NOVEL OPTIMIZATION MODELS TO GENERALIZE DEEP METRIC LEARNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

YETİ Z. GÜRBÜZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENG.

AUGUST 2022

Approval of the thesis:

NOVEL OPTIMIZATION MODELS TO GENERALIZE DEEP METRIC LEARNING

submitted by **YETİ Z. GÜRBÜZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Eng. Department, Middle East Technical University** by,

Prof. Dr. Halil KALIPÇILAR
Dean, Graduate School of **Natural and Applied Sciences** ______________

Prof. Dr. İlkay ULUSOY
Head of Department, **Electrical and Electronics Eng.** ______________

Prof. Dr. A. Aydın ALATAN
Supervisor, **Electrical and Electronics Eng.** ______________

Examining Committee Members:

Prof. Dr. Uğur HALICI
Electrical and Electronics Engineering, METU ______________

Prof. Dr. A. Aydın ALATAN
Electrical and Electronics Engineering, METU ______________

Prof. Dr. S. Serdar KOZAT
Electrical and Electronics Engineering, Bilkent University ______________

Assist. Prof. Dr. R. Gökberk CİNBİŞ
Computer Engineering, METU ______________

Assist. Prof. Dr. Hüseyin ÖZKAN
Electronics Engineering, Sabancı University ______________

Date: 24.08.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Yeti Z. Gürbüz

Signature :

ABSTRACT

Novel Optimization Models to Generalize Deep Metric Learning

Gürbüz, Yeti Z.
Ph.D., Department of Electrical and Electronics Eng.
Supervisor: Prof. Dr. A. Aydın ALATAN

August 2022, 128 pages

Deep metric learning (DML) aims to fit a parametric embedding function to data of semantic information (e.g. images) so that l2-distance between embedded samples is low whenever they share similar semantic entities. An embedding function of such behavior is attained by minimizing empirical expected pairwise loss that penalizes inter-/intra-class proximity violations in embedding space. Proxy-based methods which use a learnable embedding vector per class in their loss formulation are state-of-the-art. We first address characterizing generalization error of proxy-based methods. We reformulate DML as a chance-constrained optimization problem and through careful theoretical analysis, we show that DML with better generalization guarantees can be achieved by iteratively minimizing a proxy-based loss and re-initializing proxies with embeddings of new samples. Second, we consider critical desideratum for DML: generalization to unseen data. We analyze global average pooling (GAP) which is an effective architectural choice to aggregate information in DML. With theoretical and empirical supports, we explain effectiveness of GAP by considering each feature vector as representing a different semantic entity and GAP as a convex combination of them. Following this perspective, we generalize GAP and propose a learnable generalized sum pooling method (GSP) improving GAP with two distinct abilities: i) the ability to choose a subset of semantic entities, effectively learning to ignore nuisance information, and ii) learning the weights corresponding to the importance of each entity. We further propose a zero-shot loss to ease the learning of GSP. We show the effectiveness of our contributions with extensive evaluations on 4 popular DML benchmarks.

Keywords: metric learning, alternating projections, feature selection, zero-shot loss

ÖZ

DERİN METRİK ÖĞRENMEYİ GENELLEYEN YENİLİKÇİ OPTİMİZASYON MODELLERİ

Gürbüz, Yeti Z.
Doktora, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. A. Aydın ALATAN

