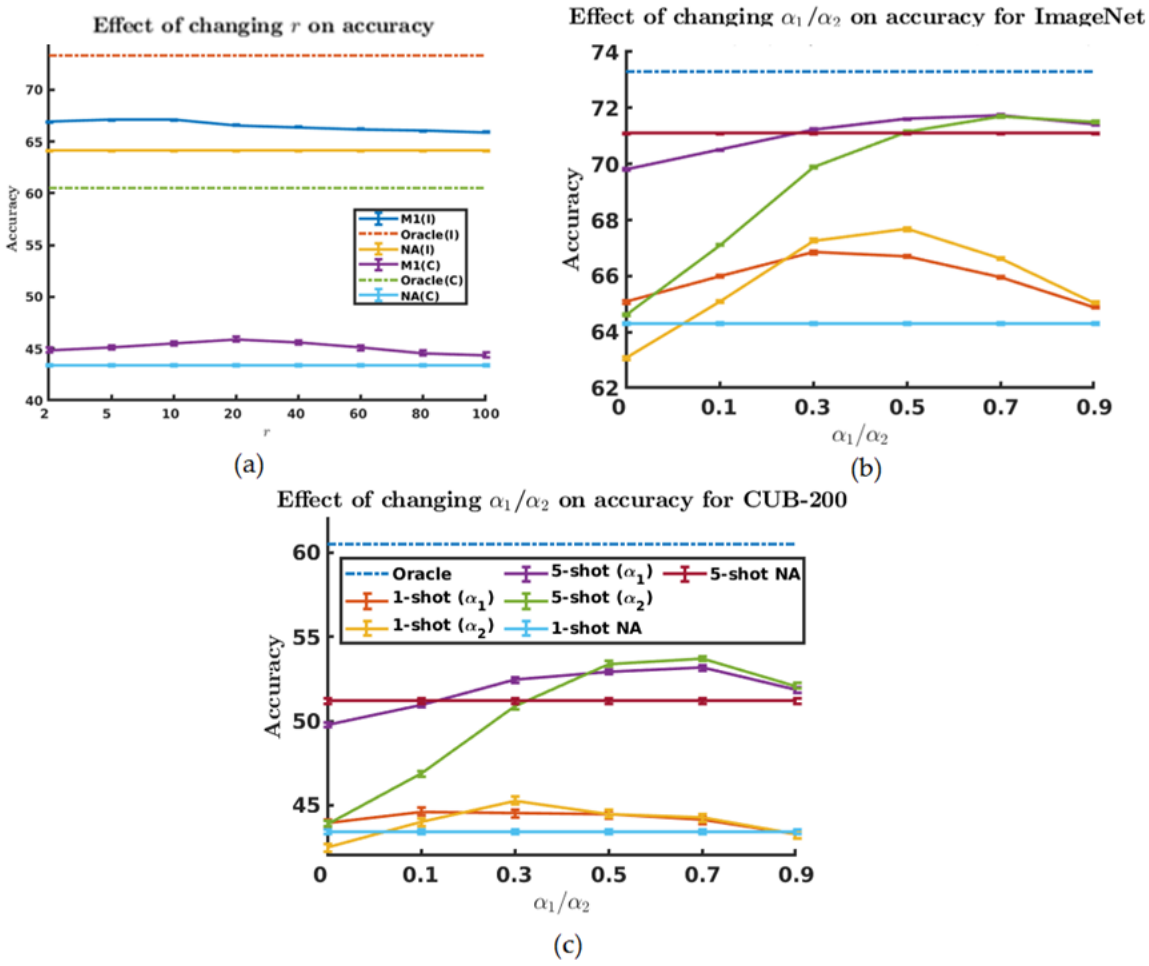


Fig. 4.6. (a) Effect of the number of subspaces r on recognition performance for both ImageNet (I) and CUB-200 (C). Effect of α_1 and α_2 on 1-shot and 5-shot recognition performance for (b) ImageNet and (c) CUB-200. Legends of (c) hold for (b) as well. α_1 in parenthesis suggests that α_1 is varied while $\alpha_2 = 1$ and vice versa. All results are over 10 trials.

performance increases as the number of subspaces increases but then decreases after reaching a peak. Initially, more linear subspaces in the neighborhood are required for estimating the local structure of the non-linear manifold. However, additional irrelevant subspaces cause inaccurate estimation of the manifold resulting in a decrease in performance.

Next, we studied the effects of α_1 and α_2 on the recognition performance for the 1- and 5-shot settings of the ImageNet and the CUB-200 datasets as reported in Figs. 4.6(b) and 4.6(c), respectively. Accordingly, we obtained an optimal performance when α_1 or α_2 is between 0 and 1. From Eq. (4.2), it suggests that the location of the novel-class prototype is within the space bounded by the few-shot sample mean ($\mathbf{x}_n$), the subspace projection ($\mathbf{c}_p$) and the weighted mean of the nearby class prototypes ($\mathbf{c}_d$). For higher number of shots, the maxima seems to move towards the right; that is, closer to α_1, α_2 values of 1. This implies more contribution from the few-shot class mean as visible from Eq. (4.2). This is intuitive because as the number of shots increases, we expect the few-shot sample mean to converge to the class prototype. In fact, for shots of 10 and higher, we obtained the maxima at $\alpha_1 = \alpha_2 = 0.9$ on both datasets. For low values of α_1, α_2 (less contribution of the few-shot sample mean), we observed a dip in performance, even sometimes worse than the NA baseline. This suggests that the contribution of the few-shot mean is important in estimating the novel-class prototype. From the plot, we see that the setting $\alpha_1 = 0$, $\alpha_2 = 1$ produces much better performance as compared to $\alpha_1 = 1$, $\alpha_2 = 0$. According to Eq. (4.2), it means that the contribution of the projection ($\mathbf{c}_p$) is more important compared to contribution of nearby prototypes ($\mathbf{c}_d$).

We also performed additional experiments with B1, where we varied the heuristic procedure to obtain the variance of the novel class σ^2. We chose the minimum, median, mean or maximum of the half-squared pair-wise distance between the base-class prototypes to obtain σ^2. The results of the experiment are shown in Fig. 4.7(a) for both the ImageNet and the CUB-200 datasets. From the experiments, we found out that there was not much difference in choosing the different heuristics. However, the minimum heuristic produced slightly lower performance than the rest for both datasets. This is probably because underestimating the variance rather than overestimating produced a prototype prediction more distant from the ground truth.

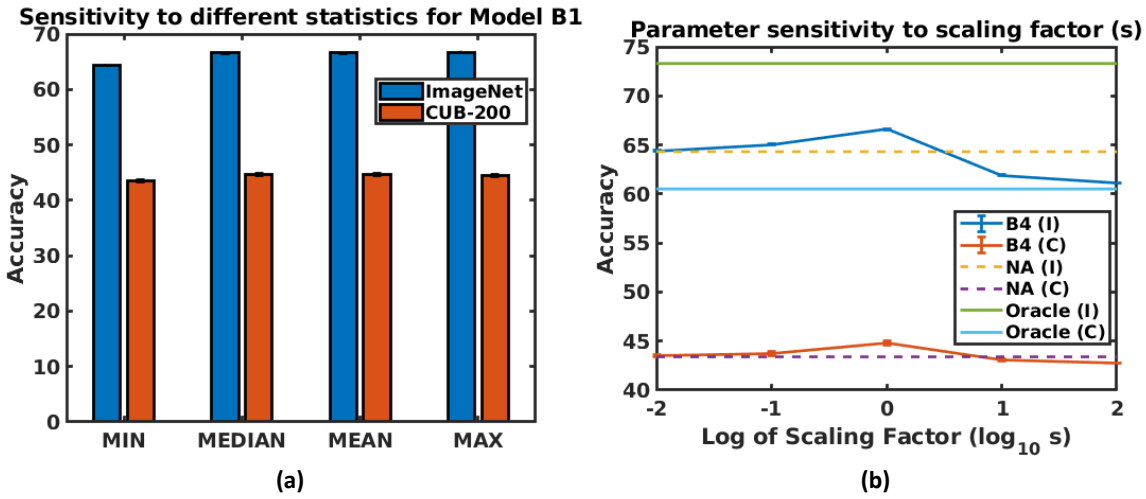


Fig. 4.7. (a) Effect of the min, median, mean and max heuristic on recognition performance of model B1. (b) Effect of hyper-parameter s on recognition performance of model B4 on ImageNet (I) and CUB-200 (C) dataset.

We also tested how the value of the scaling factor s affects the recognition performance of B4. The results of the experiment on both ImageNet and CUB-200 are shown in Fig. 4.7(b). From the plot, it is seen that the recognition performance peaked at $s = 1$ for both datasets. This suggested that B4 performed the best when the variance of the novel class was set equal to the variance of the prior distribution of the novel prototype. However, having a large value of s; that is, having a more peaky prior distribution of the mean reduced the performance, sometimes even lesser than the NA baseline.

4.4 Conclusions

In this Chapter, we have proposed a non-parametric and parametric approach to tackle a new setting in few-shot learning that assumes access to only the base-class prototypes. For the non-parametric approach, we used the structural arrangement of the class prototypes on a manifold, firstly to estimate the novel-class prototypes and secondly to induce an absorbing Markov-chain for test-time prediction. From the experiments, it is evident that our proposed method improved over the no-adaptation

baseline but there is still a lot of room for improvement to reach the Oracle-level performance. Therefore, our results serve as a benchmark for future researchers to work upon. We also found our Markov-chain-based-distance approach to only provide incremental performance improvement mainly because of different distance functions used for learning the features. This motivates us to investigate a differentiable loss function based on the manifold-based distance for training purposes. Further analysis suggests that the novel-class prototype location depends mostly on the few-shot sample mean followed by the projection and then on the location of the nearby prototypes.

As for the parametric Bayesian baseline, it is difficult to conclude what prior is the most effective. However, we found out that using approaches that have closed-form posterior distribution performed better than approximation methods. In most of the cases, the manifold-based approach performs better than the Bayesian method. Also, both the manifold-based and Bayesian-based approaches are the most effective when the number of base categories is much less compared to the number of novel categories. In the future, we would like to investigate an automatic selection mechanism for the hyperparameters and also work with alternative priors and hyper-priors for the Bayesian baseline.

5. EMBEDDING AND GENERATIVE METHODS FOR ZERO-SHOT LEARNING

5.1 Introduction

In this Chapter, we tackle a special case of the few-shot learning (FSL) problem called the zero-shot learning (ZSL) problem. The ZSL problem setting consists of having access to abundant labeled data from the base categories but no labeled data from the novel categories. However, access to side information in the form of semantic descriptors for both the base and the novel categories is available. Figure 5.1 depicts the ZSL problem in terms of how much information is available from both the source and target domains. To solve the ZSL problem, our goal is to relate the feature space and the semantic space so that a test sample can be mapped to a common space to carry out classification. Since we have abundant labeled data in the source domain, we use neural networks to learn the relation. Therefore, neural networks would serve as the structural prior for transferring knowledge to the target domain. Depending on the type of neural networks used, we propose two approaches - the embedding approach and the generative approach.

Previous embedding and generative approaches do not account for the presence of novel categories. For example, Zhang et al. [120] trained a deep neural-network-based embedding to relate the feature space and the semantic space without adapting to novel categories. Also, the method proposed in [139] trained a generative adversarial network exclusively on base categories, with a constraint for strict discrimination between only the base categories. Accordingly, these methods do not generalize well on novel categories. We propose to tackle generalization issues on novel categories through the use of constraints and post-processing steps as depicted in Fig. 5.2. The constraints make sure that the embedding or the generation process is learned

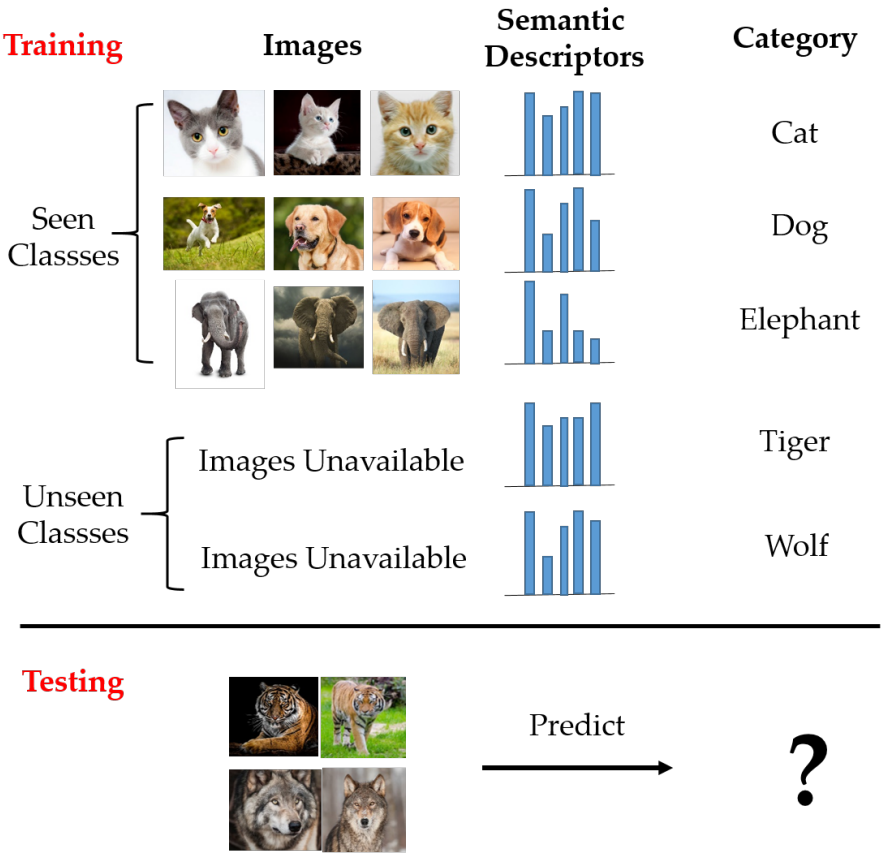


Fig. 5.1. Depiction of the zero-shot recognition problem. During training, we have lots of labeled images from seen classes (cat, dog, elephant) but no labeled images from unseen classes. We do have semantic descriptors of all the classes available. Using all the information, the goal is to recognize the unseen classes.

such that the model can discriminate between seen and unseen classes and biasness towards seen-classes is removed. This improves generalization performance on novel categories. To further improve the performance, the post-processing steps are used so that our model gets adapted to the unlabeled test data from the novel categories. This post-processing step includes domain adaptation or a calibration mechanism to bias predictions towards unseen classes.

The embedding-based approach tackles three major problems in ZSL - hubness, domain-discrepancy and seen-class biasness using a three-step approach. Firstly, a neural-network-based mapping is learned from the semantic-descriptor space to the

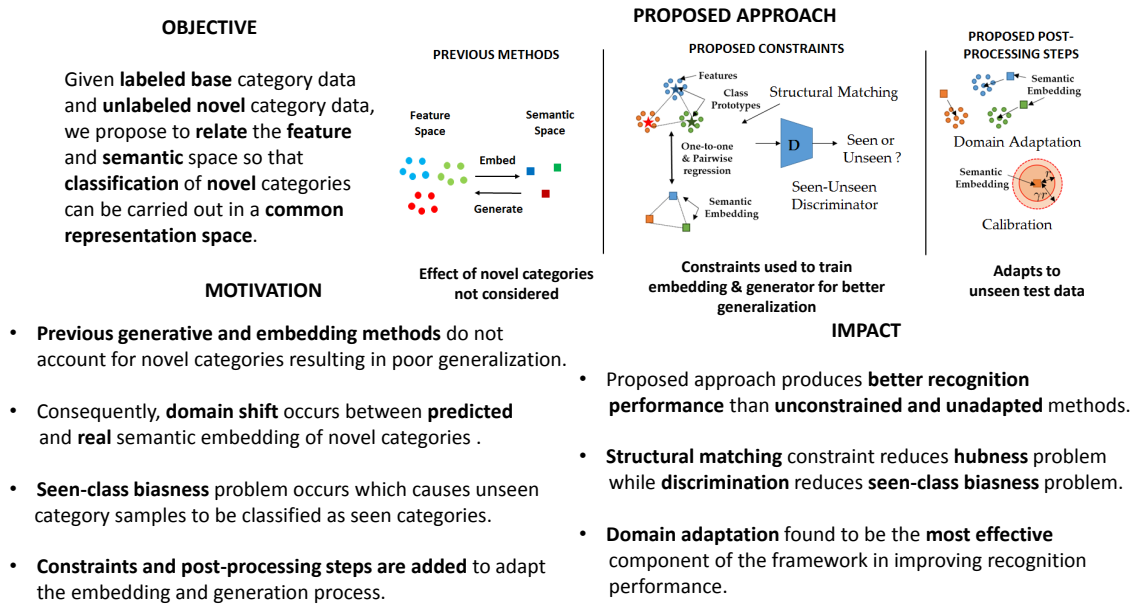


Fig. 5.2. Overall outline of our proposed approach along with motivations and impacts.

feature space. This mapping learns to minimize both one-to-one and pairwise distances between semantic embeddings and the image features of the corresponding classes. Secondly, we propose a test-time domain adaptation step to adapt the semantic embedding of the unseen classes to the test data. This is achieved by finding correspondences between the semantic descriptors and the image features. Thirdly, we propose scaled calibration on the classification scores of the seen classes. This is necessary because the ZSL model is biased towards seen classes as the unseen classes are not used in the training. The results of this embedding-based approach have been published in [155].

On the other hand, the generative approaches convert the ZSL problem into a supervised learning problem by generating data for the novel categories. Most of these methods use Generative Adversarial Networks (GAN) as the generative model. However, since the GAN is trained on only the base categories, the model prediction is biased towards base categories. Also, the generated data for the novel categories

might not accurately represent the ground truth. To address this problem, we propose a three-way solution. Firstly, we constrain the generation process such that the generated data from the novel classes can be highly discriminated from that of the base classes. This constraint tries to get rid of the biasness problem. Secondly, we enforce semantic consistency by reconstructing the semantic attributes from the generated data. Finally, we selectively adapt and transform the generated data from the novel classes to be close to the ground-truth unlabeled test data.

To evaluate our proposed approaches, we tested on standard datasets for ZSL and found our method to be highly competitive with respect to previous work. We also carried out additional studies to better understand each component of our framework. In the next few sections, we describe the details of each of the proposed approaches.

5.2 Embedding-based Approach

This section consists of the description of the embedding-based approach. Initially, we will describe the ZSL problem. This will be followed by a thorough description of the problem formulation and solution. Eventually, we will discuss the implementation and the experimental results.

5.2.1 Problem Definition

Let the training dataset $\mathcal{D}_{tr}$ consist of N_{tr} samples such that $\mathcal{D}_{tr} = \{(\mathbf{x}_i, \mathbf{a}_i, y_i), i = 1, 2, \ldots, N_{tr}\}$. Here, $\mathbf{x}_i \in \mathbb{R}^{m \times n \times c}$ is an image sample ($m \times n$ is the image size and c is the number of channels) and $\mathbf{a}_i \in \mathbb{R}^s$ is the semantic descriptor of the sample's class. Each semantic descriptor $\mathbf{a}_i$ is uniquely associated with a class label $y_i \in \mathcal{Y}_{tr}$. The goal of ZSL is to predict the class label $y_j \in \mathcal{Y}_{te}$ for the j^{th} test sample $\mathbf{x}_j$. In the traditional ZSL setting, we assume that $\mathcal{Y}_{tr} \cap \mathcal{Y}_{te} = \varnothing$; that is, the seen (training) and the unseen (testing) classes are disjoint. However, in the GZSL setting, both seen and unseen classes can be used for testing; that is, $\mathcal{Y}_{tr} \subset \mathcal{Y}_{te}$. In the training stage, we have the semantic descriptors of both the seen and unseen classes available but

no labeled training data of the unseen classes are available. The overall framework of our proposed ZSL approach is shown in Fig. 5.3.

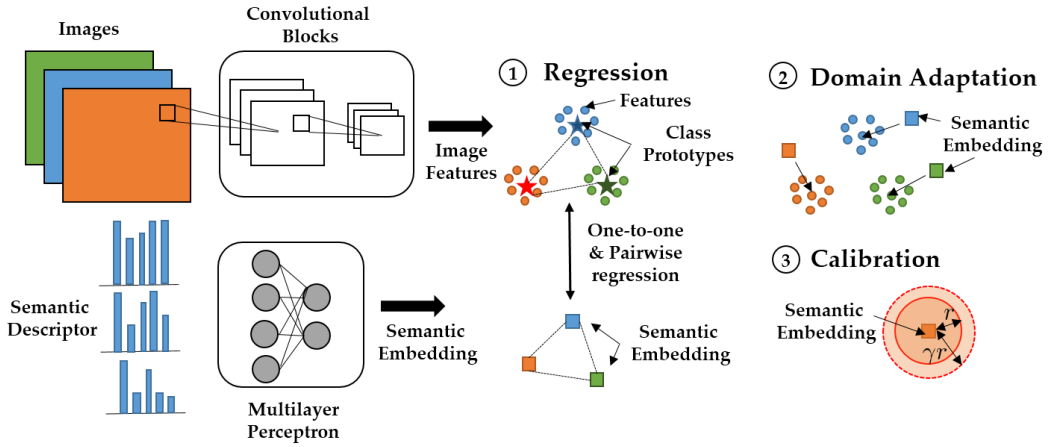


Fig. 5.3. The semantic descriptors are mapped to the image-feature space through the multi-layer perceptron. Then the semantic embeddings are regressed to the corresponding features through one-to-one and pairwise relations. After that, the semantic embeddings of the unseen classes are adapted to the unseen test data. This is followed by scaled calibration during testing when classification scores of seen classes are modified.