Ağustos 2022, 128 sayfa

Derin metrik öğrenme (DMÖ), imge gibi anlamsal bilgi içeren verilere parametrik bir fonksiyonun, fonksiyonun değerleri arasındaki l2-uzaklığı anlamsal yakınlığı yansıtacak şekilde yakıştırılmasını amaçlar. Bu davranışa sahip fonksiyonlar görüntü uzayındaki ikişerli sınıf-içi/-dışı yakınlık kısıtlarının ihlalini cezalandıran ampirik beklenen kayıp değerlerinin en-küçültülmesi ile hesaplanır. Kayıp formülasyonlarında her sınıf için öğrenilebilir görüntü vektörü içeren vekil-tabanlı yaklaşımlar en iyilerdir. Bu çalışmada öncelikle vekil-tabanlı yaklaşımların genelleştirme hatalarının karakterizasyonu incelenmiştir. DMÖ bir şans-kısıtlı en-iyileme problemi olarak yeniden tanımlanmıştır. Özenli teorik analiz ile yinelemeli olarak vekil-tabanlı kayıpların en-küçültülmesi ve vekillerin yeni örnekler ile ilklendirilmesinin DMÖ için daha iyi genelleme sağlayabileceği gösterilmiştir. İkincil olarak, DMÖ için elzem gereklilik olan eğitim-dışı sınıflar üzerine genelleme çalışılmıştır. Sıkça kullanılan etkin bir bilgi ortaklama yöntemi olan bütünsel ortalama (BO) incelenmiştir. Teorik ve deneysel bulgular ile BO'nun etkinliği her bir öznitelik vektörünün anlamsal bir olguyu temsil etmesi ve BO'nun bu olguların dışbükey bileşimi olmasıyla açıklanmıştır. Bu kapsamda BO, öğrenilebilir bir ortaklama yöntemi ile genelleştirilmiştir. Önerilen yöntem BO'ya iki ayrık özellik kazandırmaktadır: i) istenmeyen bilgiyi atmayı öğrenerek anlamsal olguların altkümesini seçebilme, ii) her olgunun önemini gözeten ağırlıkları öğrenebilme. Ek olarak, önerilen yöntemin eğitim-dışı sınıflar için genelleyebilmesine yönelik yenilikçi bir ceza fonksiyonu biçimlendirilmiştir. Bu tez çalışması kapsamında sunulan katkıların etkinliği yaygın kullanılan 4 adet referans veri-seti üzerinde yapılan kapsamlı deneysel çalışmalar ile gösterilmiştir.

Anahtar Kelimeler: metrik öğrenme, yinelemeli izdüşürme, öznitelik seçimi, sıfır-yardımlı tahmin

better late than sorry

ACKNOWLEDGMENTS

I inform that there have been many people around me during my graduate studies and these are the people who all make it happen for me to present a thesis study I am comfortable with. Unfortunately, only few of them can be mentioned here due to lack of space.

First, I express my sincere thanks to my supervisor, Prof. Dr. A. Aydın Alatan for his constant support and trust in me. I would not be aware of many research areas without his guidance and leading. I am grateful for his effort to provide me with extra funds via research and development projects. He has not only made my life more comfortable and suitable for research but also given me opportunity to be productive in my very early academic life. I have always felt privileged to be a graduate student of such a brilliant person and be one of the members of his research laboratory.

Foremost, I offer my greatest and sincere thanks to love of my life, Melis Özateş Gürbüz, for her support in both personal and academic life. She makes me rise to my feet every time I fall by believing me so much like nobody ever does. She has been with me all the time with patience and love. Especially, when I was stuck, her support and encouragement triggered my enthusiasm to move on towards finishing my studies. Additionally, expanding my vision towards optimization world and her discussions about statistics are just a few of the many contributions of her to my academic life. There is no doubt that without her I would not be where I am today.

I also express my gratitude to current and past members of our research laboratory. I offer special thanks to Emrecan Batı for his teachings and assistance about almost everything, listening me patiently every time I would like to talk about my progress in my studies and the academic discussions we have made. I thank Oğul Can for encouraging me by mentioning his trust in my knowledge on every

occasion, for being a good companion during the sleepless nights when we were preparing our paper, and for picking jungle always while I was going mid. I thank Ece Selin Böncü for her friendship and partnership during the teaching assistant works we have conducted together, and more importantly for her kind support for my extracurricular activities. I give my sincere thanks to former members, Dr. Erhan Gündoğdu, for sharing his know-how and experience with me for my studies, and Dr. Ozan Şener, for his contributions to my thesis study by widening my theoretical landscape through many discussions we have made.