5.2.2 Proposed Framework

Our proposed framework consists of three steps - Relational Matching, Domain Adaptation and Scaled Calibration. These steps are described in the following sections.

Relational Matching

Our goal is to learn a mapping $\mathbf{f}(\cdot)$ that maps a semantic descriptor $\mathbf{a}_i$ to its corresponding image feature $\phi(\mathbf{x}_i)$. Here, $\mathbf{x}_i$ is an image and $\phi(\cdot)$ represents a CNN architecture that extracts a high-dimensional feature map. The mapping $\mathbf{f}(\cdot)$ is a fully-connected neural network. Since our goal is to make the embedded semantic descriptor close to the corresponding image feature, we use a least square loss function

to minimize the difference. We also need to regularize the parameters of $\mathbf{f}(\cdot)$. Including these costs and averaging over all the instances, our initial objective function $\mathcal{L}_1$ is as follows:

$$\mathcal{L}_1 = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} ||\mathbf{f}(\mathbf{a}_i) - \phi(\mathbf{x}_i)||_2^2 + \lambda_r g(\mathbf{f}) \ , \tag{5.1}$$

where $g(\cdot)$ is the regularization loss for the mapping function. The loss function $\mathcal{L}_1$ minimizes the point-to-point discrepancy between the semantic descriptors and the image features. To account for the structural matching between the semantic-descriptor space and the image-feature space, we try to minimize the inter-class pairwise relations in these two spaces. Thus, we construct relational matrices for both the semantic descriptors and image features. The semantic relational matrix $\mathbf{D}_a$ is established such that each element, $[\mathbf{D}_a]_{uv} = ||\mathbf{f}(\mathbf{a}^u) - \mathbf{f}(\mathbf{a}^v)||_2^2$, where $\mathbf{a}^u$ and $\mathbf{a}^v$ are semantic descriptors of seen categories u and v, respectively. The image feature relational matrix $\mathbf{D}_\phi$ is constructed such that each element, $[\mathbf{D}_\phi]_{uv} = ||\overline{\phi}^u - \overline{\phi}^v||_2^2$, where $\overline{\phi}^u$ and $\overline{\phi}^v$ are mean representations of the categories u and v, respectively. $\overline{\phi}^u$ can be represented as

$$\overline{\phi}^u = \frac{1}{|\mathcal{Y}_{tr}^u|} \sum_{y_i \in \mathcal{Y}_{tr}^u} \phi(\mathbf{x}_i) \ , \tag{5.2}$$

where the summation is over the representations of class u, and $|\mathcal{Y}_{tr}^u|$ is the cardinality of the training set of class u. A similar formula holds for class v. For structural alignment, we want the two relational matrices, $\mathbf{D}_a$ and $\mathbf{D}_\phi$, to be close to one another. Hence, we want to minimize the structural alignment loss function $\mathcal{L}_2$,

$$\mathcal{L}_2 = ||\mathbf{D}_a - \mathbf{D}_\phi||_F^2 \ , \tag{5.3}$$

where $||\cdot||_F^2$ stands for the Frobenius norm. Combining the loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$, we have the total loss $\mathcal{L}_{total}$,

$$\mathcal{L}_{total} = \mathcal{L}_1 + \rho \mathcal{L}_2 \ , \tag{5.4}$$

where $\rho \geq 0$ weighs the loss contribution of $\mathcal{L}_2$. $\mathcal{L}_{total}$ is to be optimized with respect to the parameters of the semantic-descriptor-to-visual-feature-space mapping $\mathbf{f}(\cdot)$.

Domain Adaptation

After the training is carried out, domain discrepancy may be present between the mapped semantic descriptors and the image features of unseen categories. This is because the unseen data has not been used in the training and our regularized model does not generalize well for the unseen categories. Hence, we need to adapt the mapped semantic descriptors for the unseen categories using the test data from the unseen categories. Let the mapped descriptors for the unseen categories be stacked vertically in the form of a matrix $\mathbf{A} \in \mathbb{R}^{n_u \times d}$, where n_u is the number of unseen categories and d is the dimension of the mapped semantic-descriptor space, and therefore it is also the dimension of the image-feature space. Let $\mathbf{U} \in \mathbb{R}^{o_u \times d}$ be the unseen test dataset, where o_u is the number of test instances from the unseen categories. For adapting the mapped descriptors, we propose to find the point-to-point correspondence between the descriptors and the test data. Let the correspondence be represented as a matrix $\mathbf{C} \in \mathbb{R}^{n_u \times o_u}$. We want to rearrange the rows of $\mathbf{U}$ such that each row of the modified matrix corresponds to the row in $\mathbf{A}$. This is done by minimizing the following loss function $\mathcal{L}_3$,

$$\mathcal{L}_3 = ||\mathbf{CU} - \mathbf{A}||_F^2 \ . \tag{5.5}$$

This loss function enforces that $\mathbf{CU}$ produces the adapted semantic descriptors. However, a problem may exist that an instance in $\mathbf{U}$ corresponds to more than one descriptor in $\mathbf{A}$. This would essentially result in a test sample corresponding to more than one category. To avoid that, we use an additional group-based regularization function $\mathcal{L}_4$ using Group-Lasso,

$$\mathcal{L}_4 = \sum_j \sum_c ||[\mathbf{C}]_{I_c j}||_2 \ , \tag{5.6}$$

where I_c corresponds to the indices of those rows in $\mathbf{A}$ that belong to the unseen class c. Therefore, $[\mathbf{C}]_{I_c j}$ is the vector consisting of the row indices from I_c and the j^{th} column. Since $\mathbf{C}$ is a correspondence matrix, some constraints should be enforced such as $\mathbf{C} \geq \mathbf{0}$, $\mathbf{C}\mathbf{1}_{o_u} = \mathbf{1}_{n_u}$ and $\mathbf{C}^T\mathbf{1}_{n_u} = \frac{n_u}{o_u}\mathbf{1}_{o_u}$, where $\mathbf{1}_n$ is an $n \times 1$ vector of one's.

The second equality constraint is scaled by the factor $\frac{n_u}{o_u}$ to account for the difference in the number of instances in the mapped semantic-descriptor space and the image-feature space for the unseen categories. Hence, the domain adaptation optimization problem becomes

$$\min_{\mathbf{C}} \{\mathcal{L}_3 + \lambda_g \mathcal{L}_4\} \quad s.t. \ \mathbf{C} \geq \mathbf{0}, \mathbf{C}\mathbf{1}_{o_u} = \mathbf{1}_{n_u}, \mathbf{C}^T\mathbf{1}_{n_u} = \frac{n_u}{o_u}\mathbf{1}_{o_u} \ , \tag{5.7}$$

where λ_g weighs the loss function $\mathcal{L}_4$.

The above optimization problem is convex and can be efficiently solved using the conditional gradient method [161]. The conditional gradient method requires solving a linear program as an intermediate step over the constraints $\mathbf{C} \in \mathcal{D} = \{\mathbf{C} : \mathbf{C} \geq \mathbf{0}, \mathbf{C}\mathbf{1}_{o_u} = \mathbf{1}_{n_u}, \mathbf{C}^T\mathbf{1}_{n_u} = \frac{n_u}{o_u}\mathbf{1}_{o_u}\}$ as shown in Algorithm 8. The linear program of finding the intermediate variable $\mathbf{C}_d$ in Algorithm 8 can be easily solved using a network simplex formulation of the earth-mover's distance problem [193].

Algorithm 8: Conditional Gradient Method (CG).

Intitialize : $\mathbf{C}_0 = \frac{1}{(n_u o_u)}\mathbf{1}_{n_u \times o_u}$, $t = 1$

Repeat

$\mathbf{C}_d = \underset{\mathbf{C}}{\text{argmin}} \ \text{Tr}(\nabla_{\mathbf{C}=\mathbf{C}_0}(\mathcal{L}_3 + \lambda_g \mathcal{L}_4)^T\mathbf{C}), \ \ s.t. \ \mathbf{C} \in \mathcal{D}$

$\mathbf{C}_1 = \mathbf{C}_0 + \alpha(\mathbf{C}_d - \mathbf{C}_0), \quad \text{for} \quad \alpha = \frac{2}{t+2}$

$\mathbf{C}_0 = \mathbf{C}_1 \quad \text{and} \quad t = t + 1$

Until Convergence

Output : $\mathbf{C}_0 = \arg\min_{\mathbf{C}} \{\mathcal{L}_3 + \lambda_g \mathcal{L}_4\} \quad s.t. \quad \mathbf{C} \in \mathcal{D}$

Once the final solution of the correspondence matrix $\mathbf{C}_0$ in Algorithm 8 is obtained, we inspect $\mathbf{C}_0$. For each test instance, we assign the class correspondence to the highest value of the correspondence variable. This is done for all the test instances. The new semantic descriptors are obtained by taking the mean of the feature instances belonging to the corresponding class. The adapted semantic descriptors are then stacked vertically in the matrix $\mathbf{A}'$.

Scaled Calibration

In the GZSL setting, it is known that the classification results are biased towards the seen categories [130]. To counteract the bias, we propose the use of multiplicative calibration on the classification scores. In our case, we use 1-Nearest Neighbor (1-NN) with the Euclidean distance metric as the classifier. The classification score for a test point is given by the Euclidean distance of the test image feature to the mapped semantic descriptor of a category. For a test point $\mathbf{x}$, we adjust the classification scores on the seen categories as follows

$$\hat{y} = \underset{c \in \mathcal{T}}{\operatorname{argmin}} \ ||\mathbf{x} - \mathbf{f}(\mathbf{a}^c)||_2 \cdot \mathbb{I}[c \in \mathcal{S}] \ , \tag{5.8}$$

where $\mathbb{I}[\cdot] = \gamma$ if $c \in \mathcal{S}$ and 1 if $c \in \mathcal{U}$ and $\mathcal{S} \cup \mathcal{U} = \mathcal{T}$. Here, $\mathcal{S}$,$\mathcal{U}$ and $\mathcal{T}$ represent the sets of seen, unseen and all categories, respectively. The effect of scaling is to change the effective variance of the seen categories. When the nearest-neighbor classification is carried out with the Euclidean distance metric, it assumes that all classes have equal variance. But since the unseen categories are not used for learning the embedding space, the variance of the unseen-category features is not accounted for. That is why the Euclidean distance metric for the seen categories needs to be adjusted for. For $\gamma > 1$, if we obtain a balanced performance between the seen and unseen classes, it implies that the variance of the seen classes has been overestimated. Similarly, if we obtain a balanced performance for $\gamma < 1$, it means that the variance of the seen classes has been underestimated. The overall procedure of our proposed zero-shot learning method from training to testing is given in Algorithm 9.

5.2.3 Experimental Results

Following the previous experimental settings [148], we used the following four datasets for evaluation: **AwA2** [118] (Animal with Attributes) contains 37,322 images of 50 classes of animals. 40 classes of animals are considered to be the seen categories while 10 classes of animals are considered to be the unseen categories. Each

Algorithm 9: Proposed Zero-shot Learning Algorithm.

Input: Training Dataset $\{(\mathbf{x}_i, \mathbf{a}_i, y_i)\}_{i=1}^{N_{tr}}$

Parameters: λ_r, ρ, λ_g, γ

Repeat (Training)

Sample Minibatch of $\{(\mathbf{x}_i, \mathbf{a}_i)\}$ pairs

Gradient descent $\mathcal{L}_1 + \rho\mathcal{L}_2$ w.r.t parameters of $\mathbf{f}(\cdot)$

Until Convergence

Input: Test Dataset $\{(\mathbf{x}_i)\}_{i=1}^{N_{te}}$

Apply Algorithm 8 to obtain adapted descriptors

of unseen classes $\mathbf{A}'$ (Adaptation)

Repeat for each test point $\mathbf{x}$ (Testing)

$\hat{y} = \underset{c \in \mathcal{T}}{\text{argmin}} \ ||\mathbf{x} - \mathbf{f}(\mathbf{a}^c)||_2 \cdot \mathbb{I}[c \in \mathcal{S}]$ (Calibration)

Until all test points covered

class is associated with a 85-dimensional continuous semantic descriptor. **aPY** [194] (attribute Pascal and Yahoo) consists of 20 seen categories and 12 unseen categories. Each category has an associated 64-dimensional semantic descriptor. **CUB** [182] (Caltech-UCSD Birds-200-2011) is a fine-grained dataset consisting of 11,788 images of birds. For evaluation, all the bird categories are split into 150 seen classes and 50 unseen classes. Each class is associated with a 312-dimensional continuous semantic descriptor. **SUN** [195] (Scene UNderstanding database) consists of 14340 scene images. Among these, 645 scene categories are selected as seen categories while 72 categories are selected as unseen categories and it consists of a 102-dimensional semantic descriptor.

For the purpose of evaluation, we used class-wise accuracy because it prevents dense-sampled classes from dominating the performance. Accordingly, class-wise accuracy is averaged as follows

$$acc = \frac{1}{|\mathcal{Y}|} \sum_{y=1}^{|\mathcal{Y}|} \frac{\text{No. of correct predictions in class } y}{\text{No. of samples in class } y}, \tag{5.9}$$

where $|\mathcal{Y}|$ is the number of testing classes. In the GZSL case, class-wise accuracy of both seen and unseen classes are obtained separately and then averaged using harmonic mean H [148]. This is done so that the performance on seen classes does not dominate the overall accuracy,

$$H = \frac{2 \times acc_s \times acc_u}{acc_s + acc_u}, \tag{5.10}$$

where acc_s and acc_u are the class-wise accuracy on seen and unseen categories, respectively. In the GZSL classification setting, the search space of predicted categories consists of both seen and unseen categories. Based on [148] and for fair comparison, a single trial of experimental results on a large batch of training and testing dataset is reported.

For the experiments, we used a two-layer feedforward neural network for the semantic embedding $\mathbf{f}(\cdot)$. The dimensionality of the hidden layer was chosen as 1600, 1600, 1200 and 1600 for the **AwA2**, **aPY**, **CUB** and **SUN** datasets, respectively. The activation used was ReLU. The image features used were the ResNet-101. We compared different variations of our proposed method with previous approaches. OURS-R variation is with the training stage including the structural loss $\mathcal{L}_2$. OURS-RA includes the structural loss as well as the domain adaptation stage including the loss functions $\mathcal{L}_3$ and $\mathcal{L}_4$. OURS-RC includes the structural loss as well as the calibrated testing stage. OURS-RAC includes all the components of structural loss, domain adaptation and calibrated testing. Without all these components, the proposed method reduces to the Deep Embedding Model (DEM) [120] baseline. The parameters $(\lambda_r, \rho, \lambda_g, \gamma)$ for the **AwA2**, **aPY**, **CUB** and **SUN** datasets are set as $(10^{-3}, 10^{-1}, 10^{-1}, 1.1)$, $(10^{-4}, 10^{-1}, 10^{-1}, 1.1)$, $(10^{-2}, 0, 10^{-1}, 1.1)$ and $(10^{-5}, 10^{-1}, 10^{-1}, 1.1)$, respectively. For the OURS-RAC variation, we used different calibration parameter values of $0.98, 1.1, 0.97, 0.999$ for the **AwA2**, **aPY**, **CUB** and **SUN** datasets, respectively. ρ was set to 0 for the **CUB** dataset because it is a fine-grained dataset and since the categories are very close to each other in the feature space, structural matching does not provide additional information. In Table 5.1, we reported class-wise accuracy results for the conventional unseen classes setting (**tr**), general-

ized unseen classes setting (**u**), generalized seen classes setting (**s**), and the Harmonic mean (**H**) of the generalized accuracies.

Table 5.1. Results of variations of our proposed approach in comparison with previous methods on the **AwA2**, **aPY**, **CUB** and **SUN** datasets. The best results of each setting in each dataset are shown in boldface.

	AwA2				**aPY**				**CUB**				**SUN**			
Method	**tr**	**u**	**s**	**H**	**tr**	**u**	**s**	**H**	**tr**	**u**	**s**	**H**	**tr**	**u**	**s**	**H**
DAP [118]	46.1	0.0	84.7	0.0	33.8	4.8	78.3	9.0	40.0	1.7	67.9	3.3	39.9	4.2	25.1	7.2
IAP [118]	35.9	0.9	87.6	1.8	36.6	5.7	65.6	10.4	24.0	0.2	**72.8**	0.4	19.4	1.0	37.8	1.8
CONSE [137]	44.5	0.5	**90.6**	1.0	26.9	0.0	**91.2**	0.0	34.3	1.6	72.2	3.1	38.8	6.8	39.9	11.6
CMT [123]	37.9	0.5	90.0	1.0	28.0	1.4	85.2	2.8	34.6	7.2	49.8	12.6	39.9	8.1	21.8	11.8
SSE [136]	61.0	8.1	82.5	14.8	34.0	0.2	78.9	0.4	43.9	8.5	46.9	14.4	51.5	2.1	36.4	4.0
LATEM [128]	55.8	11.5	77.3	20.0	35.2	0.1	73.0	0.2	49.3	15.2	57.3	24.0	55.3	14.7	28.8	19.5
ALE [121]	62.5	14.0	81.8	23.9	39.7	4.6	73.7	8.7	54.9	23.7	62.8	34.4	58.1	21.8	33.1	26.3
DEVISE [122]	59.7	17.1	74.7	27.8	**39.8**	4.9	76.9	9.2	52.0	23.8	53.0	32.8	56.5	16.9	27.4	20.9
SJE [127]	61.9	8.0	73.9	14.4	32.9	3.7	55.7	6.9	53.9	23.5	59.2	33.6	53.7	14.7	30.5	19.8
ESZSL [126]	58.6	5.9	77.8	11.0	38.3	2.4	70.1	4.6	53.9	12.6	63.8	21.0	54.5	11.0	27.9	15.8
SYNC [125]	46.6	10.0	90.5	18.0	23.9	7.4	66.3	13.3	55.6	11.5	70.9	19.8	56.3	7.9	**43.3**	13.4
SAE [129]	54.1	1.1	82.2	2.2	8.3	0.4	80.9	0.9	33.3	7.8	54.0	13.6	40.3	8.8	18.0	11.8
GFZSL [196]	63.8	2.5	80.1	4.8	38.4	0.0	83.3	0.0	49.3	0.0	45.7	0.0	60.6	0.0	39.6	0.0
SR [197]	63.8	20.7	73.8	32.3	38.4	13.5	51.4	21.4	**56.0**	24.6	54.3	33.9	61.4	20.8	37.2	26.7
DEM [120]	**67.1**	30.5	86.4	45.1	35.0	11.1	75.1	19.4	51.7	19.6	57.9	29.2	40.3	20.5	34.3	25.6
OURS-R	63.4	36.5	80.6	50.3	29.9	15.3	71.4	25.2	46.6	20.2	48.6	28.6	59.9	21.7	38.1	27.6
OURS-RA	64.4	**61.8**	69.9	65.6	35.4	30.4	72.9	42.9	52.6	**47.6**	41.0	44.1	**67.5**	**54.4**	36.6	**43.7**
OURS-RC	63.4	57.9	72.0	64.2	29.9	26.4	53.3	35.3	46.6	27.2	43.9	33.6	59.9	42.4	32.6	36.8
OURS-RAC	64.4	60.6	72.3	**65.9**	35.4	**34.1**	63.5	**44.4**	52.6	44.0	45.1	**44.6**	**67.5**	54.1	36.6	**43.7**

From the table, we observed that our proposed approach outperforms previous methods by a large margin in the generalized harmonic mean setting. To be more specific, our proposed method produces an improvement of around 20%, 23%, 10% and 16% harmonic mean accuracy over the previous best approach for the **AwA2**, **aPY**, **CUB** and **SUN** datasets, respectively. The large improvement in performance can be attributed to our three-step procedure for improvement. Using only the structural matching (OURS-R), we produced better results than previous approaches except for the **CUB** dataset, where it produces a harmonic mean accuracy of about 28%. This is because **CUB** requires minute fine-grained feature extraction. Addi-

tional usage of domain adaptation (OURS-RA) and calibrated testing (OURS-RC) produced much better results than OURS-R for all the datasets. However, domain adaptation produced better result than the calibration procedure. This is because our correspondence-based approach produced class-specific adaptation of the unseen class semantic embeddings. The scaled-calibration procedure is not class-specific and just differentiates between seen and unseen classes. It also does not adapt to the test data.