No thank is enough for my family's support. My parents, Nesrin - Başar Gürbüz, and my sister, Börte Gürbüz Özgür, as well as my parents-in-*love* Mediha-Selim Özateş have made everything all possible with their endless support. They have put their ultimate effort into my academic life since the very first day and always reminded me of not worrying about anything else but my studies. I can say that our parents especially have carried the weight of this thesis study on their shoulders as much as me and Melis have done.

Last, but not least, I give my sincere thanks to my thesis monitoring committee members Prof. Dr. Uğur Halıcı and Dr. R. Gökberk Cinbiş for their constructive thorough detailed feedback after every progress meeting to improve the quality and presentation of my studies.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xii

LIST OF TABLES xvi

LIST OF FIGURES xvii

LIST OF ABBREVIATIONS xxi

CHAPTERS

1 INTRODUCTION 1

  1.1 Problem Formulation 2

  1.2 Evaluation of DML Methods 3

  1.3 Summary of SOTA Approaches and Our Motivation 4

  1.4 Contributions 5

  1.5 The Outline of the Thesis 6

2 RELATED WORK 9

  2.1 Metric Learning 9

    2.1.1 Ranking Losses in DML 10

    2.1.2 Pair Mining and Sample Generation in DML 12

Pair mining. 12
Sample generation. 13
2.1.3 Proxy-based DML 13
2.1.4 Enhancing the Embedding Vectors 14
Ensemble methods. 14
Enhancing the loss gradient. 14
2.1.5 Disentangling Features 15
2.1.6 Characterizing Generalization Bounds 16
2.2 Other Related Work 16
2.2.1 Feature Pooling Techniques 16
2.2.2 Optimal Transport Based Operators 19
2.2.3 Zero-shot and Few-shot Learning 19
3 CHANCE CONSTRAINED PROGRAMMING FOR DEEP METRIC LEARNING 21
3.1 Preliminaries 22
3.2 Method 23
3.2.1 Chance Constrained Formulation of Metric Learning . . . 24
3.2.2 Reducing Chance Constraints 26
3.2.3 Solving the Feasibility Problem 28
3.2.4 Implementation Details 32
3.3 Experimental Work 33
3.3.1 Proof of the Concept 33
3.3.2 Deep Metric Learning Experiments 35

3.3.2.1 Setup 35

3.3.2.2 Results 41

3.3.2.3 Ablations 44

4 IMPLICATIONS OF GLOBAL AVERAGE POOLING 51

4.1 Preliminaries 52

4.2 Analysis of Global Average Pooling 53

4.2.1 Classification Problem 54

4.2.2 Metric Learning Problem 55

4.3 Empirical Validation 60

4.4 Discussion 63

5 GENERALIZED SUM POOLING FOR DEEP METRIC LEARNING 65

5.1 Preliminaries 67

5.2 Method 68

5.2.1 Generalized Sum Pooling as a Linear Program 68

5.2.2 Generalized Sum Pooling as a Differentiable Layer 70

5.2.3 Cross-batch Zero-shot Regularization 72

5.2.4 Implementation Details 75

5.3 Experiments 75

5.3.1 Synthetic Study 75

5.3.2 Cifar Collage Experiments 77

5.3.3 Evaluation of Zero-shot Prediction Loss 79

5.3.4 Deep Metric Learning Experiments 79

5.3.4.1 Setup 79

5.3.4.2 Results 84

5.3.4.3 Ablations 88

5.3.4.4 Computational Analysis 89

5.3.4.5 Hyperparameter Selection 91

5.3.5 Comparison to Other Pooling Alternatives 93

6 SUMMARY AND CONCLUSIONS 97

6.1 Summary 97

6.2 Conclusion 99

6.3 Future Directions 101

REFERENCES 103

A PROOFS FOR CHAPTER 3 117

A.1 Proof for Lemma 3.2.1 117

A.2 Proof for Proposition 3.2.1 118

A.3 Proof for Proposition 3.2.2 119

B PROOFS FOR CHAPTER 5 123

B.1 Proof for Claim 5.2.1 123

B.2 Proof for Proposition 5.2.1 123

B.3 Proof for Proposition 5.2.2 125

VITA 127

LIST OF TABLES

TABLES

Table 3.1 Comparison with the existing methods for the retrieval task on SOP, InShop, CUB, Cars 42