It is to be noted that the difference in performance between OURS-RA and OURS-RAC is negligible. This is because the domain adaptation step transforms the unseen semantic embeddings away from the seen categories towards the unseen categories, thus reducing the bias towards the seen categories and rendering further calibration ineffective. The effect of domain adaptation is visualized in Fig. 5.4 for the **AwA2** dataset using t-SNE [174]. In Fig. 5.4(a), the unseen class semantic embeddings (blue) remained very close to the seen class features (maroon). However, with the domain adaptation step, the unseen class semantic embeddings get transformed to near the center of unseen class feature clusters (green) as shown in Fig. 5.4(b).

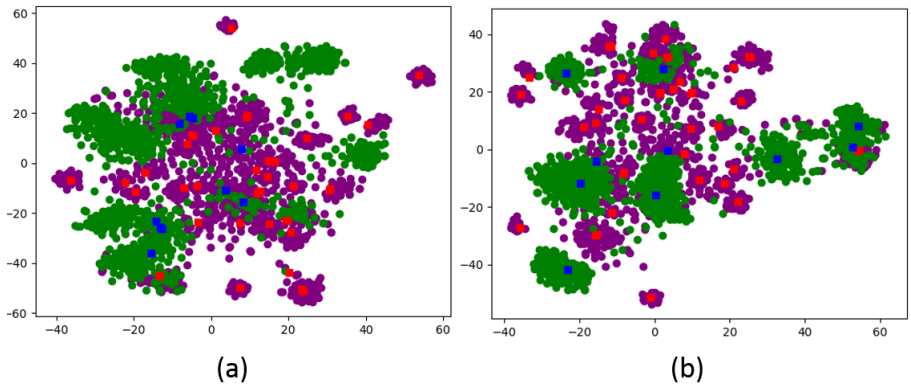


Fig. 5.4. 2D t-SNE map of the embedded instances. (a) Without domain adaptation and (b) with domain adaptation for the **AwA2** dataset. Here, the seen and unseen image features are shown in maroon and green, respectively. The embedded semantic descriptors for the seen and unseen classes are shown in red and blue color, respectively.

We also analyzed the effect of the structural matching by varying ρ and observed how the class-wise accuracy changes. We carried out experiments using the **AwA2** and **SUN** datasets, the results of which are reported in Fig. 5.5. We also reported the DEM baseline ($\rho = 0$) in dotted lines. From the plots, the Conventional Unseen and the Generalized Seen accuracies are better than or equal to the baseline for only a small range of ρ. On the other hand, the Generalized Unseen accuracy is greater than the baseline over a large range of ρ for the **AwA2** dataset while it oscillated about the baseline for the **SUN** dataset. For the **SUN** dataset, we do not have a significant gain over the baseline because **SUN** is a fine-grained dataset where structural matching does not carry additional information. The goal of structural regularization is to exploit the pairwise relations among classes so as to generalize better to novel classes. Therefore, we did not see huge difference in performance from the baseline for the Generalized Seen accuracy. Surprisingly, there was a drop in conventional unseen accuracy as ρ was increased. This might be probably because there was no overlap between the classes used for testing and the classes used for structural matching. This is not the case though in the generalized setting.

We also studied the effect of varying the calibration parameter γ on the generalized accuracy for the **AwA2** and **SUN** datasets. The results are shown in Fig. 5.6. As expected, the generalized unseen accuracy increases and the generalized seen accuracy decreases with increasing γ. The peak of the harmonic mean accuracy was observed close to when the seen and unseen accuracies became equal. The maximum unseen accuracy is less than the maximum seen accuracy for the **AwA2** dataset because the unseen classes are less separated and therefore more difficult to classify. The situation is reversed for the **SUN** dataset where the maximum unseen accuracy is more than the maximum seen accuracy.

We also reported convergence results of the test accuracy with respect to the number of epochs for both the **AwA2** and the **SUN** datasets in Figs. 5.7 and 5.8, respectively. We used the OURS-R variation with $\rho = 0.1$ to compare with the DEM baseline. The convergence rate for the baseline and OURS-R variation seems

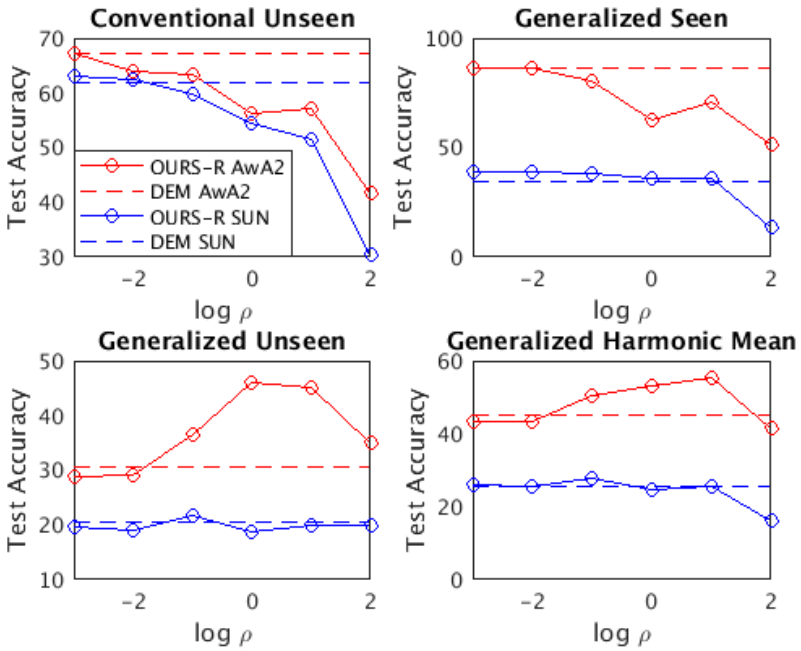


Fig. 5.5. Results of class-wise accuracy as ρ is varied for different settings on the **AwA2** and the **SUN** datasets. The baseline used is DEM. The different performance settings are Conventional Unseen Accuracy (Left Top), Generalized Seen Accuracy (Right Top), Generalized Unseen Accuracy (Left Bottom) and Generalized Harmonic Mean Accuracy (Right Bottom).

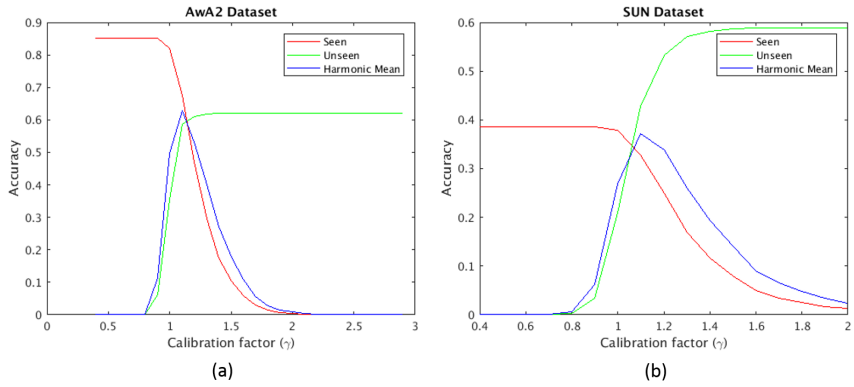


Fig. 5.6. Results of Generalized Seen Accuracy (Red), Generalized Unseen Accuracy (Green) and Generalized Harmonic Mean Accuracy (Blue) as the calibration parameter γ is varied on the **AwA2** and **SUN** datasets.

to be similar in all the settings for both datasets. However, our steady-state values were higher for the generalized unseen and generalized harmonic mean setting. For the conventional unseen and generalized seen setting, our steady-state value was less than the baseline. The reason is explained previously while describing performance sensitivity to ρ.

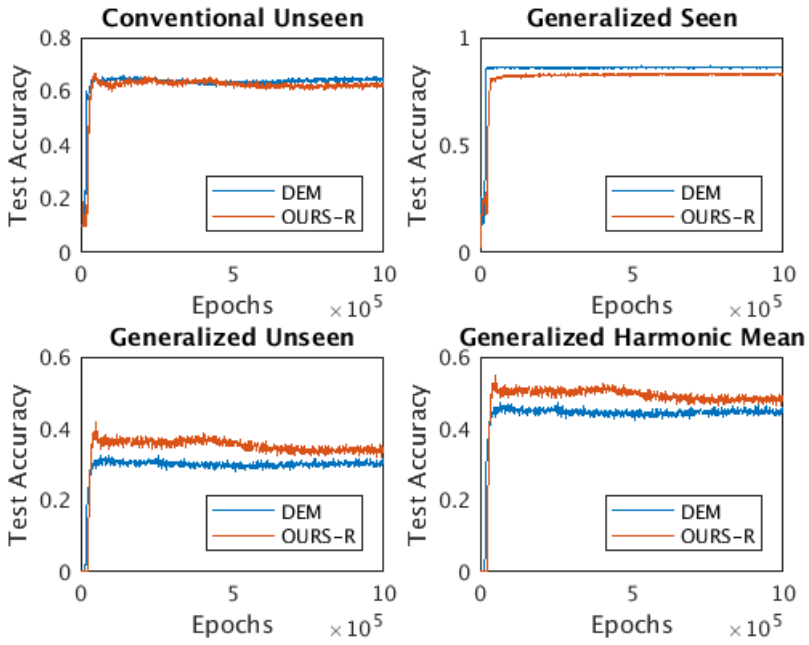


Fig. 5.7. Convergence results of test accuracy with respect to the number of epochs under different settings for the **AwA2** dataset. OURS-R results are shown in red color while the DEM baseline is shown in blue color.

We also studied the effect of varying the number of test unseen samples per class on the generalized harmonic mean accuracy. We used OURS-RA variation of our model for this study. $\rho = 0.1$ was set for the experiments on the **AwA2** (blue color) and the **SUN** (yellow color) datasets and the result was reported in Fig. 5.9. When the fraction is 0.01 for the **SUN** dataset, the number of samples in some classes becomes zero and therefore the performance is not reported. From the results, it is seen that the test accuracy was stable with change in the fraction of total number of samples used for testing. There is a slight increase in accuracy with decreasing number of

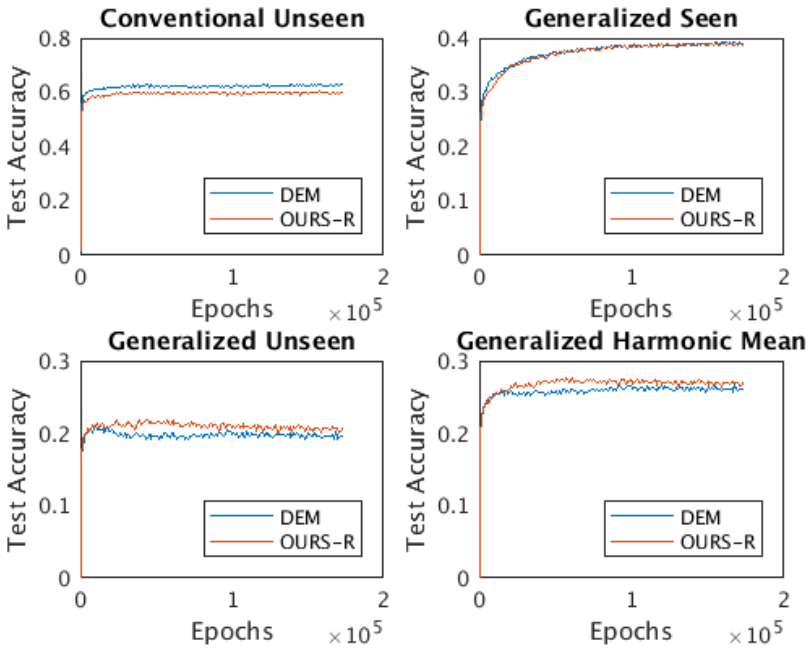


Fig. 5.8. Convergence results of test accuracy with respect to the number of epochs under different settings for the **SUN** dataset. OURS-R results are shown in red color while the DEM baseline is shown in blue color.

samples, which is surprising because domain adaptation would perform poorly with less number of samples. However, this effect is nullified since the probability of including challenging examples is reduced and so we observed a slight improvement in performance.

We also studied how the test performance varies as the number of seen classes for training is reduced for the **AwA2** dataset using OURS-R model. We set $\rho = 0.1$ and reported results over 5 trials in Fig. 5.10. We observed that the change in the seen-class accuracy is not much because the training and testing distributions are the same. The conventional unseen-class accuracy dips by a large amount as the number of training classes decreases because there is less representative information to be transferred to novel categories. However, we obtained a peak for the generalized unseen accuracy results at a fraction of 0.4 of the number of seen classes. This is because as the number of training classes decreases, the amount of representative

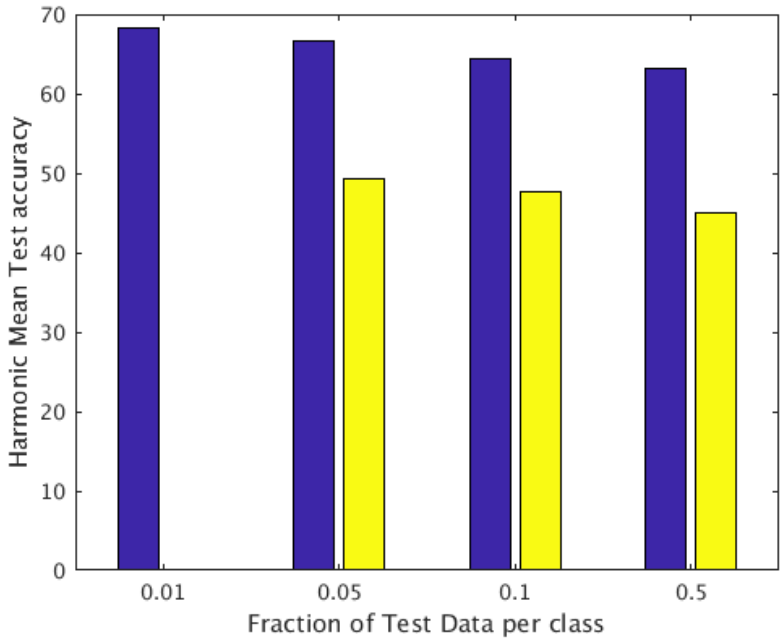


Fig. 5.9. Generalized Harmonic Mean Accuracy results as the number of test samples per class is varied for the **AwA2** (blue) and **SUN** (yellow) datasets.

information decreases, causing decrease in performance. On the other hand, less number of seen classes implies less bias towards seen categories and improvement of unseen-class accuracies. Also, there is large performance variation for unseen-class accuracy because training and testing distributions are different and the performance can vary depending on how related are the training classes to the unseen classes in a trial.

We also performed experiments to find whether the OURS-R variant reduces hubness compared to DEM. The hubness of a set of predictions is measured using the skewness of the 1-Nearest-Neighbor histogram (N_1). The N_1 histogram is a frequency plot for $N_1[i]$ of the number of times a search solution i (in our case a class attribute) is found as the Nearest Neighbor for the test samples. Less skewness of N_1 histogram implies less hubness of the predictions. We used the test samples of the unseen classes in the generalized setting for both DEM and OURS-R on the **AwA2** and the **aPY** datasets. We used $\rho = 0.1$ and reported results averaging over 5 trials in Table 5.2.

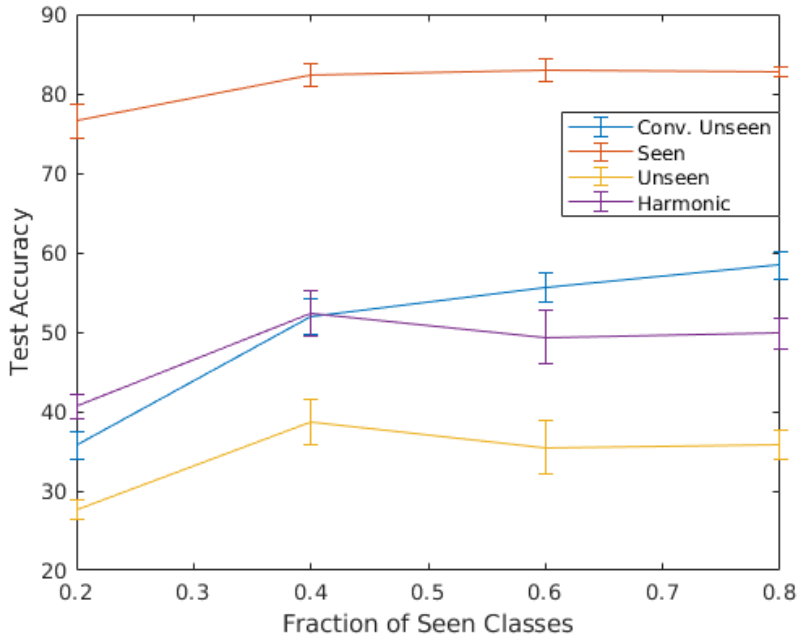


Fig. 5.10. Test accuracy results as the number of seen classes used for training is varied for the **AwA2** dataset.

From the results, OURS-R method produced less skewness of the N_1 histogram on both the datasets. This implies that using the additional structural term reduces hubness and therefore the curse of dimensionality is reduced.

Table 5.2. Hubness comparison using skewness for DEM and OURS-R methods on the **AwA2** and **aPY** datasets.

Skewness	AwA2	aPY
DEM	3.39	1.85
OURS-R	2.41	1.33

5.3 Generative Approach

This section consists of the description of the generative approach. We will redefine the ZSL problem for the sake of convenience. This will be followed by a thorough description of the problem formulation and solution. Eventually, we will discuss the implementation and the experimental results.

5.3.1 Problem Description

The training dataset $\mathcal{D}_{tr}$ contains N_{tr} samples, where $\mathcal{D}_{tr} = \{(\mathbf{I}_i, \mathbf{a}_i, y_i), i = 1, 2, \ldots, N_{tr}\}$. Each $\mathbf{I}_i \in \mathbb{R}^{h \times w \times c}$ is an image where h is the image height, w is the image width and c is the number of channels in the image. $\mathbf{a}_i \in \mathbb{R}^s$ is the semantic descriptor of the sample category, where s is the dimension of the descriptor. $\mathbf{a}_i$ has an unique association with the category label $y_i \in \mathcal{Y}_{tr}$. Our aim is to predict the category $y_j \in \mathcal{Y}_{te}$ for the j^{th} test image $\mathbf{I}_j$. In the traditional ZSL setting, $\mathcal{Y}_{te}$ consists of only the unseen classes. In the generalized ZSL setting, $\mathcal{Y}_{te}$ consists of both seen and unseen classes. We also assume access to the semantic descriptors of both seen and unseen classes all the time.The overall framework of our proposed generative ZSL approach is shown in Fig. 5.11.

5.3.2 Proposed Approach

Background on GANs

GANs [20] were developed to synthesize data that have similar distribution as the real data. They are trained by playing a minimax game between two neural networks - the generator $\mathbf{G}$ and the critic $\mathbf{C}$. Based on a previous study [139], it has been shown that generating image features instead of image pixels provide better performance for the ZSL task. Hence, we resort to generating features instead of pixels in this work. The generator $\mathbf{G}$ uses input as noise $\mathbf{z} \in \mathbb{R}^z$ sampled from a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and then outputs the synthetic features $\tilde{\mathbf{x}} \in \mathbb{R}^d$, where

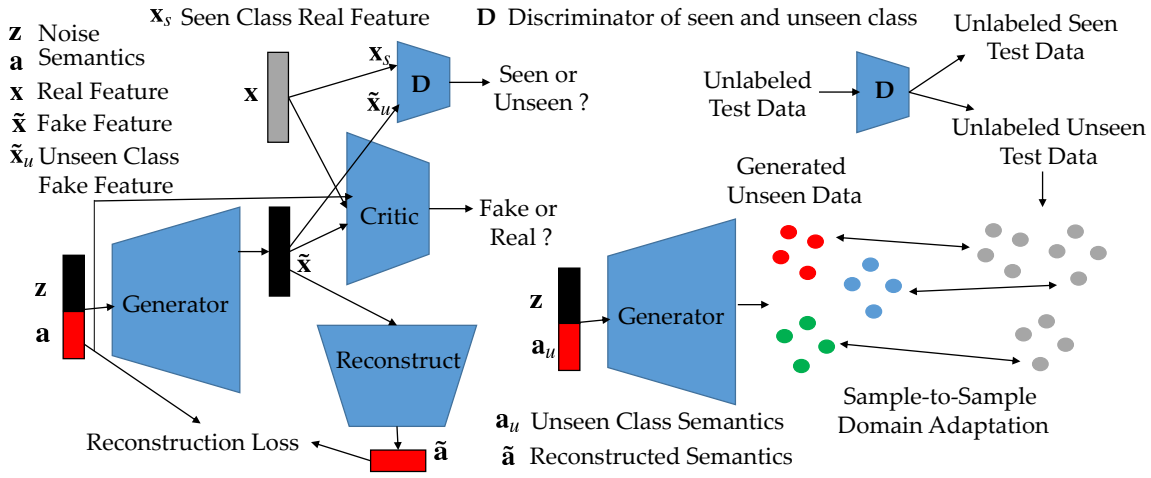


Fig. 5.11. In the proposed framework, the generator is fed with a noise input and a semantic attribute as the conditional information. The generator is trained to synthesize visual features that try to fool the critic from distinguishing real and synthesized data. The discriminator **D** tries to distinguish between real seen features and fake unseen features. The synthesized features are also enforced to reconstruct their corresponding semantic descriptors. After the training is over, the generator is used to synthesize labeled data from the unseen classes. The discriminator **D** is used to separate out unlabeled unseen test data from all of the unlabeled test data. The generated unseen data is then adapted and transformed to the unlabeled unseen test data. The real seen data and this transformed unseen data is used for training a classifier.

z and d are the dimensionality of the noise and the feature space, respectively. The critic **C** then tries to distinguish the synthetic feature $\tilde{\mathbf{x}}$ from the real features $\mathbf{x}$. On the other hand, **G** tries to fool **C** from correctly classifying seen and unseen features. The optimizing loss function used is the Jensen-Shannon (JS) divergence between the real and the synthetic distributions. At the end of the training, **G** produces features indistinguishable from the real features. However, this conventional GAN framework suffers from the vanishing gradient problem especially when the probability distributions do not overlap. As a result, **G** is not able to receive gradient information from the loss function and hence unable to learn properly. To tackle this problem, the Wasserstein Generative Adversarial Network (WGAN) [140] was introduced, which uses the Wasserstein distance [198] instead of the JS Divergence.

It also uses an appropriate 1-Lipschitz continuity constraint [199] on $\mathbf{C}$. This removes the vanishing gradient problem, resulting in more stable training. In our case, we use the conditional WGAN, where the semantic descriptor $\mathbf{a}$ is used in addition to the noise $\mathbf{z}$ as the input to $\mathbf{G}$. This allows $\mathbf{G}$ to generate samples of a particular class whose semantic descriptor is $\mathbf{a}$. Overall, the optimization problem of WGAN is

$$\min_{\mathbf{G}} \max_{\mathbf{C}} \mathcal{L}_{WG} = \mathbb{E}[\mathbf{C}(\mathbf{x})] - \mathbb{E}[\mathbf{C}(\mathbf{G}(\mathbf{z}, \mathbf{a}))] - \lambda_{\mathbf{1}} \mathbb{E}[(||\nabla_{\hat{\mathbf{x}}} \mathbf{C}(\hat{\mathbf{x}})||_{\mathbf{2}} - \mathbf{1})^{\mathbf{2}}], \tag{5.11}$$

where $\lambda_l > 0$ is a hyper-parameter and $\mathbb{E}[\cdot]$ is the expectation operator. ∇ is the gradient operator and $||\cdot||_2$ is the 2-norm. The generator $\mathbf{G}(\mathbf{z}, \mathbf{a})$ synthesizes data with noise $\mathbf{z}$ and the semantic descriptor $\mathbf{a}$ as the input. The regularization $\mathbb{E}[(||\nabla_{\hat{\mathbf{x}}} \mathbf{C}(\hat{\mathbf{x}})||_{\mathbf{2}} - \mathbf{1})^{\mathbf{2}}]$ enforces 1-Lipschitz continuity constraint. Also, $\hat{\mathbf{x}} = \nu \mathbf{x} + (1 - \nu)\mathbf{G}(\mathbf{z}, \mathbf{a})$, where ν is sampled from the uniform distribution $\mathcal{U}[0, 1]$.

Use of Seen-Unseen Class Discriminator

The goal of the discriminator network $\mathbf{D}$ is to distinguish between seen and unseen visual features in order to reduce the seen-class biasness problem. The input to $\mathbf{D}$ will be either real seen features or synthesized unseen features. If the input to $\mathbf{D}$ is $\mathbf{x}$, $\mathbf{D}(\mathbf{x})$ provides two outputs: (a) $\mathbf{D}_s(\mathbf{x})$, that is, the probability of $\mathbf{x}$ belonging to a seen class, and (b) $\mathbf{D}_u(\mathbf{x})$, that is the probability of $\mathbf{x}$ belonging to an unseen class. Obviously, $\mathbf{D}_s(\mathbf{x}) + \mathbf{D}_u(\mathbf{x}) = 1$ to satisfy softmax probability output constraints. To learn the parameters of $\mathbf{D}$, we minimize the cross-entropy classification loss function $\mathcal{L}_D$,

$$\mathcal{L}_D = -\mathbb{E}[\log \mathbf{D}_s(\mathbf{x}_s)] - \mathbb{E}[\log \mathbf{D}_u(\mathbf{G}(\mathbf{z}, \mathbf{a}_u))], \tag{5.12}$$

where $\mathbf{x}_s$ is the real seen class feature and $\mathbf{a}_u$ is the unseen class semantic descriptor. It is important to note that $\mathcal{L}_D$ is minimized with respect to $\mathbf{D}$ only. For optimizing $\mathbf{G}$, we need to synthesize features that can highly discriminate between seen and unseen classes. In other words, $\mathbf{D}$ should produce peaky or low entropy probability outputs. On the other hand, high entropy probability output corresponds to obtaining equal

probabilities of 0.5 for seen and unseen classes. To obtain high entropy probability outputs, we minimize the following cross-entropy loss function $\mathcal{L}_E$,

$$\mathcal{L}_E = -\mathbb{E}[0.5 \log \mathbf{D}_s(\mathbf{G}(\mathbf{z}, \mathbf{a})) + 0.5 \log \mathbf{D}_u(\mathbf{G}(\mathbf{z}, \mathbf{a}))] \tag{5.13}$$

Since our goal is to obtain low entropy outputs, we would maximize $\mathcal{L}_E$ with respect to the parameters of $\mathbf{G}$ only. Enforcing low entropy probability outputs would eventually let $\mathbf{G}$ generate unseen class features whose predictions are not biased towards seen classes.

Reconstructing Semantic Descriptors from Visual Features

The problem with using the seen-unseen class discriminator $\mathbf{D}$ is that it might generate unseen class data that will not be semantically consistent with the ground truth class structural arrangement. As a result, there is a need to reconstruct the generated features back into the semantic descriptors. For that reason, we use the reconstruction network $\mathbf{R}$ as shown in Fig. 5.11. However, we do not jointly learn $\mathbf{G}$ and $\mathbf{R}$ because it might interfere with each of the network's learning process. We pre-train the reconstruction network using the real features from the seen classes and its corresponding semantic attributes. The pre-training can be done by minimizing the loss function $\mathcal{L}_R$,

$$\mathcal{L}_R = \mathbb{E}[||\mathbf{R}(\mathbf{x}) - \mathbf{a}||_2^2]. \tag{5.14}$$

If we just minimize $\mathcal{L}_R$, we can find $\mathbf{R}$ that maps features to semantic descriptors only for seen classes but this mapping would not generalize to unseen classes. We also do not have access to real labeled data from the unseen classes. Hence, we model the pairwise relations between classes and expect this structural relationship of classes to aid in better generalization. To model the structural relation between classes, we construct distance matrices for both the mapped features and the semantics. The semantic distance matrix $\mathbf{D}_a$ is constructed where each element of the matrix $[\mathbf{D}_a]_{ij} = ||\mathbf{a}^i - \mathbf{a}^j||_2^2$. $\mathbf{a}^i$ and $\mathbf{a}^j$ are semantic descriptors of the seen categories i and j, respectively. The mapped visual feature distance matrix $\mathbf{D}_r$ is constructed where

each element of the matrix $[\mathbf{D}_r]_{ij} = ||\overline{\mathbf{r}}^i - \overline{\mathbf{r}}^j||_2^2$. $\overline{\mathbf{r}}^j$ and $\overline{\mathbf{r}}^j$ are mean of the mapped visual features of the seen categories i and j, respectively. $\overline{\mathbf{r}}^i$ can be represented as

$$\overline{\mathbf{r}}^i = \frac{1}{|\mathcal{Y}_s^i|} \sum_{y_k \in \mathcal{Y}_s^i} \mathbf{R}(\mathbf{x}_k) \ , \tag{5.15}$$

where the summation is done for the class i. $|\mathcal{Y}_s^i|$ is the number of training samples in seen class i. We can repeat the same formula for class j. For pairwise structural alignment between the classes, we expect the two distance matrices, $\mathbf{D}_a$ and $\mathbf{D}_r$, to be similar to one another in the Euclidean sense. Hence, we minimize the following loss function $\mathcal{L}_S$,

$$\mathcal{L}_S = ||\mathbf{D}_a - \mathbf{D}_r||_F^2 \ , \tag{5.16}$$

where $|| \cdot ||_F^2$ is the Frobenius norm. Overall, we combine the loss functions $\mathcal{L}_R$ and $\mathcal{L}_S$ to obtain

$$\mathcal{L}_{RS} = \mathcal{L}_R + \lambda_s \mathcal{L}_S \ , \tag{5.17}$$

where $\lambda_s > 0$ weighs the contribution of $\mathcal{L}_S$. Let us assume that $\mathcal{L}_{RS}$ is minimized with respect to $\mathbf{R}$ to obtain $\mathbf{R}^*$. This optimized reconstruction network $\mathbf{R}^*$ is passed into $\mathcal{L}_R$ to obtain $\mathcal{L}_{R^*}$. So, after this pre-training stage is over and during training of $\mathbf{G}$, we minimize the following loss function,

$$\mathcal{L}_{R^*} = \mathbb{E}[||\mathbf{R}^*(\mathbf{G}(\mathbf{z}, \mathbf{a})) - \mathbf{a}||_2^2]. \tag{5.18}$$

The loss function $\mathcal{L}_{R^*}$ is optimized with respect to the parameters of $\mathbf{G}$ and the generated features are sampled from both seen and unseen classes. Thus, we have decoupled the learning of the reconstruction network and the generator and also preserved the semantic consistency of the classes.

Domain Adaptation

After the training of the generator $\mathbf{G}$, the critic $\mathbf{C}$ and the discriminator $\mathbf{D}$ are completed, we then proceed to adapt the generated unseen data with respect to the unlabeled test data. However, the test data can belong to both seen and unseen

classes. Therefore, we reuse the optimized discriminator $\mathbf{D}^*$ to separate out the unseen data from the test data. We select those samples as unseen whose unseen class probability output is greater than a threshold, i.e., $\mathbf{D}_u^*(\mathbf{x}) > t_h$. It is important to note that we allow the threshold to vary, $t_h \in [0, 1]$, instead of fixing $t_h = 0.5$. This is because the discriminator had been calibrated on the generated data instead of the ground-truth test data and $t_h = 0.5$ might not be an appropriate threshold for distinguishing seen and unseen classes. Let the selected unseen class data be $\mathbf{X}_u^s \in \mathbb{R}^{n_s \times d}$ and the generated unseen class data be $\mathbf{X}_u^g \in \mathbb{R}^{n_g \times d}$, where n_s is the number of selected data points and n_g is the number of generated data points. Our goal is to adapt $\mathbf{X}_u^g$ such that it is transformed close to $\mathbf{X}_u^s$. For adapting $\mathbf{X}_u^g$, we use a sample-to-sample correspondence approach inspired from [149]. The correspondence tries to find a matching between the generated samples and selected unseen test data. The correspondence is represented using a matching matrix $\mathbf{M} \in \mathbb{R}^{n_g \times n_s}$. Each row of $\mathbf{M}$ corresponds to a generated sample while each column corresponds to a selected sample. $\mathbf{M}$ operates on $\mathbf{X}_u^s$ and rearranges the rows of $\mathbf{X}_u^s$ such that they are closer to the rows of $\mathbf{X}_u^g$. To find $\mathbf{M}$, we minimize the following loss function, $\mathcal{L}_M$,

$$\mathcal{L}_M = ||\mathbf{M}\mathbf{X}_u^s - \mathbf{X}_u^g||_F^2 \ . \tag{5.19}$$

In addition, we also exploit the labeled information of the generated data by using a Group-Lasso (GL) regularization. This GL regularization enforces a smooth transformation by ensuring that a sample in $\mathbf{X}_u^s$ corresponds to samples of only a single class in $\mathbf{X}_u^g$. Accordingly, the GL regularization is described as

$$\mathcal{L}_{GL} = \sum_j \sum_c ||[\mathbf{M}]_{I_c j}||_2 \ , \tag{5.20}$$

where I_c consists of the row indices in $\mathbf{X}_u^g$ belonging to unseen class c. Accordingly, $[\mathbf{M}]_{I_c j}$ is a list of elements consisting of the row indices of I_c and the j^{th} column. We also enforce constraints on $\mathbf{M}$ such as $\mathbf{M} \geq \mathbf{0}$, $\mathbf{M}\mathbf{1}_{n_s} = \mathbf{1}_{n_g}$ and $\mathbf{M}^T\mathbf{1}_{n_g} = \frac{n_g}{n_s}\mathbf{1}_{n_s}$, where $\mathbf{1}_n$ is an $n \times 1$ vector of one's. The factor $\frac{n_g}{n_s}$ is used to take care of the difference in the

number of samples between $\mathbf{X}_u^s$ and $\mathbf{X}_u^g$. Using these constraints, the optimization problem can be formulated as

$$\begin{aligned} &\min_{\mathbf{M}} \{\mathcal{L}_M + \lambda_{gl}\mathcal{L}_{GL}\} \\ &\text{subject to } \mathbf{M} \geq \mathbf{0}, \mathbf{M}\mathbf{1}_{n_s} = \mathbf{1}_{n_g}, \mathbf{M}^T\mathbf{1}_{n_g} = \frac{n_g}{n_s}\mathbf{1}_{n_s}, \end{aligned} \tag{5.21}$$

where $\lambda_{gl} > 0$ weighs the contribution of the regularization $\mathcal{L}_{GL}$. This convex optimization problem is bounded by simplex constraints and it can be efficiently solved using the Frank-Wolfe algorithm also known as the conditional gradient method [161]. This conditional gradient method iteratively solves a set of linear programs over the same set of constraints as the original problem after which it converges to the solution. Let the constraints of the linear program be denoted as $\mathbf{M} \in \mathcal{M} = \{\mathbf{M} : \mathbf{M} \geq \mathbf{0}, \mathbf{M}\mathbf{1}_{n_s} = \mathbf{1}_{n_g}, \mathbf{M}^T\mathbf{1}_{n_g} = \frac{n_g}{n_s}\mathbf{1}_{n_s}\}$. Accordingly, the linear program can be solved efficiently using a network-simplex approach to the min-flow problem [193]. The iterative method of obtaining $\mathbf{M}$ is described in Algorithm 10.

Algorithm 10: Conditional Gradient (CG) Approach.

Intitialize : $\mathbf{M}_0 = \frac{1}{(n_g n_s)}\mathbf{1}_{n_g \times n_s}$, $t = 1$

Repeat

Use network simplex to solve for $\mathbf{M}_d = \underset{\mathbf{M} \in \mathcal{M}}{\operatorname{argmin}} \ \operatorname{Tr}(\nabla_{\mathbf{M}=\mathbf{M}_0}(\mathcal{L}_M + \ \lambda_{gl}\mathcal{L}_{GL})^T\mathbf{M})$

$\mathbf{M}_I = \mathbf{M}_0 + \alpha(\mathbf{M}_d - \mathbf{M}_0)$, where $\alpha = \frac{2}{t+2}$

$\mathbf{M}_0 = \mathbf{M}_I$ and $t = t + 1$

Until Convergence

Output : $\mathbf{M}^* = \mathbf{M}_0 = \underset{\mathbf{M} \in \mathcal{M}}{\operatorname{argmin}}\{\mathcal{L}_M + \lambda_{gl}\mathcal{L}_{GL}\}$

Using Algorithm 10, we can obtain the solution of the correspondence matrix as $\mathbf{M}^*$. This solution can then be used to transform $\mathbf{X}_u^g$ such that the transformed generated data is $\mathbf{M}^*\mathbf{X}_u^s$. Finally, the transformed generated unseen class data and the real seen class data are used to train a classifier to recognize the test data into one of the base and novel categories.

It is important to note that the domain-adaptation step used in this work extends the domain-adaptation method proposed in our earlier work [155]. In [155], we assumed access to the unseen test data while in our current case, we use the threshold mechanism with the discriminator **D** to select the unseen test data. Also, in [155] we carried out domain adaptation between the semantic embeddings and the unseen test data while in this current work, we carry out domain adaptation between the generated unseen data and the test unseen data. The overall procedure of our proposed framework is shown in Algorithm 11.

5.3.3 Experiments and Discussions

To validate our proposed constrained generative approach for the ZSL and the GZSL settings, we conducted experiments on five standard datasets. These datasets contain a wide variation of fine-grained and coarse-grained categories and hence serve as standard benchmarks for ZSL and GZSL.

Dataset Description

For evaluation purposes, we experimented with the following five datasets: (a) Animals with Attributes (**AwA**) [118], containing 30,475 images of animals distributed across 40 seen and 10 unseen classes. It consists of a 85-dimensional semantic descriptor. (b) Attribute Pascal and Yahoo (**aPY**) [194], which contains 15,339 images from PASCAL VOC 2008 dataset [200] and Yahoo search queries distributed across 20 seen and 12 unseen classes. It consists of a 64-dimensional semantic descriptor. (c) Caltech-UCSD Birds-200 (**CUB**) [182] contains 11,778 images of bird species distributed across 150 seen and 50 unseen classes. It consists of a 312-dimensional semantic descriptor. (d) Scene Understanding (**SUN**) [195] contains 14,340 images distributed across 645 seen and 72 unseen classes. It consists of a 102-dimensional semantic descriptor. (e) Flowers (**FLO**) [201] contains 8,189 images of flower species distributed across 82 seen and 20 unseen categories. It consists of a 1024-dimensional

Algorithm 11: Proposed ZSL framework.

Given: The training dataset $\mathcal{D}_{tr}$ containing N_{tr} samples from base categories where $\mathcal{D}_{tr} = \{(\mathbf{I}_i, \mathbf{a}_i, y_i), i = 1, 2, \ldots, N_{tr}\}$. We also have semantic descriptors $\mathbf{a}_u$ of unseen classes

Hyper-parameters: $\lambda_l, \lambda_s, \lambda_{gl}, \lambda_r, \lambda_e, t_h$

Randomly initialize parameters of reconstructor (**R**), generator (**G**), critic (**C**) and discriminator (**D**)

Step 1: Pre-train reconstruction network **R**

For each epoch

  For each sampled batch

    Take gradient descent step of $\mathcal{L}_{RS}$ with respect to parameters of **R**

  End For

End For

Optimized reconstruction network $\mathbf{R}^*$ obtained.

Step 2: Train generator **G**, critic **C** and discriminator **D**

For each epoch

  For each sampled batch

    Take gradient ascent steps of $\mathcal{L}_{WG}$ with respect to parameters of **C**

    Take gradient descent step of $\mathcal{L}_D$ with respect to parameters of **D**

    Take gradient descent step of $\mathcal{L}_{WG} + \lambda_r \mathcal{L}_{R^*} - \lambda_e \mathcal{L}_E$ with respect to parameters of **G**

  End For

End For

Optimized $\mathbf{C}^*$, $\mathbf{D}^*$ and $\mathbf{G}^*$ obtained.

Step 3: Adapt generated unseen data to test data

Use $\mathbf{G}^*$ to generate data for unseen classes

Select unseen test data using $\mathbf{D}_u^*(\mathbf{x}) > t_h$ thresholding mechanism

Adapt and transform generated unseen data $\mathbf{X}_u^g$ towards selected unseen test data $\mathbf{X}_u^s$ using Algorithm 10

Step 4: Train classifier using the transformed generated unseen data and the real seen data

semantic descriptor. For all these datasets, the image features used were the 2048-dimensional ResNet-101 [23], pre-trained on ImageNet. Example of images from these datasets are shown in Fig. 5.12.

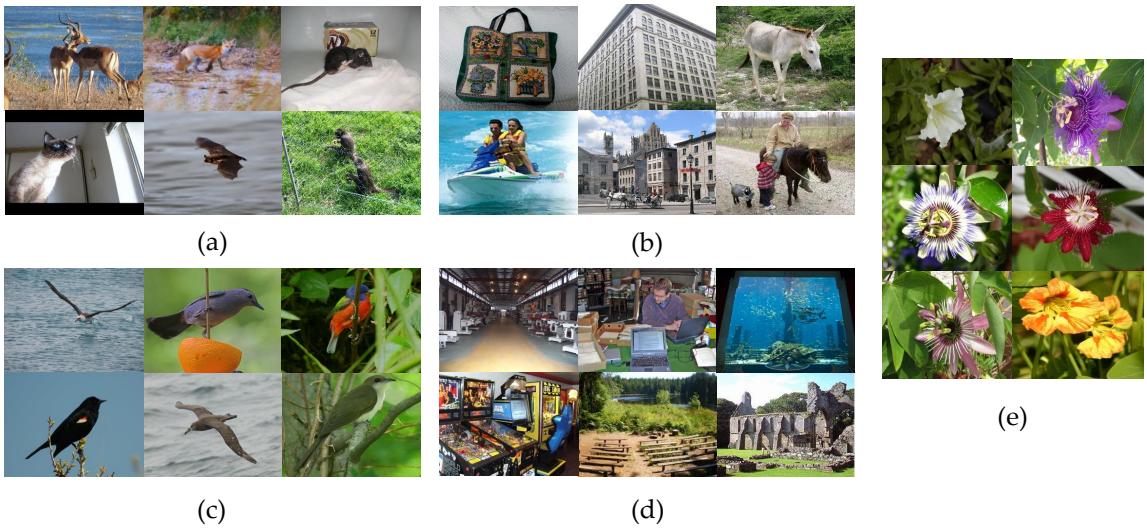


Fig. 5.12. Example images from (a) **AwA**, (b) **aPY**, (c) **CUB**, (d) **SUN** and (e) **FLO** datasets.

Implementation Details

The dimension of the noise input **z** to the generator **G** is kept the same as the dimension of the semantic descriptor. Accordingly, the dimension of **z** is 85, 64, 312, 102 and 1024 for the **AwA**, **aPY**, **CUB**, **SUN** and **FLO** datasets, respectively. The generator **G**, critic **C** and reconstructor **R** all use a two-layer fully connected neural network with 4096 hidden units and ReLU activation function. The discriminator uses single layer softmax outputs with two nodes representing seen and unseen categories. The optimization algorithm used is Adam [179]. The number of epochs used for training on the **AwA**, **aPY**, **CUB**, **SUN** and **FLO** datasets are 30, 40, 56, 40 and 80, respectively. The number of conditional gradient iterations is fixed at 20. The hyper-parameters λ_l, λ_{gl} and λ_s are set as 10, 0.1 and 0.1, respectively. The values

$(\lambda_r, \lambda_e, t_h)$ were set as $(1, 1e-3, 0.5)$, $(1e-4, 10, 0.2)$, $(1, 1e-2, 0.2)$, $(1, 0.1, 0.2)$ and $(1, 1e-4, 0.5)$ for the **AwA**, **aPY**, **CUB**, **SUN** and **FLO** datasets, respectively.

Comparison Studies

For comparison with previous work, we used class-wise accuracy as the performance metric because it prevents categories with larger number of samples to dominate the performance. The class-wise accuracy (acc) can be expressed as

$$acc = \frac{1}{|\mathcal{Y}|} \sum_{y=1}^{|\mathcal{Y}|} \frac{\text{No. of correct predictions for class } y}{\text{No. of samples for class } y} \tag{5.22}$$

where $|\mathcal{Y}|$ is the number of testing classes. In the GZSL setting, we test on both seen and unseen class samples with a label space containing all the categories. As a result, we obtain acc_s and acc_u as the class-wise accuracy on the seen and unseen class test samples, respectively. acc_s and acc_u are averaged to obtain the harmonic mean H such that

$$H = \frac{2 \times acc_s \times acc_u}{acc_s + acc_u}. \tag{5.23}$$

H is therefore the performance metric in the GZSL setting, which balances the class-wise accuracy performance between the seen and unseen categories. In Table 5.3, we compared our method with previous work on the following settings: (a) Traditional unseen class setting (**tr**), which considers acc on only the unseen classes as the label space; (b) Generalized unseen class setting (**u**), which considers acc_u on all the classes as the label space; (c) Generalized seen class setting (**s**), which considers acc_s on all the classes as the label space, and (d) the harmonic mean of acc_u and acc_s (**H**). Among the compared methods, GAZSL [141], CLSWGAN [139] and C-UWGAN [147] used Generative Adversarial Network (GAN) as the backbone for generating features. SE-ZSL [144], CVAE-ZSL [143] and CADA-VAE [146] used Variational Autoencoder (VAE) as the backbone for generating features. GFZSL [196] used Gaussian Mixture Model (GMM) as the generative framework. Rest of the compared methods are non-

generative in the sense that they just try to relate the feature and semantic spaces.

Table 5.3. Results of our proposed approach as compared with previous methods on the **AwA**, **aPY**, **CUB**, **SUN** and **FLO** datasets. The best results are shown in boldface.

	AwA				**aPY**				**CUB**				**SUN**				**FLO**			
Method	**tr**	**u**	**s**	**H**	**tr**	**u**	**s**	**H**	**tr**	**u**	**s**	**H**	**tr**	**u**	**s**	**H**	**tr**	**u**	**s**	**H**
DAP [118]	46.1	0.0	84.7	0.0	33.8	4.8	78.3	9.0	40.0	1.7	67.9	3.3	39.9	4.2	25.1	7.2	-	-	-	-
IAP [118]	35.9	0.9	87.6	1.8	36.6	5.7	65.6	10.4	24.0	0.2	**72.8**	0.4	19.4	1.0	37.8	1.8	-	-	-	-
CONSE [137]	44.5	0.5	**90.6**	1.0	26.9	0.0	**91.2**	0.0	34.3	1.6	72.2	3.1	38.8	6.8	39.9	11.6	-	-	-	-
CMT [123]	37.9	0.5	90.0	1.0	28.0	1.4	85.2	2.8	34.6	7.2	49.8	12.6	39.9	8.1	21.8	11.8	-	-	-	-
SSE [136]	61.0	8.1	82.5	14.8	34.0	0.2	78.9	0.4	43.9	8.5	46.9	14.4	51.5	2.1	36.4	4.0	-	-	-	-
LATEM [128]	55.8	11.5	77.3	20.0	35.2	0.1	73.0	0.2	49.3	15.2	57.3	24.0	55.3	14.7	28.8	19.5	-	-	-	-
ALE [121]	62.5	14.0	81.8	23.9	39.7	4.6	73.7	8.7	54.9	23.7	62.8	34.4	58.1	21.8	33.1	26.3	48.5	13.3	61.6	21.9
DEVISE [122]	59.7	17.1	74.7	27.8	39.8	4.9	76.9	9.2	52.0	23.8	53.0	32.8	56.5	16.9	27.4	20.9	45.9	9.9	44.2	16.2
SJE [127]	61.9	8.0	73.9	14.4	32.9	3.7	55.7	6.9	53.9	23.5	59.2	33.6	53.7	14.7	30.5	19.8	53.4	13.9	47.6	21.5
ESZSL [126]	58.6	5.9	77.8	11.0	38.3	2.4	70.1	4.6	53.9	12.6	63.8	21.0	54.5	11.0	27.9	15.8	51.0	11.4	56.8	19.0
SYNC [125]	46.6	10.0	90.5	18.0	23.9	7.4	66.3	13.3	55.6	11.5	70.9	19.8	56.3	7.9	43.3	13.4	-	-	-	-
SAE [129]	54.1	1.1	82.2	2.2	8.3	0.4	80.9	0.9	33.3	7.8	54.0	13.6	40.3	8.8	18.0	11.8	-	-	-	-
GFZSL [196]	63.8	2.5	80.1	4.8	38.4	0.0	83.3	0.0	49.3	0.0	45.7	0.0	60.6	0.0	39.6	0.0	-	-	-	-
SR [197]	63.8	20.7	73.8	32.3	38.4	13.5	51.4	21.4	56.0	24.6	54.3	33.9	61.4	20.8	37.2	26.7	-	-	-	-
DEM [120]	67.1	30.5	86.4	45.1	35.0	11.1	75.1	19.4	51.7	19.6	57.9	29.2	40.3	20.5	34.3	25.6	-	-	-	-
RAC [155]	64.4	60.6	72.3	65.9	35.4	**34.1**	63.5	44.4	52.6	44.0	45.1	44.6	**67.5**	**54.1**	36.6	**43.7**	-	-	-	-
GAZSL [141]	68.2	19.2	86.5	31.4	41.1	14.2	78.6	24.0	55.8	23.9	60.6	34.3	61.3	21.7	34.5	26.7	60.5	28.1	77.4	41.2
CLSWGAN [139]	68.2	57.9	61.4	59.6	40.5	32.9	61.7	42.9	57.3	43.7	57.7	49.7	60.8	42.6	36.6	39.4	67.2	59.0	73.8	65.6
SE-ZSL [144]	69.2	58.3	68.1	62.8	-	-	-	-	59.6	41.5	53.3	46.7	63.4	40.9	30.5	34.9	-	-	-	-
CVAE-ZSL [143]	65.8	-	-	51.2	-	-	-	-	52.1	-	-	34.5	61.7	-	-	26.7	-	-	-	-
C-UWGAN [147]	66.8	59.6	63.4	59.8	-	-	-	-	58.6	47.9	59.3	53.0	59.9	47.2	33.8	39.4	70.3	61.6	69.2	65.2
CADA-VAE [146]	64.0	**75.0**	55.8	63.9	-	-	-	-	**60.4**	53.5	51.6	52.4	61.8	35.7	**47.2**	40.6	-	-	-	-
OUR METHOD	**73.3**	61.9	81.1	**70.2**	**41.9**	33.7	73.6	**46.2**	59.7	**53.6**	53.1	**53.3**	58.7	51.7	33.9	40.9	**70.7**	**69.4**	**78.6**	**73.7**

tr = Traditional unseen accuracy, **u** = Generalized unseen accuracy, **s** = Generalized seen accuracy, **H** = Harmonic mean of **u** and **s**.

From the results in Table 5.3, our proposed method produces the best harmonic mean accuracy on most of the datasets. Specifically, our proposed method produces 4.3, 1.8, 0.3 and 8.1 point-improvement on the **AwA**, **aPY**, **CUB** and **FLO** datasets, respectively. The small improvement on the **CUB** dataset is mainly because the categories are fine-grained and it is difficult to classify. For similar reasons, our proposed method does not produce the best results on the **CUB** and **SUN** datasets in the traditional setting. The average performance on the **SUN** dataset is the poorest because it contains the largest number of seen and unseen categories for testing from among all the datasets. Among the compared methods, the non-generative methods

produce the worst results in the GZSL setting. This is because most of these methods overfit to the seen categories as evident from their high performance in the generalized seen accuracy setting **s**. For example, CONSE [137] produces the best performance of $\mathbf{s} = 90.6$ and $\mathbf{s} = 91.2$ on the **AwA** and **aPY** datasets, respectively. Among the generative based methods, there is no clear winner between GAN-based and VAE-based methods. However, the GMM-based model GFZSL [196] performed poorly on the GZSL setting. This is because GMMs are not expressive enough to model the data generating distribution of the unseen classes. Also, C-UWGAN [147] used a cycle-consistency network similar to our reconstruction network. Thus, the improvement of our proposed method over C-UWGAN can be attributed to the discriminator network and the domain-adaptation step. Still, we need to figure out the contribution of each component of our proposed approach. We thus conduct ablation analysis in the next sub-section.

Ablation Study

In this sub-section, we carry out ablations to study the contribution of each component to the recognition performance of our framework - R (Reconstruction network), D (Discriminator network) and A (Domain-adaptation step) on top of the WGAN baseline B. The results of the study on the **AwA**, **CUB** and **FLO** datasets are shown in Table 5.4. It is important to note that the discrimination step is a prerequisite for the domain-adaptation step. This is because the domain-adaptation procedure requires the discriminator network for separating out the seen and unseen class features from the unlabeled test data. From the results, it is clear that adding each component on top of one another improves the recognition performance gradually both in the generalized and conventional settings. However, the improvement in performance when using discrimination or reconstruction is around 1-2 points compared to using domain-adaptation where it improves around 2-8 points. This suggests that the domain-adaptation step is the most important component of our framework. The

Table 5.4. Harmonic mean accuracy results of different ablations of our framework in the GZSL setting. Conventional unseen accuracy is shown in brackets. Here, B stands for the WGAN baseline, R stands for the reconstruction network, D stands for the discriminator network and A stands for the domain adaptation step.

Dataset/Ablation	B	BR	BD	BRD	BDA
AwA	59.7	60.7	61.4	63.1	69.3
	(65.9)	(66.1)	(66.3)	(66.4)	(68.6)
CUB	48.8	49.8	49.4	49.9	51.4
	(55.3)	(56.6)	(56.1)	(57.6)	(58.9)
FLO	64.1	65.2	64.9	65.6	69.2
	(64.8)	(64.9)	(65.1)	(65.5)	(68.7)

performance of using just the reconstruction (BR) is greater than using just the discrimination (BD) on the **CUB** dataset but it is the reverse for the **AwA** dataset. This shows that the effect of discrimination or reconstruction depends on the dataset used for comparison. Also, the contribution of our framework over the baseline is more for the **AwA** dataset compared to fine-grained datasets like **CUB** and **FLO**. Similarly, the contribution in the generalized setting is more than that in the conventional setting suggesting that our framework caters to GZSL by considering the contribution of the unseen classes.

Visualization of Generated Features

In this subsection, we visualize the generated features using the t-SNE [174] tool as shown in Figs. 5.13 and 5.14. Since we are concerned about the distributional difference between the generated and the real features of the novel categories, we only plot the novel class feature distribution. Figure 5.13 shows the plot of the ten unseen categories of the **AwA** dataset. The light blue colored features in the background are the unlabeled test data from the unseen categories. The colored clusters consist of the generated data from all the ten categories, where different colors represent different

categories. These generated features are obtained after the training of our proposed framework without the domain-adaptation step. Evidently, the generated clusters do not overlap with real clusters of the novel categories. However, when we include the domain-adaptation step as shown in Fig. 5.14, the generated and the real clusters overlap and clump together as single clusters for each of the novel categories. As a result, the domain-shift problem between the generated and real data is reduced. Hence, training the final classifier on these adapted generated features increases the recognition performance.

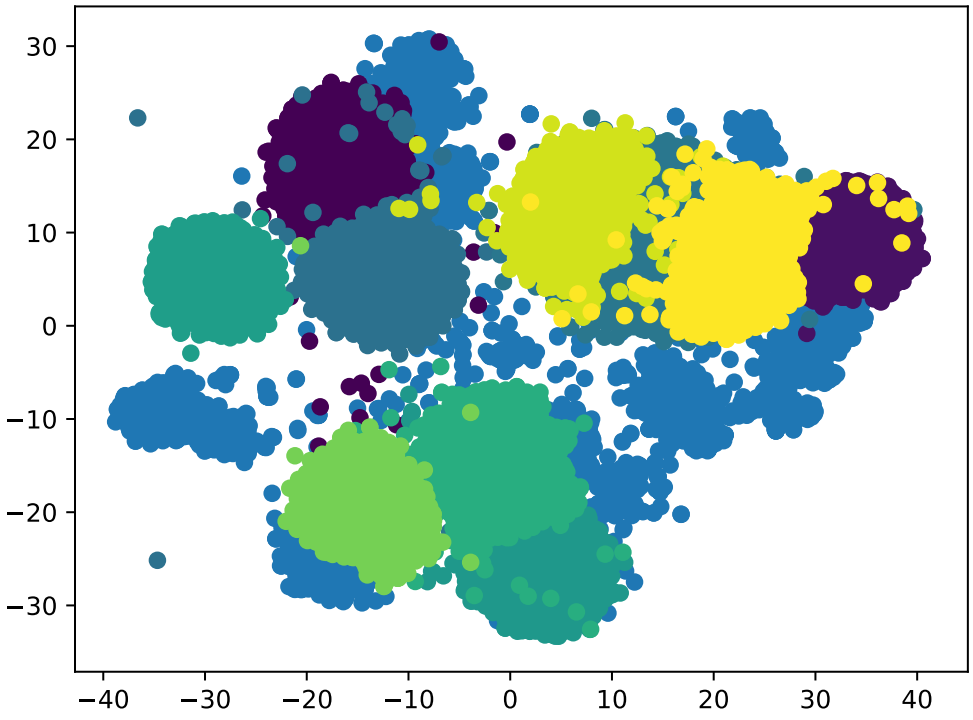


Fig. 5.13. Generated visual features for the unseen classes of the **AwA** dataset when the reconstructor and the discriminator is used along with WGAN. The colored features represent different classes while the light blue features are the unlabeled test data from the unseen classes.

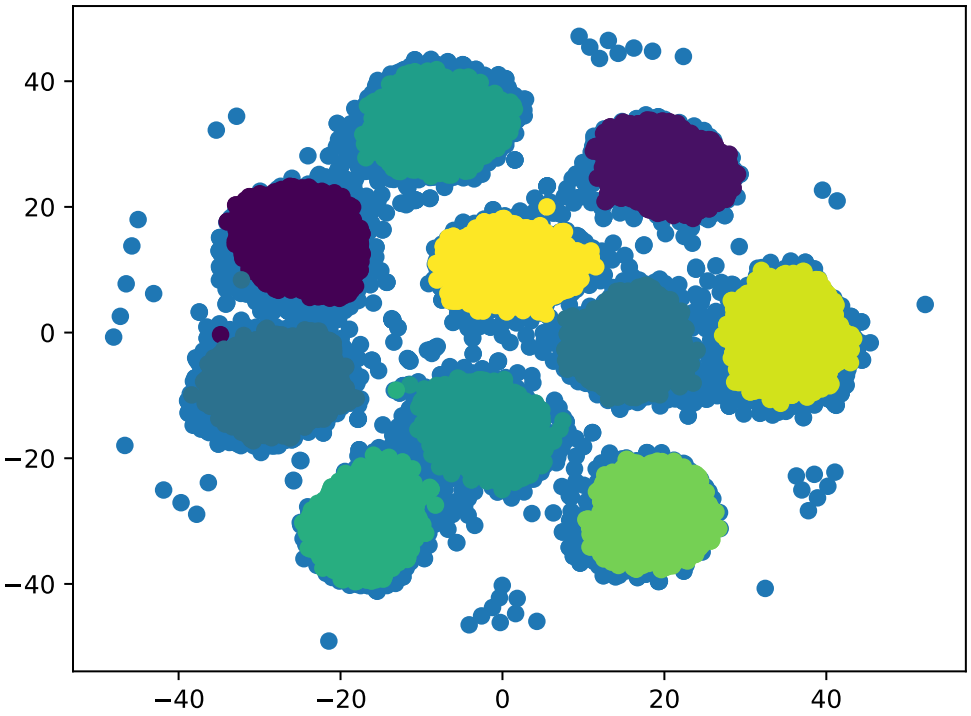


Fig. 5.14. Generated visual features for the unseen classes of the **AwA** dataset when the reconstructor and the discriminator is used along with WGAN and the domain adaptation step is performed.

Convergence Study

We also studied how the test accuracies change with increasing number of epochs. The results on the **AwA** and **CUB** datasets is shown in Figs. 5.15 and 5.16, respectively. From the results, it is seen that the test accuracy of our proposed framework reaches higher values compared to WGAN in almost all the settings. However, the difference between our framework and WGAN is less for the **CUB** dataset because the dataset is fine-grained and it is more difficult to obtain better results without using specialized learning architectures. Again, there is not much difference between our framework and WGAN for the conventional unseen setting because the label

space consists only of unseen categories. As a result, the discrimination and reconstruction network does not have any effect on distinguishing the seen categories from the unseen ones.

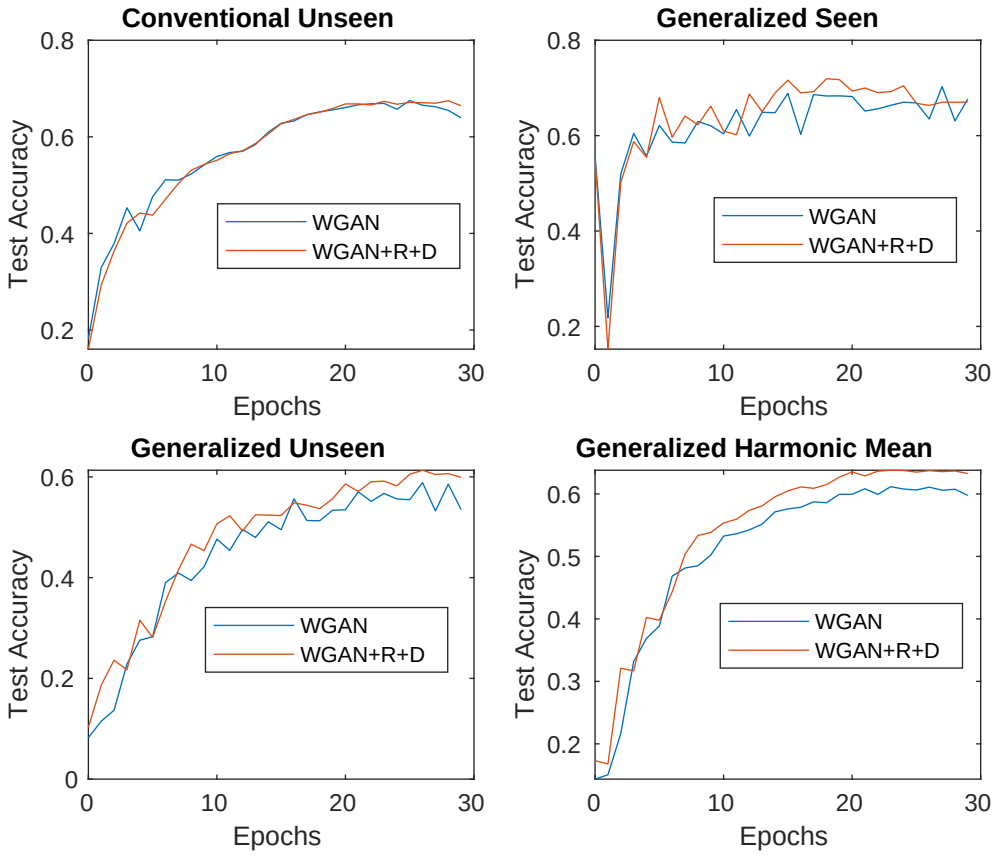


Fig. 5.15. Convergence results of test accuracy with increasing epochs for the **AwA** dataset. R and D stands for the reconstruction and discrimination network respectively.

Effect of Number of Generated Features

We also study the effect of the number of generated features per novel class on test accuracies. The results of the study are carried out on the **FLO** and **CUB** datasets as shown in Figs. 5.17 and 5.18, respectively. From the results, we see that

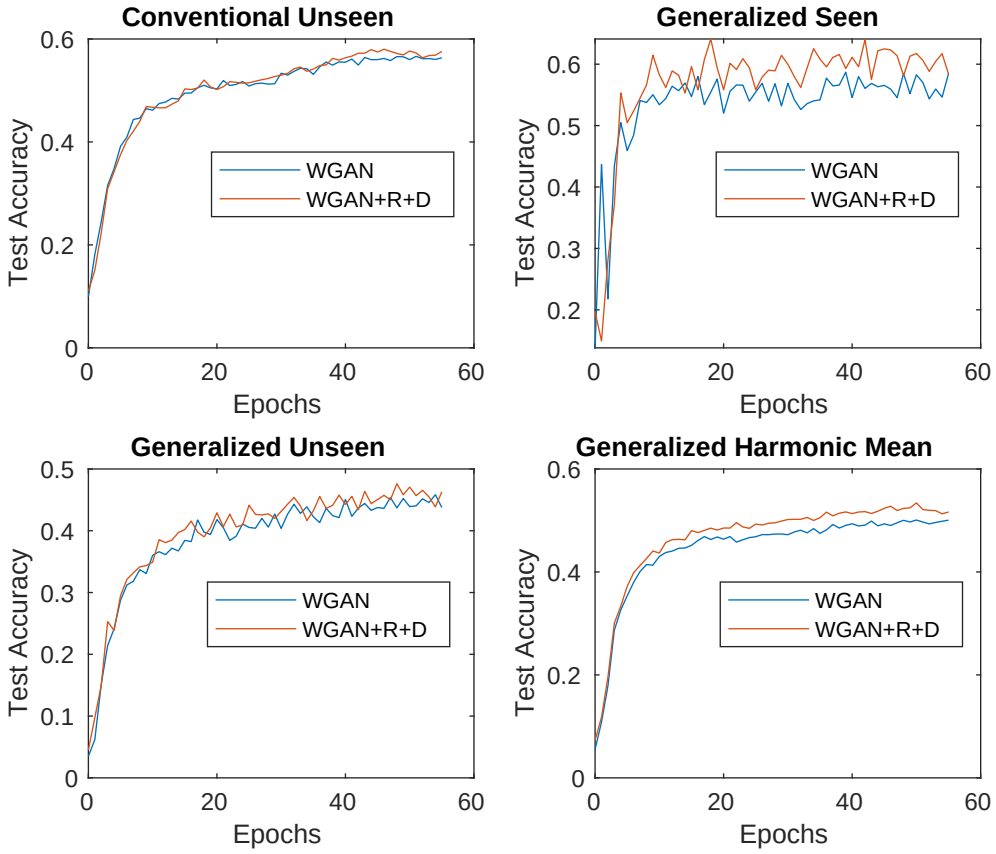


Fig. 5.16. Convergence results of test accuracy with increasing epochs for the **CUB** dataset. R and D stands for the reconstruction and discrimination network respectively.

the recognition performance saturated beyond having a certain number of generated features. This is because the data diversity of a class does not improve beyond having a certain number of samples per class. For both the datasets, the improvement of WGAN+R+D is incremental over WGAN because both the datasets are fine-grained. Therefore, it is difficult to improve the performance by a large margin without fine-grained learning architectures. However, with the inclusion of the domain-adaptation step (A) in WGAN+R+D+A, the recognition performance improved by a larger amount especially for the unseen class accuracy. Also, the generalized seen class accuracy (**s**) dipped when the number of generated features per novel class was in-

creased. This is because when the number of features of the novel classes was less and the data distribution was imbalanced, most of the test samples were classified as seen categories. However, as the number of novel category samples increased, the imbalance was reduced, resulting in mis-classification for the seen categories.

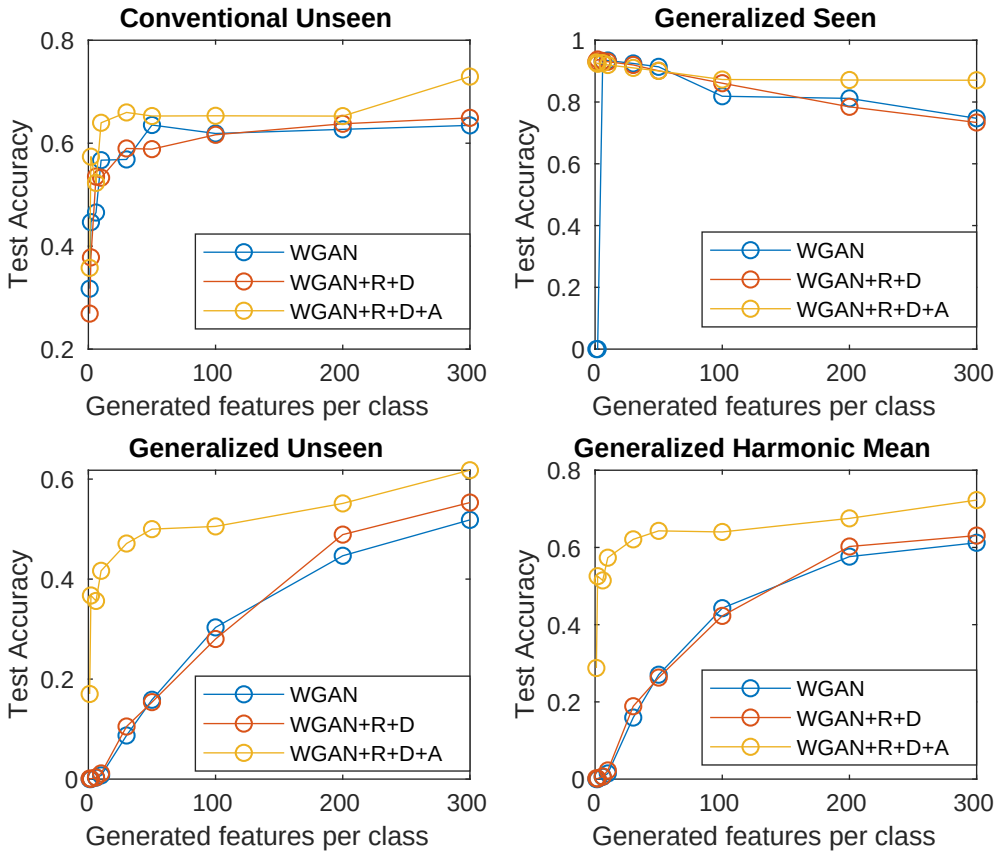


Fig. 5.17. Test accuracy results as the number of generated features per unseen class is increased for the **FLO** dataset. R, D and A stands for the reconstruction network, discrimination network and domain adaptation step respectively.

Parameter Sensitivity Studies

We further study the effect of hyper-parameters λ_r, λ_e and t_h on the recognition performance. The effectiveness of λ_l, λ_{gl} and λ_s has already been studied in [199],

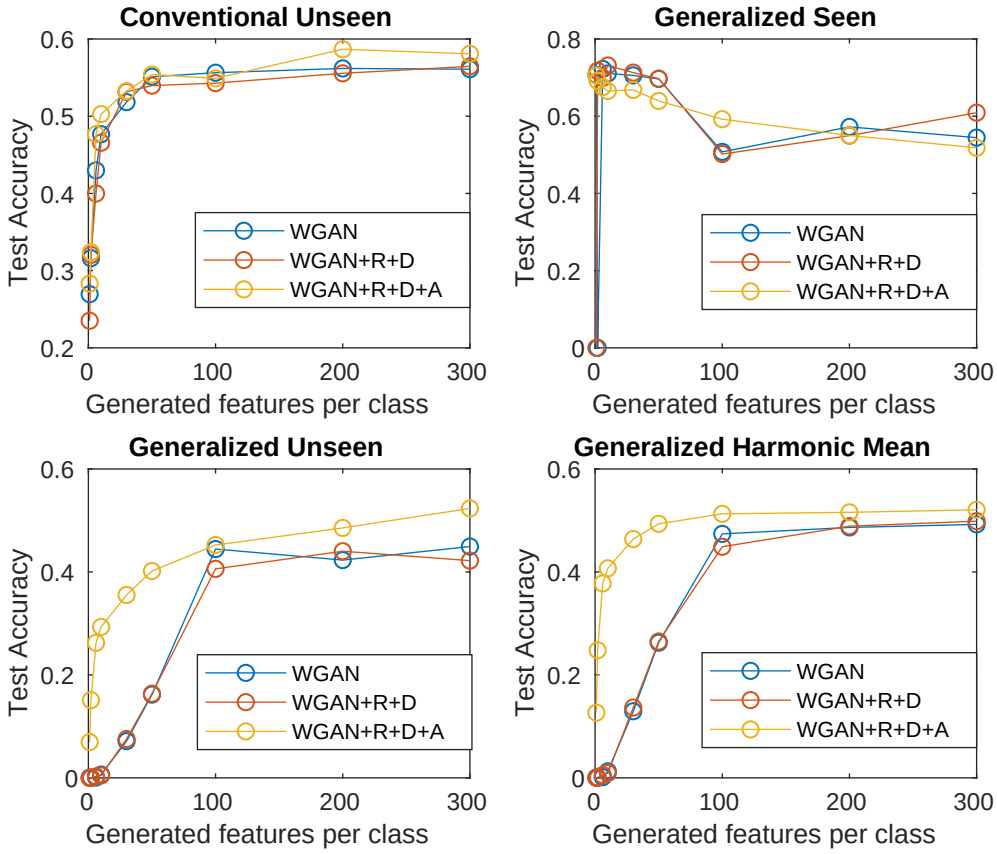


Fig. 5.18. Test accuracy results as the number of generated features per unseen class is increased for the **CUB** dataset.

[149] and [155], respectively. Figure 5.19 shows the results of the generalized test accuracy as the threshold t_h is varied for the **AwA** and **FLO** datasets. The dotted lines represents the oracle situation where we have access to the unseen class test data and we do not need any thresholding mechanism to adapt the generated data to the test data. So, the oracle level is the highest possible performance possible using our domain-adaptation mechanism. On the **AwA** dataset, our method reached very close to the oracle level at $t_h = 0.2$ suggesting the effectiveness of our thresholding step. For the **FLO** dataset, the class-level granularity is finer as a result of which the gap between the oracle and the test accuracy is more. With increasing threshold t_h, only a few highly confident samples were selected from the unlabeled test data. These

confident samples were situated further from the classification decision boundary as a result of which they did not capture the large variation of the unseen class dataset. This caused poor domain-adaptation between the generated data and the selected data, eventually resulting in worse recognition performance.

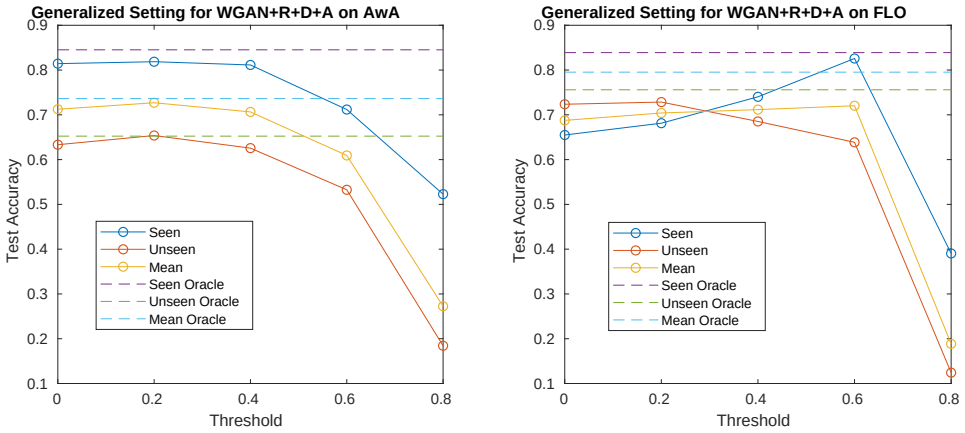


Fig. 5.19. Test accuracy results as the threshold t_h is varied for the **AwA** and the **FLO** dataset . Here, oracle setting considers the case when the unseen test data is known and domain adaptation is carried out without threshold selection mechanism.

Figure 5.20 shows the parameter sensitivity of the test accuracies with respect to λ_e on the **AwA** and **FLO** datasets. The plot shows a monotonically decreasing response, where with increasing λ_e, the test accuracy dipped. This suggests that small values of λ_e are recommended, preferably of the order 10^{-4}. Figure 5.21 shows the parameter sensitivity of the test accuracy with respect to λ_r on the **AwA** and **FLO** datasets. The response is similar to that of λ_e; that is, the test accuracy is monotonically decreasing with increasing λ_r. However, the response due to change in λ_r was a lot more noisy. This is probably because the reconstruction module enforced many-to-one mapping from features to semantic-descriptors. This constraint might not be always satisfied, resulting in sudden changes in the response. Also, the test accuracy response on the **FLO** dataset was relatively more stable. This is because the classes in this dataset were comparatively closer to one another and the diversity

of the features was less. Therefore, the many-to-one mapping constraint using the reconstruction module was easier to be satisfied.

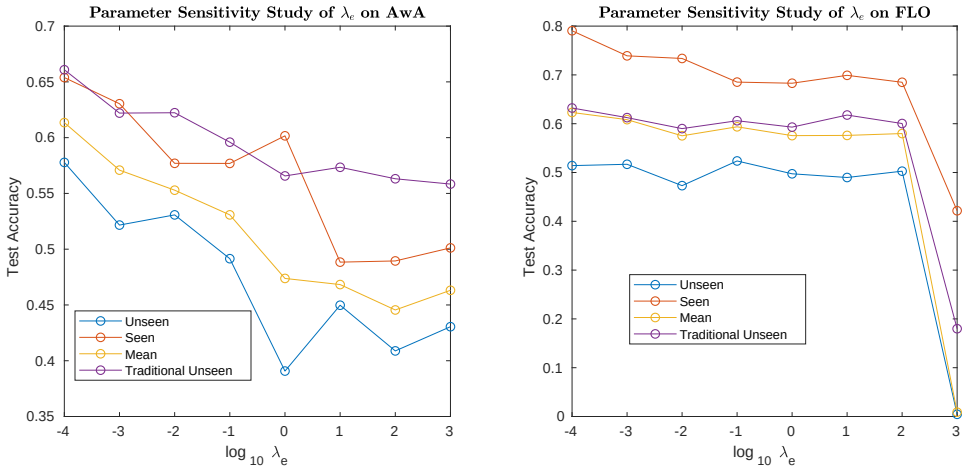


Fig. 5.20. Test accuracy results as λ_e is varied for the **AwA** and the **FLO** dataset.

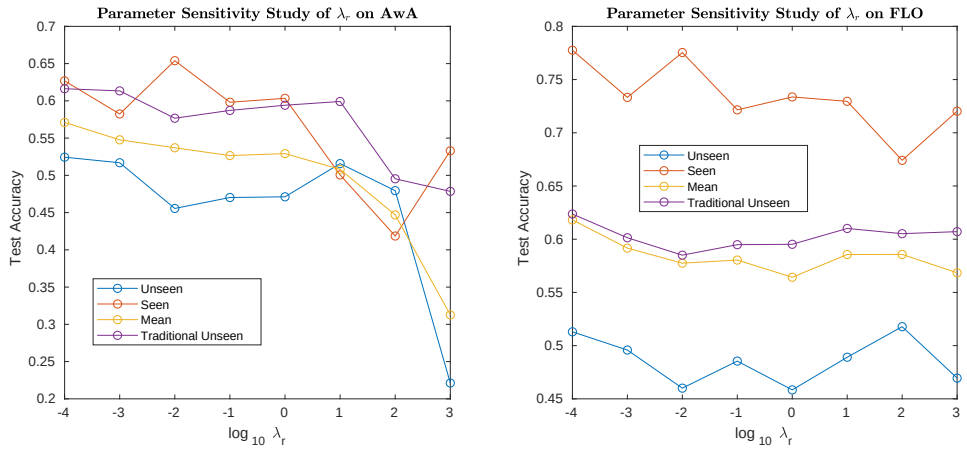


Fig. 5.21. Test accuracy results as λ_r is varied for the **AwA** and the **FLO** dataset.

5.4 Conclusion

In this Chapter, we proposed two approaches to tackle the zero-shot learning and the generalized zero-shot learning problem. Firstly, we proposed an embedding

approach to relate the feature space and the semantic space, where the structural matching loss term reduces the hubness problem and hence improves recognition performance. Also, the calibration of seen class recognition scores prevents the seen class biasness problem. As for the second approach, we proposed a GAN-based constrained framework to generate data for the novel classes. The constraints are in the form of a discriminator and a reconstruction network, both of which improve performance over a WGAN baseline. Both of these approaches contain a domain-adaptation post-processing step that is found to be the most effective in improving the performance. Also, both the embedding and generative approaches produce competitive recognition performance when compared with previous work. However, the generative framework produces better performance compared to the embedding one for most of the datasets.

6. CONCLUSIONS AND FUTURE RESEARCH

6.1 Summary and Conclusions

In this thesis, we have proposed different frameworks for solving different sub-problems in transfer learning. All of the proposed frameworks learn a structural prior from the data-abundant source domain and apply that on the data-starved target domain. The structural priors that we learned are graphs, manifolds and neural networks. The structural prior then encodes relationships among data entities like samples, class-prototypes and semantics. The choice of the structural prior depended on the type of transfer learning sub-problem especially the information availability from the two domains. Accordingly, we proposed frameworks for four different transfer learning problems as described below.

The first subproblem that we considered was unsupervised domain adaptation where we have labeled source domain data and unlabeled target domain data and both domains have distribution discrepancy but consists of the same set of categories. To minimize the distribution discrepancy, our goal was to transform the source domain close to the target domain by exploiting structural similarity between the two domains. Accordingly, we proposed three competitive methods to carry out the transformation between the domains. The first method used graph matching between the source and target domain samples to minimize domain discrepancy. So, we used graphs as the structural prior to encode sample-to-sample relationship. The second method used hyper-graph matching to minimize domain discrepancy. In addition, we also added a preprocessing step where we used only a set of exemplars to construct the hyper-graph, thus increasing computational efficiency. For the third method, we used graph matching for learning a domain-invariant representation. In addition, we also used the pseudo-labels of the target domain to further refine our

model. When these methods were tested on standard domain adaptation datasets for object recognition, they produced much better results than previous methods especially global methods. Also, using higher order information in the form of hypergraphs improved the performance over just using graph matching. Among the three methods, the third method produced the best recognition performance. The training and inference time of the third method was the slowest, the fastest being the first method. This is because the third method learned better representations for domain adaptation. However, it involved learning a large number of parameters resulting in computational inefficiency.

The second sub-problem we considered was few-shot learning. In this case, we had abundant labeled data from the source domain but only sparsely labeled data from the target domain. However, the categories in the source and target domain were different. We identified two problems that caused few-shot learning to perform poorly compared to traditional supervised learning. Firstly, the number of samples in the novel classes were very less compared to the dimensionality of the feature space. This caused the feature space to be sparse resulting in severe overfitting. To address this problem, we proposed a low-dimensional relative representation. This representation used pairwise distances between the few-shot samples thus exploiting the structural information from the data. The second problem was that we could not estimate the mean and variance of the novel classes because of only few-samples from these novel classes. To address this problem, we learned a structural prior in the form of a neural network that would predict the mean and variance. This neural-network based structural prior modeled relationship between samples and class prototypes and helped in improving recognition performance. Our framework was tested on standard few-shot image recognition datasets where it performed considerably better than previous methods. Upon doing ablation analysis, we found the relative feature representation to be most effective in improving the recognition performance. Also, the learned representations were found to be visually more discriminative as compared to previous methods.

We also addressed the problem of few-shot learning in a limited information setting where we only had access to source class prototypes instead of the abundant labeled data. This setting was previously not described in the few-shot learning literature. To address the problem, we proposed a non-parametric approach where the class prototypes were assumed to lie close to a non-linear manifold. Thus, the manifold was used as a structural prior to relate different class prototypes. Accordingly, the novel class prototypes were estimated by projecting the few-shot samples onto the non-linear manifold. Classification was also carried out using a Markov-chain-based manifold distance instead of the traditional nearest-neighbor classification. We also compared our framework with a Bayesian baseline. Results on two image recognition datasets suggested that the manifold-based approach performed better than the Bayesian approach in most of the cases. However, both of them were much better than the no-adaptation baseline in the few-shot regime. The manifold-based distance performed only incrementally better than the nearest-neighbor approach. Among the Bayesian approaches, the simple priors with closed-form posteriors performed better than variational Bayes based approximation methods.

Finally, we addressed the zero-shot learning problem. In this problem, we do not have any labeled data in the target domain. Rather semantic information for all the classes are available. Our goal was to relate the feature space and semantic space using a generative and non-generative approach. For the generative approach, we used a constrained generative adversarial network. The constraints were used to discriminate between seen-unseen classes and to reconstruct semantics from the features. On the other hand, for the non-generative approach, we used structural matching for relating the semantic space and the feature space. Therefore, we used a neural network based structural prior to related the samples and the semantics. In addition, we used domain adaptation as a post-processing step to adapt to the unseen unlabeled data. From the experiments on five standard datasets on zero shot image recognition, our methods performed much better than previous approaches, mainly

due to the domain adaptation step. Also, the generative method performed better than the non-generative approach.

Overall, the contribution of this thesis was the exploration of different mathematical structures and using them as priors for improving performance on different transfer learning problems. The reason why the structural prior based approaches out-performed previous methods is because these structures encode relational knowledge between different entities thus transferring more useful from the source domain to the target domain. The summary of our conributions is shown in Table 6.1.

Table 6.1. Brief summary of the contributions to various transfer learning problems. The second column describes the contributions and their type.

Problem	**Major Contribution**
UDA [149, 150, 152]	Discrepancy: Graph/Hyper-graph Matching Optimization: Conditional Gradient+Network Simplex/ADMM
FSL [153]	Representation: Relative Features Classification: Predictive Statistics
HTL	Estimation: Manifold Classification: Absorbing Markov Chain
ZSL [155]	Constraints: Structural Matching, Discrimination and Reconstruction Post-Processing: Scaled Calibration and Domain Adaptation

6.2 Suggestions for future research

In this thesis, we have addressed different sub-problems of transfer learning in settings that might still be unrealistic. We assume that we have access to labeled data in the source domain which might not always be available. Also, the target domain data becomes available in batch form rather than in an online incremental manner. Similarly, it is assumed that the source domain and target domain modalities are the same. These limitations and assumptions prevent transfer learning from being

deployed to real-world and real-time situations. Therefore, it is worthwhile to relax these assumptions and extend our methods as part of future work. Accordingly, we suggest the following research directions.

In all of the transfer learning sub-problems, we assume that the source domain contains abundant labeled data. However, it might be difficult to have all the source classes labeled. In fact, the labeling might be imbalanced where some classes might have lots of labels and some might not be labeled at all. Since it is difficult to predict ahead of time which of the classes might be labeled, it is appropriate to assume that all the source domain classes are unlabeled. This calls for an unsupervised transfer learning setting. To solve this transfer learning setting, we could do preprocessing where the unlabeled source domain data are clustered to obtain the different class labels. After that, we could proceed with the traditional transfer learning steps. However, the clustering mechanism should be robust enough to handle mis-labels, noise, outliers etc. Alternatively, we could extract transferable knowledge in an unsupervised fashion such that the knowledge can be extracted using distance metrics or some other surrogate tasks used in self-supervised learning.

At the same time, we assume that the target domain data arrives in a batch. On the contrary, in the real world, target domain data arrives sequentially in a streaming manner. Therefore, our proposed approaches cannot be directly applied to this online transfer learning case. So, we need to adapt our models so that they can capture changes in distribution sequentially. We need to develop online versions of our model so that we do not have to train a model from scratch whenever a new target domain data becomes available. As an example, we cannot directly use the correspondence variable for unsupervised domain adaptation. This is because the target domain samples are few at a time and we have to find a correspondence variable each time new samples appear. Instead, it would be better if we find a dynamic relation connecting the correspondence variables at each step. Similarly, few-shot learning and zero-shot learning models should incrementally update the model parameters as

new classes appear. Finally, there is a need to develop more time-efficient versions of our algorithms so they can be used as real-time solutions.

One of the assumptions in our transfer learning models is that the feature space is homogeneous, that is, the source and target domain data lie in the same feature space. On the other hand, we can relax the assumption and account for heterogeneous feature spaces. For example, the source domain can be a 2D image and the target domain can be its 3D point cloud representation. In such situations, our methods cannot work directly since the representation space and dimensionality would be different for the different domains. Therefore, we need to find a new subspace where the source and the target domain data can be mapped. In this new representation space, we could use our traditional homogeneous transfer learning algorithms. However, there is a need to investigate different possibilities about the type and property of the subspace and what metric minimization is required to obtain the source and target domain mapping.

REFERENCES

REFERENCES

[1] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[2] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[3] M. Minsky and S. A. Papert, *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[4] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[8] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *Intern. J. of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advan. Neu. Inf. Proc. Syst.*, 2012, pp. 1097–1105.

[13] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8595–8598.

[14] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[15] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[16] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature medicine*, vol. 25, no. 1, p. 65, 2019.

[17] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[19] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, p. 115, 2017.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[21] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2009, pp. 248–255.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016, pp. 770–778.

[24] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowledge Data Engg.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[25] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, pp. 1–40, 2016.

[26] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017.

[27] J. Shu, Z. Xu, and D. Meng, "Small sample learning in big data era," *arXiv preprint arXiv:1808.04572*, 2018.

[28] W. Jiang, E. Zavesky, S.-F. Chang, and A. Loui, "Cross-domain learning methods for high-level visual concept classification," in *Proc. IEEE Int. Conf. Image Processing*, 2008, pp. 161–164.

[29] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A dasvm classification technique and a circular validation strategy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 770–787, 2010.

[30] L. Duan, I. W. Tsang, D. Xu, and S. J. Maybank, "Domain transfer svm for video concept detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2009, pp. 1375–1381.

[31] J. Yang, R. Yan, and A. G. Hauptmann, "Cross-domain video concept detection using adaptive svms," in *Proc. ACM Int. Conf. on Multimedia*, 2007, pp. 188–197.

[32] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *Intern. Conf. Mach. Learn.*, 2004, p. 114.

[33] T. Kanamori, S. Hido, and M. Sugiyama, "Efficient direct density ratio estimation for non-stationarity adaptation and outlier detection," in *Advan. Neu. Inf. Proc. Syst.*, 2009, pp. 809–816.

[34] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *Advan. Neu. Inf. Proc. Syst.*, 2008, pp. 1433–1440.

[35] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," *Dataset Shift in Machine Learning*, vol. 3, no. 4, p. 5, 2009.

[36] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf, "Correcting sample selection bias by unlabeled data," in *Advan. Neu. Inf. Proc. Syst.*, 2007, p. 601.

[37] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[38] H. Daumé III, "Frustratingly easy domain adaptation," *arXiv preprint arXiv:0907.1815*, 2009.

[39] R. Gopalan, R. Li, and R. Chellappa, "Unsupervised adaptation across domain shifts by generating intermediate data representations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2288–2302, 2014.

[40] ——, "Domain adaptation for object recognition: An unsupervised approach," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 999–1006.

[41] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2012, pp. 2066–2073.

[42] B. Gong, K. Grauman, and F. Sha, "Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation." in *Intern. Conf. Mach. Learn.*, 2013, pp. 222–230.

[43] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2960–2967.

[44] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[45] J. Zhang, W. Li, and P. Ogunbona, "Joint geometrical and statistical alignment for visual domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1859–1867.

[46] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[47] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," *arXiv preprint arXiv:1206.4683*, 2012.

[48] N. Farajidavar, T. E. de Campos, and J. Kittler, "Adaptive transductive transfer machine." in *British Machine Vision Conference*, 2014.

[49] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1853–1865, 2017.

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

[51] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks." in *Intern. Conf. Mach. Learn.*, 2015, pp. 97–105.

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

[53] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi, "Domain generalization for object recognition with multi-task autoencoders," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2551–2559.

[54] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *European Conf. Computer Vision Workshops*, 2016, pp. 443–450.

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

[56] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.

[57] J. Shen, Y. Qu, W. Zhang, and Y. Yong, "Wasserstein distance guided representation learning for domain adaptation," in *AAAI*, 2018, pp. 3–9.

[58] K. Yan, L. Kou, and D. Zhang, "Learning domain-invariant subspace using domain features and independence maximization," *IEEE transactions on cybernetics*, vol. 48, no. 1, pp. 288–299, 2017.

[59] C. Chen, Z. Chen, B. Jiang, and X. Jin, "Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3296–3303.

[60] Y. Pan, T. Yao, Y. Li, Y. Wang, C.-W. Ngo, and T. Mei, "Transferrable prototypical networks for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2239–2247.

[61] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4893–4902.

[62] C.-Y. Lee, T. Batra, M. H. Baig, and D. Ulbricht, "Sliced wasserstein discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 285–10 295.

[63] X. Peng, Z. Huang, X. Sun, and K. Saenko, "Domain agnostic learning with disentangled representations," in *International Conference on Machine Learning*, 2019, pp. 5102–5112.

[64] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *International Conference on Machine Learning*, 2018, pp. 1994–2003.

[65] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Advances in Neural Information Processing Systems*, 2018, pp. 1640–1650.

[66] X. Chen, S. Wang, M. Long, and J. Wang, "Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation," in *International Conference on Machine Learning*, 2019, pp. 1081–1090.

[67] Y. Zhang, H. Tang, K. Jia, and M. Tan, "Domain-symmetric networks for adversarial domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5031–5040.

[68] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, 2006.

[69] R. Salakhutdinov, J. B. Tenenbaum, and A. Torralba, "One-shot learning with a hierarchical nonparametric bayesian model," in *Proc. Intern. Conf. Machine Learning*, 2012, pp. 195–206.

[70] A. Wong and A. L. Yuille, "One shot learning via compositions of meaningful patches," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1197–1205.

[71] B. M. Lake, R. R. Salakhutdinov, and J. Tenenbaum, "One-shot learning by inverting a compositional causal process," in *Advan. Neu. Inf. Proc. Syst.*, 2013, pp. 2526–2534.

[72] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[73] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum, "One shot learning of simple visual concepts," in *Proc. Annual Conf. of the Cognitive Science Society*, 2011, p. 2.

[74] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (O. Chapelle, B. Scholkopf, and A. Zien, eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[75] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[76] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *Advan. Neu. Inf. Proc. Syst.*, 2014, pp. 766–774.

[77] B. Hariharan and R. Girshick, "Low-shot visual recognition by shrinking and hallucinating features," in *Proc. of IEEE Int. Conf. on Computer Vision (ICCV), Venice, Italy*, 2017.

[78] Y.-X. Wang, R. Girshick, M. Herbert, and B. Hariharan, "Low-shot learning from imaginary data," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[79] A. Mehrotra and A. Dukkipati, "Generative adversarial residual pairwise networks for one shot learning," *arXiv preprint arXiv:1703.08033*, 2017.

[80] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan, "Do we need more training data?" *Intern. J. of Computer Vision*, pp. 1–17, 2015.

[81] E. L. Denton, S. Chintala, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1486–1494.

[82] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, A. Kumar, R. Feris, R. Giryes, and A. Bronstein, "Delta-encoder: an effective sample synthesis method for few-shot object recognition," in *Advan. Neu. Inf. Proc. Syst.*, 2018, pp. 2845–2855.

[83] R. Zhang, T. Che, Z. Ghahramani, Y. Bengio, and Y. Song, "Metagan: An adversarial approach to few-shot learning," in *Advan. Neu. Inf. Proc. Syst.*, 2018, pp. 2365–2374.

[84] H. Gao, Z. Shou, A. Zareian, H. Zhang, and S.-F. Chang, "Low-shot learning via covariance-preserving adversarial augmentation networks," in *Advan. Neu. Inf. Proc. Syst.*, 2018, pp. 975–985.

[85] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, 2015.

[86] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advan. Neu. Inf. Proc. Syst.*, 2016, pp. 3630–3638.

[87] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4080–4090.

[88] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2624–2637, 2013.

[89] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[90] B. Oreshkin, P. R. López, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," in *Advan. Neu. Inf. Proc. Syst.*, 2018, pp. 721–731.

[91] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International Conference on Learning Representations*, 2017.

[92] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, 2016, pp. 3981–3989.

[93] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[94] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *NIPS 2017 Workshop on Meta-Learning*, 2017.

[95] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. J. Hwang, and Y. Yang, "Learning to propagate labels: Transductive propagation network for few-shot learning," in *Intern. Conf. Learn. Repr.*, 2019.

[96] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, June 2019.

[97] T. Munkhdalai, X. Yuan, S. Mehri, and A. Trischler, "Rapid adaptation with conditionally shifted neurons," in *Intern. Conf. Mach. Learn.*, 2018.

[98] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International Conference on Machine Learning*, 2016, pp. 1842–1850.

[99] T. Munkhdalai and H. Yu, "Meta networks," in *International Conference on Machine Learning*, 2017, pp. 2554–2563.

[100] P. Shyam, S. Gupta, and A. Dukkipati, "Attentive recurrent comparators," in *International Conference on Machine Learning*, 2017, pp. 3173–3181.

[101] Y.-X. Wang and M. Hebert, "Learning to learn: Model regression networks for easy small sample learning," in *European Conference on Computer Vision*, 2016, pp. 616–634.

[102] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[103] S. Qiao, C. Liu, W. Shen, and A. L. Yuille, "Few-shot image recognition by predicting parameters from activations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018, pp. 7229–7238.

[104] H. Qi, M. Brown, and D. G. Lowe, "Low-shot learning with imprinted weights," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018, pp. 5822–5830.

[105] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," in *Intern. Conf. Learn. Repr.*, 2019.

[106] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," in *Intern. Conf. Learn. Repr.*, 2018.

[107] E. Triantafillou, R. Zemel, and R. Urtasun, "Few-shot learning through an information retrieval lens," in *Advan. Neu. Inf. Proc. Syst.*, 2017, pp. 2255–2265.

[108] Y. Wang and Q. Yao, "Few-shot learning: A survey," *arXiv preprint arXiv:1904.05046*, 2019.

[109] F. Orabona, C. Castellini, B. Caputo, A. E. Fiorilla, and G. Sandini, "Model adaptation with least-squares svm for adaptive hand prosthetics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2009, pp. 2897–2903.

[110] T. Tommasi, F. Orabona, and B. Caputo, "Learning categories from few examples with multi model knowledge transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 928–941, 2014.

[111] I. Kuzborskij, F. Orabona, and B. Caputo, "From n to n+ 1: Multiclass transfer incremental learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2013, pp. 3358–3365.

[112] L. Jie, T. Tommasi, and B. Caputo, "Multiclass transfer learning from unconstrained priors," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 1863–1870.

[113] I. Kuzborskij and F. Orabona, "Stability and hypothesis transfer learning," in *Intern. Conf. Mach. Learn.*, 2013, pp. 942–950.

[114] ——, "Fast rates by transferring from auxiliary hypotheses," *Machine Learning*, vol. 106, no. 2, pp. 171–195, 2017.

[115] I. Kuzborskij, F. Orabona, and B. Caputo, "Transfer learning through greedy subset selection," in *Proc. Intern. Conf. Image Analysis and Processing*, 2015, pp. 3–14.

[116] P. R. Diana Benavides-Prado, Yun Sing Koh, "Accgensvm: Selectively transferring from previous hypotheses," in *Proc. Intern. Joint Conf. Artificial Intel.*, 2017, pp. 1440–1446.

[117] Y.-X. Wang and M. Hebert, "Learning by transferring from unsupervised universal sources." in *Proc. Assoc. Advance. Artificial Intel.*, 2016, pp. 2187–2193.

[118] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 453–465, 2014.

[119] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advan. Neu. Inf. Proc. Syst.*, 2013, pp. 3111–3119.

[120] L. Zhang, T. Xiang, and S. Gong, "Learning a deep embedding model for zero-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2017.

[121] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1425–1438, 2016.

[122] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov *et al.*, "Devise: A deep visual-semantic embedding model," in *Advan. Neu. Inf. Proc. Syst.*, 2013, pp. 2121–2129.

[123] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, "Zero-shot learning through cross-modal transfer," in *Advan. Neu. Inf. Proc. Syst.*, 2013, pp. 935–943.

[124] Z. Zhang and V. Saligrama, "Zero-shot learning via joint latent similarity embedding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016, pp. 6034–6042.

[125] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha, "Synthesized classifiers for zero-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016, pp. 5327–5336.

[126] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *Intern. Conf. Mach. Learn.*, 2015, pp. 2152–2161.

[127] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele, "Evaluation of output embeddings for fine-grained image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2015, pp. 2927–2936.

[128] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele, "Latent embeddings for zero-shot classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016, pp. 69–77.

[129] E. Kodirov, T. Xiang, and S. Gong, "Semantic autoencoder for zero-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2017, pp. 4447–4456.

[130] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha, "An empirical study and analysis of generalized zero-shot learning for object recognition in the wild," in *European Conference on Computer Vision*, 2016, pp. 52–68.

[131] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong, "Transductive multi-view zero-shot learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2332–2345, 2015.

[132] E. Kodirov, T. Xiang, Z. Fu, and S. Gong, "Unsupervised domain adaptation for zero-shot learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2452–2460.

[133] M. Rostami, S. Kolouri, Z. Murez, Y. Owekcho, E. Eaton, and K. Kim, "Zero-shot image classification using coupled dictionary embedding," *arXiv preprint arXiv:1906.10509*, 2019.

[134] S. M. Shojaee and M. S. Baghshah, "Semi-supervised zero-shot learning by a clustering-based approach," *arXiv preprint arXiv:1605.09016*, 2016.

[135] Y. Li, D. Wang, H. Hu, Y. Lin, and Y. Zhuang, "Zero-shot recognition using dual visual-semantic mapping paths," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2017, pp. 5207–5215.

[136] Z. Zhang and V. Saligrama, "Zero-shot learning via semantic similarity embedding," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4166–4174.

[137] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean, "Zero-shot learning by convex combination of semantic embeddings," in *Intern. Conf. Learning Representations*, 2013.

[138] M. Bucher, S. Herbin, and F. Jurie, "Generating visual representations for zero-shot classification," in *International Conference on Computer Vision (ICCV) Workshops: TASK-CV: Transferring and Adapting Source Knowledge in Computer Vision*, 2017.

[139] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata, "Feature generating networks for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5542–5551.

[140] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[141] Y. Zhu, M. Elhoseiny, B. Liu, X. Peng, and A. Elgammal, "A generative adversarial approach for zero-shot learning from noisy texts," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1004–1013.

[142] J. Li, M. Jing, K. Lu, Z. Ding, L. Zhu, and Z. Huang, "Leveraging the invariant side of generative zero-shot learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[143] A. Mishra, S. Krishna Reddy, A. Mittal, and H. A. Murthy, "A generative model for zero shot learning using conditional variational autoencoders," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2188–2196.

[144] V. K. Verma, G. Arora, A. Mishra, and P. Rai, "Generalized zero-shot learning via synthesized examples," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018.

[145] H. Huang, C. Wang, P. S. Yu, and C.-D. Wang, "Generative dual adversarial network for generalized zero-shot learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[146] E. Schonfeld, S. Ebrahimi, S. Sinha, T. Darrell, and Z. Akata, "Generalized zero- and few-shot learning via aligned variational autoencoders," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[147] R. Felix, V. B. Kumar, I. Reid, and G. Carneiro, "Multi-modal cycle-consistent generalized zero-shot learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 21–37.

[148] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

[149] D. Das and C. S. G. Lee, "Sample-to-sample correspondence for unsupervised domain adaptation," *Engineering Applications of Artificial Intelligence*, vol. 73, pp. 80–91, 2018.

[150] ——, "Unsupervised domain adaptation using regularized hyper-graph matching," in *IEEE Intern. Conf. Image Processing*, 2018, pp. 3758–3762.

[151] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[152] D. Das and C. S. G. Lee, "Graph matching and pseudo-label guided deep unsupervised domain adaptation," in *Proceedings of the International Conference on Artificial Neural Networks*, 2018.

[153] D. Das and C. S. G. Lee, "A two-stage approach to few-shot learning for image recognition," *IEEE Transactions on Image Processing*, vol. 29, pp. 3336–3350, 2020.

[154] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 2, pp. 218–233, 2003.

[155] D. Das and C. S. George Lee, "Zero-shot image recognition using relational matching, adaptation and calibration," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[156] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Proc. Intern. Society Optics Photonics*. International Society for Optics and Photonics, 1992, pp. 586–606.

[157] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding*, vol. 89, no. 2, pp. 114–141, 2003.

[158] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce, "A tensor-based algorithm for high-order graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 12, pp. 2383–2395, 2011.

[159] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[160] E. D. Andersen, "Complexity of solving conic quadratic problems," http://erlingdandersen.blogspot.com/2013/11/complexity-of-solving-conic-quadratic.html, 2013, accessed: 2010-09-30.

[161] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics (NRL)*, vol. 3, no. 1-2, pp. 95–110, 1956.

[162] M. Jaggi, "Revisiting frank-wolfe: Projection-free sparse convex optimization." in *Intern. Conf. Mach. Learn.*, 2013, pp. 427–435.

[163] A. Mosek, "The mosek optimization software," *Online at http://www. mosek. com*, vol. 54, pp. 2–1, 2010.

[164] D. J. Kelly, "The minimum cost flow problem and the network simplex solution method," Ph.D. dissertation, 1991.

[165] P. Germain, A. Habrard, F. Laviolette, and E. Morvant, "A pac-bayesian approach for domain adaptation with specialization to linear classifiers," in *Intern. Conf. Mach. Learn.*, 2013, pp. 738–746.

[166] E. Zhong, W. Fan, Q. Yang, O. Verscheure, and J. Ren, "Cross validation framework to choose amongst models and datasets for transfer learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 547–562.

[167] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European Conf. Computer Vision*, 2010, pp. 213–226.

[168] J. Zheng, M.-Y. Liu, R. Chellappa, and P. J. Phillips, "A grassmann manifold-based domain adaptation approach," in *Proc. IEEE Int. Conf. Pattern Recognition*, 2012, pp. 2095–2099.

[169] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007.

[170] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European Conf. Computer Vision*, 2006, pp. 404–417.

[171] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition." in *Intern. Conf. Mach. Learn.*, 2014, pp. 647–655.

[172] S. Si, D. Tao, and B. Geng, "Bregman divergence-based regularization for transfer subspace learning," *IEEE Trans. Knowledge Data Engg.*, vol. 22, no. 7, pp. 929–942, 2010.

[173] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2200–2207.

[174] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[175] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," in *Proc. Annual Meeting Association Computational Linguistics*, 2007, pp. 440–447.

[176] D. Dueck and B. J. Frey, "Non-metric affinity propagation for unsupervised image categorization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2007, pp. 1–8.

[177] J. Friedman, T. Hastie, and R. Tibshirani, "A note on the group lasso and a sparse group lasso," *arXiv preprint arXiv:1001.0736*, 2010.

[178] O. Chapelle, B. Schlkopf, and A. Zien, *Semi-Supervised Learning*, 1st ed. The MIT Press, 2010.

[179] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Intern. Conf. Learning Representations*, 2015.

[180] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2014, pp. 1717–1724.

[181] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowledge Data Engg.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[182] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.

[183] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Univ. of Torornto, Tech. Rep., 2009.

[184] N. Hilliard, L. Phillips, S. Howland, A. Yankov, C. D. Corley, and N. O. Hodas, "Few-shot learning with metric-agnostic conditional embeddings," *arXiv preprint arXiv:1802.04376*, 2018.

[185] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi, "Learning feed-forward one-shot learners," in *Advan. Neu. Inf. Proc. Syst.*, 2016, pp. 523–531.

[186] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Intern. Conf. Mach. Learn.*, 2015, pp. 448–456.

[187] L. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," in *Intern. Conf. Learning Representations*, 2017.

[188] H. Edwards and A. Storkey, "Towards a neural statistician," in *Intern. Conf. Learning Representations*, 2017.

[189] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[190] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[191] I. Kuzborskij, F. Orabona, and B. Caputo, "Scalable greedy algorithms for transfer learning," *Comput. Vis. Image Under.*, vol. 156, pp. 174–185, 2017.

[192] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Hubs in space: Popular nearest neighbors in high-dimensional data," *Journal of Machine Learning Research*, vol. 11, no. Sep, pp. 2487–2531, 2010.

[193] N. Bonneel, M. Van De Panne, S. Paris, and W. Heidrich, "Displacement interpolation using lagrangian mass transport," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6, p. 158, 2011.

[194] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2009, pp. 1778–1785.

[195] G. Patterson and J. Hays, "Sun attribute database: Discovering, annotating, and recognizing scene attributes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2012, pp. 2751–2758.

[196] V. K. Verma and P. Rai, "A simple exponential family framework for zero-shot learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2017, pp. 792–808.

[197] Y. Annadani and S. Biswas, "Preserving semantic relations for zero-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018, pp. 7603–7612.

[198] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.

[199] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, 2017, pp. 5767–5777.

[200] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[201] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE, 2008, pp. 722–729.

[202] E. Kokiopoulou, J. Chen, and Y. Saad, "Trace optimization and eigenproblems in dimension reduction methods," *Numerical Linear Algebra with Applications*, vol. 18, no. 3, pp. 565–602, 2011.

[203] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

APPENDICES

A. ALGORITHM FOR FINDING EXTRINSIC MANIFOLD MEAN

To calculate the extrinsic mean, our goal is to solve for $\mathbf{S}$ from the following optimization problem,

$$\min \sum_{i=1}^{r} \frac{1}{2} ||\mathbf{S}\mathbf{S}^T - \mathbf{S}_i\mathbf{S}_i^T||_{\mathcal{F}}^2 \quad \text{such that} \quad \mathbf{S}^T\mathbf{S} = \mathbf{I} \; .$$

We reformulate the objective function for each part of the sum. Let $\mathbf{Q}_i = \mathbf{S}_i\mathbf{S}_i^T$. Accordingly,

$$\begin{aligned} \frac{1}{2}||\mathbf{S}\mathbf{S}^T - \mathbf{Q}_i||_{\mathcal{F}}^2 &= \frac{1}{2} \operatorname{Tr}[(\mathbf{S}\mathbf{S}^T - \mathbf{Q}_i)^T(\mathbf{S}\mathbf{S}^T - \mathbf{Q}_i)] \\ &= \frac{1}{2} \operatorname{Tr}(\mathbf{S}\mathbf{S}^T\mathbf{S}\mathbf{S}^T - \mathbf{Q}_i^T\mathbf{S}\mathbf{S}^T - \mathbf{S}\mathbf{S}^T\mathbf{Q}_i + \mathbf{Q}_i^T\mathbf{Q}_i) \end{aligned}$$

where Tr indicates a matrix Trace operation. Using the properties of Trace operation, $\operatorname{Tr}(\mathbf{AB}) = \operatorname{Tr}(\mathbf{BA})$, $\operatorname{Tr}(\mathbf{ABC}) = \operatorname{Tr}(\mathbf{BCA}) = \operatorname{Tr}(\mathbf{CAB})$, and the fact that $\mathbf{Q}_i = \mathbf{Q}_i^T$ as well as the constraint $\mathbf{S}^T\mathbf{S} = \mathbf{I}$, we obtain the expression:

$$\begin{aligned} &\frac{1}{2} \operatorname{Tr}(\mathbf{S}^T\mathbf{S} - 2\mathbf{S}^T\mathbf{Q}_i^T\mathbf{S} + \mathbf{Q}_i^T\mathbf{Q}_i) \\ &= \frac{1}{2} \operatorname{Tr}(\mathbf{S}^T\mathbf{S}) - \operatorname{Tr}(\mathbf{S}^T\mathbf{Q}_i^T\mathbf{S}) + \frac{1}{2} \operatorname{Tr}(\mathbf{Q}_i^T\mathbf{Q}_i) \\ &= \frac{1}{2} \operatorname{Tr}(\mathbf{I}) - \operatorname{Tr}(\mathbf{S}^T\mathbf{Q}_i^T\mathbf{S}) + \frac{1}{2} \operatorname{Tr}(\mathbf{Q}_i^T\mathbf{Q}_i) \\ &= - \operatorname{Tr}(\mathbf{S}^T\mathbf{Q}_i^T\mathbf{S}) + \text{terms not depending on } \mathbf{S}. \end{aligned}$$

Therefore, the objective function becomes to maximize $\sum_{i=1}^{r} \operatorname{Tr}(\mathbf{S}^T\mathbf{Q}_i^T\mathbf{S})$ or to maximize $\operatorname{Tr}(\mathbf{S}^T(\sum_{i=1}^{r} \mathbf{S}_i\mathbf{S}_i^T)\mathbf{S})$ given the constraints $\mathbf{S}^T\mathbf{S} = \mathbf{I}$. The solution to this problem is well known and can be framed as an eigenvalue problem as described in [202]. The algorithm for finding the extrinsic mean as an eigenvalue problem is described in Algorithm 12.

Algorithm 12: Computation of Extrinsic Manifold Mean.

Given: r orthonormal linear subspaces represented as $\mathbf{S}_i \in \mathbb{R}^{d \times (q+1)}$ for $i \in \{1, 2, ..., r\}$.

Goal: Obtain the Extrinsic Manifold Mean $\bar{\mathbf{S}}$ of the subspaces

Sum outer products of the matrices : $\mathbf{A} \leftarrow \sum_{i=1}^{r} \mathbf{S}_i \mathbf{S}_i^T$

Obtain right-eigen value decomposition : $\mathbf{E} \leftarrow \text{eig}(\mathbf{A})$

Keep top $q+1$ eigen-vectors of $\mathbf{E}$ to obtain $\bar{\mathbf{S}}$

Result *Extrinsic Manifold Mean obtained*

B. TIME AND SPACE COMPLEXITY OF THE MANIFOLD APPROACH

In the overall algorithm of the manifold-based approach, the computationally intensive steps are: (a) computation of the extrinsic manifold mean and (b) the absorbing Markov-chain-based inference procedure. Computing the extrinsic mean requires the eigenvalue decomposition of a $d \times d$ matrix that has the time complexity $O(d^3)$, where d is the dimensionality of the feature space.

In the absorbing Markov-chain process, we have the following operation to calculate the steady-state probabilities - $\mathbf{u}_{n_a}^{\infty} = \mathbf{u}_{n_t}^{0}(\mathbf{I}-\mathbf{T})^{-1}\mathbf{A}+\mathbf{u}_{n_a}^{0}$. Here $\mathbf{u}^0 = [\mathbf{u}_{n_t}^0 \quad \mathbf{u}_{n_a}^0]$ and $\mathbf{u}^{\infty} = [\mathbf{u}_{n_t}^{\infty} \quad \mathbf{u}_{n_a}^{\infty}]$. Also $\mathbf{u}_{n_t}^0 \in \mathbb{R}^{1\times n_t}, \mathbf{u}_{n_a}^0 \in \mathbb{R}^{1\times n_a}, \mathbf{T} \in \mathbb{R}^{n_t \times n_t}, \mathbf{A} \in \mathbb{R}^{n_t \times n_a}$. The dominant time-consuming step is the multiplication operation. In the multiplication operation, we have an inverse operation that is $O(n_t^3)$. Then the multiplication $(\mathbf{I}-\mathbf{T})^{-1}\mathbf{A}$ is $O(n_t^2 n_a)$. Then the multiplication $\mathbf{u}^0((\mathbf{I}-\mathbf{T})^{-1}\mathbf{A})$ is $O(n_t n_a)$. Hence, the overall time complexity is $O(n_t^3 + n_t^2 n_a + n_t n_a) = O(n_t^3 + n_t^2 n_a)$. But we have two steps of the absorbing Markov-chain process, where the n_b base and n_{nov} novel classes are interchanged for the roles of transient and absorbing states. Hence, the time complexity becomes $O(n_b^3 + n_b^2 n_{nov}) + O(n_{nov}^3 + n_{nov}^2 n_b) = O((n_b + n_{nov})(n_b^2 + n_{nov}^2))$ after some factorization steps.

In terms of the space complexity for the extrinsic manifold mean calculation, we have to construct and store r quantities of $d\times(q+1)$ dimensional matrices representing the linear subspaces. Therefore, the corresponding space complexity is $O(rd(q+1))$. For the absorbing Markov-chain process, one needs to store the probability transition matrix, which is a square matrix of size $(n_b + n_{nov}) \times (n_b + n_{nov})$, where n_b and n_{nov} are the numbers of base and novel categories, respectively. Hence, the corresponding space complexity is $O((n_b + n_{nov})^2)$.

C. DERIVATION OF CIRCULARLY DEPENDENT EQUATIONS OF THE BAYESIAN APPROACH

Our goal is to derive the circularly dependent equations (i.e., Eqs. (4.10)-(4.12)). These equations were obtained using a variational Bayes mean-field approximation on the posterior when we use a normal prior on the mean $\boldsymbol{\mu}$ and a gamma prior on the precision λ.

If we use a mean-field approximation, we assume that the posterior distribution $q(\cdot)$ can be factorized as

$$q(\boldsymbol{\mu}, \lambda) = q_1(\boldsymbol{\mu}) q_2(\lambda),$$

where $q_1(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_V, \lambda_V)$ and $q_2(\lambda) = Ga(\lambda|\alpha_V, \beta_V)$. If we use an alternating optimization procedure [203], we have

$$\log q_1(\boldsymbol{\mu}) \propto \mathbb{E}_\lambda[\log p_1(\mathbf{X}|\boldsymbol{\mu}, \lambda) + \log p_2(\boldsymbol{\mu}) + \log p_3(\lambda)]$$
$$\log q_2(\lambda) \propto \mathbb{E}_{\boldsymbol{\mu}}[\log p_1(\mathbf{X}|\boldsymbol{\mu}, \lambda) + \log p_2(\boldsymbol{\mu}) + \log p_3(\lambda)],$$

where $p_1(\mathbf{X}|\boldsymbol{\mu}, \lambda) = \prod_{i=1}^{k} \mathcal{N}(\mathbf{x}_{ni}|\boldsymbol{\mu}, \lambda^{-1}), p_2(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_0, \lambda_0^{-1})$ and $p_3(\lambda) = Ga(\lambda|\alpha, \beta)$. So accordingly,

$$\log q_1(\boldsymbol{\mu}) \propto \mathbb{E}_\lambda[\log(\prod_{i=1}^{k} \mathcal{N}(\mathbf{x}_{ni}|\boldsymbol{\mu}, \lambda^{-1})) + \log(\mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_0, \lambda_0^{-1})) + \log(Ga(\lambda|\alpha, \beta))].$$

Upon expanding the normal and gamma distributions and separating out the constant terms, we have

$$\log q_1(\boldsymbol{\mu}) \propto -\frac{1}{2}\mathbb{E}_\lambda\Big[\sum_{i=1}^{k} ||\mathbf{x}_{ni} - \boldsymbol{\mu}||_2^2 \lambda + ||\boldsymbol{\mu} - \boldsymbol{\mu}_0||_2^2 \lambda_0\Big]$$
$$= -\frac{1}{2}[\sum_{i=1}^{k} ||\mathbf{x}_{ni} - \boldsymbol{\mu}||_2^2 \mathbb{E}_\lambda(\lambda) + ||\boldsymbol{\mu} - \boldsymbol{\mu}_0||_2^2 \lambda_0].$$

Since the posterior of λ is from the $Ga(\lambda|\alpha,\beta)$ distribution, $\mathbb{E}_\lambda(\lambda) = \frac{\alpha_V}{\beta_V}$. After plugging it in and separating out the constants with respect to $\boldsymbol{\mu}$, we obtain

$$\log q_1(\boldsymbol{\mu}) \propto -\frac{1}{2}\big[(k\frac{\alpha_V}{\beta_V} + \lambda_0)\boldsymbol{\mu}^T\boldsymbol{\mu} - 2(\sum_{i=1}^{k}\mathbf{x}_{ni}\frac{\alpha_V}{\beta_V} + \boldsymbol{\mu}_0\lambda_0)^T\boldsymbol{\mu}\big],$$

which can then be rearranged as

$$\log q_1(\boldsymbol{\mu}) \propto -\frac{1}{2}(k\frac{\alpha_V}{\beta_V} + \lambda_0)||\boldsymbol{\mu} - \frac{\sum_{i=1}^{k}\mathbf{x}_{ni}\frac{\alpha_V}{\beta_V} + \boldsymbol{\mu}_0\lambda_0}{k\frac{\alpha_V}{\beta_V} + \lambda_0}||_2^2.$$

Rearranging, we obtain

$$q_1(\boldsymbol{\mu}) \propto \exp\big(-\frac{1}{2}(k\frac{\alpha_V}{\beta_V} + \lambda_0)||\boldsymbol{\mu} - \frac{\sum_{i=1}^{k}\mathbf{x}_{ni}\frac{\alpha_V}{\beta_V} + \boldsymbol{\mu}_0\lambda_0}{k\frac{\alpha_V}{\beta_V} + \lambda_0}||_2^2\big),$$

which is of the form $q_1(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_V, \lambda_V)$, where
$\boldsymbol{\mu}_V = \frac{\sum_{i=1}^{k}\mathbf{x}_{ni}\frac{\alpha_V}{\beta_V} + \boldsymbol{\mu}_0\lambda_0}{k\frac{\alpha_V}{\beta_V} + \lambda_0}$ and $\lambda_V = k\frac{\alpha_V}{\beta_V} + \lambda_0$.

To derive $q_2(\lambda)$, we proceed accordingly and have

$$\log q_2(\lambda) \propto \mathbb{E}_{\boldsymbol{\mu}}\Big[\log\big(\prod_{i=1}^{k}\mathcal{N}(\mathbf{x}_{ni}|\boldsymbol{\mu},\lambda^{-1})\big) + \log\big(\mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_0,\lambda_0^{-1})\big) + \log\big(Ga(\lambda|\alpha,\beta)\big)\Big].$$

Upon expanding the normal and gamma distributions and separating out the constant terms, we have

$$\log q_2(\lambda) \propto (\alpha + \frac{dk}{2} - 1)\log\lambda - \lambda\Big(\beta + \frac{1}{2}\mathbb{E}_{\boldsymbol{\mu}}\Big[\sum_{i=1}^{k}||\mathbf{x}_{ni} - \boldsymbol{\mu}||_2^2\Big]\Big).$$

We next expand

$$\mathbb{E}_{\boldsymbol{\mu}}\Big[\sum_{i=1}^{k}||\mathbf{x}_{ni} - \boldsymbol{\mu}||_2^2\Big] = \sum_{i=1}^{k}\mathbf{x}_{ni}^T\mathbf{x}_{ni} + k\mathbb{E}_{\boldsymbol{\mu}}[\boldsymbol{\mu}^T\boldsymbol{\mu}] - 2\sum_{i=1}^{k}\mathbf{x}_{ni}^T\mathbb{E}_{\boldsymbol{\mu}}[\boldsymbol{\mu}].$$

We can easily show that $\mathbb{E}_{\boldsymbol{\mu}}[\boldsymbol{\mu}] = \boldsymbol{\mu}_V$ and $\mathbb{E}_{\boldsymbol{\mu}}[\boldsymbol{\mu}^T\boldsymbol{\mu}] = \frac{d}{\lambda_V} + \boldsymbol{\mu}_V^T\boldsymbol{\mu}_V$ using the identity $\mathbb{E}_a[a^2] = \mathrm{Var}_a[a] + \mathbb{E}_a[a]^2$, where a is each element of $\boldsymbol{\mu}_V$.

Plugging the expressions back, we obtain

$$\log q_2(\lambda) \propto (\alpha + \frac{dk}{2} - 1)\log\lambda - \lambda\Big(\beta + \frac{1}{2}\big(\sum_{i=1}^{k}\mathbf{x}_{ni}^T\mathbf{x}_{ni} - 2\sum_{i=1}^{k}\mathbf{x}_{ni}^T\boldsymbol{\mu}_V + k(\frac{d}{\lambda_V} + \boldsymbol{\mu}_V^T\boldsymbol{\mu}_V)\big)\Big)$$

or

$$q_2(\lambda) \propto \lambda^{\alpha+\frac{dk}{2}-1} \exp\Big(-\lambda\big(\beta + \frac{1}{2}(\sum_{i=1}^{k} \mathbf{x}_{ni}^T \mathbf{x}_{ni} - 2\sum_{i=1}^{k} \mathbf{x}_{ni}^T \boldsymbol{\mu}_V + k(\frac{d}{\lambda_V} + \boldsymbol{\mu}_V^T \boldsymbol{\mu}_V))\big)\Big),$$

which can be identified as a gamma distribution such that $q_2(\lambda) = Ga(\lambda|\alpha_V, \beta_V)$, where $\alpha_V = \alpha + \frac{dk}{2}$ and

$$\beta_V = \beta + \frac{1}{2}(\sum_{i=1}^{k} \mathbf{x}_{ni}^T \mathbf{x}_{ni} - 2\sum_{i=1}^{k} \mathbf{x}_{ni}^T \boldsymbol{\mu}_V + k(\frac{d}{\lambda_V} + \boldsymbol{\mu}_V^T \boldsymbol{\mu}_V)).$$

Hence, the expressions for the circularly dependent equations have been derived.

VITA

VITA

Debasmit Das was born in Kolkata, India on September 10, 1992. He received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Roorkee in 2014. Since August 2014, he has been working towards his Ph.D. degree at the School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana. As a graduate student, he was a research assistant in the Assistive Robotics Technology Laboratory (ARTLab) and a teaching assistant at the School of Electrical and Computer Engineering. He is a student member of IEEE and he also regularly reviews machine learning articles for IEEE, ACM and Springer journals. His research interest is in label-efficient learning, especially in transfer learning problems like domain adaptation, few-shot learning and zero-shot learning.