Table 3.2 Comparison with the existing methods for the retrieval task in conventional experimental settings with BN-Inception and ResNet50 backbones where superscripts denote embedding size. Red: the best. Blue: the second best. Bold: previous SOTA. †*Results obtained from* [1]. 44

Table 3.3 Total steps of training in SOP and In-shop 48

Table 5.1 Comparison with the existing methods for the retrieval task on SOP, InShop, CUB, Cars. 85

Table 5.2 Comparison with the existing methods for the retrieval task in conventional experimental settings with BN-Inception and ResNet50 backbones where superscripts denote embedding size. Red: the best. Blue: the second best. Bold: previous SOTA. †*Results obtained from* [1]. 87

Table 5.3 Effects of the two components: Zero-shot prediction loss (ZSP) and Generalized Sum Pooling (GSP) 88

Table 5.4 Initial 2-level fractional factorial screening experiments for hyperparameter tuning (conducted in CUB). 91

LIST OF FIGURES

FIGURES

Figure 3.1 Simple illustration of our chance constrained DML formulation over two sets where each set's elements are the embedding function parameters, *i.e.*,θ, that yield an embedding space in which the distances to anchor samples assess the semantic dissimilarity with high probability (*i.e.*,satisfying chance constraints). We consider the parameters of the desired embedding function, *i.e.*,$f(\cdot;\theta^*)$, to lie in the intersection of such sets. We show that to each such set, there corresponds a proxy-based DML solution. Hence, we solve DML as a set intersection problem via solving multiple proxy-based DML problems. 24

Figure 3.2 Illustration of our method (CCP) and the geometry of the embedding space before, (a), and after, (b), our method (through iterations 1-4), where boxes are the converged proxies and the circles are the next proxies as the result of k-Center. In proxy-based DML, proxies are coalesced into one whereas with CCP, we have diverse proxies, resulting reduced covering radius. 33

Figure 3.3 The geometry of the embedding space before, (a), and after, (b), our method (through iterations 1-3), relating how the generalization efforts in training domain transfer to the geometry of test domain on CUB dataset with Contrastive+CCP. We use 2-D TSNE embeddings of the validation data in the visualization, in which we report MAP@R, average covering radius and average inter-class pairwise distances. 34

Figure 3.4 Summary of relative improvements 43

Figure 3.5 Bayesian search on λ-$\#proxy$ space 45

Figure 3.6 Analysis of the relation between the number of proxies and the pool size used for the proxy selection on CUB (left) and Cars (right) dataset with C2-CCP. 46

Figure 3.7 Impact of alternating proxies (a) and typical distribution of the steps per projection problem (b). 47

Figure 3.8 Analysis of batch size dependence of the performance on CUB (left) and Cars (right) dataset with C2-CCP. 48

Figure 4.1 Visualization of Proposition 4.2.1: equivalence of global average pooling to convex combination of prototype vectors corresponding to semantic entities. Once local patches are embedded to a vector space according to the semantic meaning, collection of local patches captures the latent semantic representation of the image. 58

Figure 4.2 Empirical illustration of the local feature distribution and feature prototypes of Cifar10 dataset on the 2D embedding space. We observe that prototypes correspond to some semantic entities. 59

Figure 4.3 Illustration of class convex hulls where the vertices are the active prototypes for the corresponding class. Points are colored with respect to class labels. We observe overlapping class convex hulls, meaning that the samples of different classes are likely to be mapped close. 60

Figure 4.4 GAP and PCC embedding of an image from *car* class. . . . 61

Figure 4.4 (cont'd) GAP and PCC embedding of an image from *dog* class. . . . 61

Figure 4.4 (cont'd) GAP and PCC embedding of an image from *car* class. For sample images of several classes (*dog, car, horse*), we plot their GAP embedding as well as the PCC embedding where each local feature is assigned to its closest prototype and convex combination of the prototypes yields the global embedding vector. Combination coefficients are proportional to the amount of local features assigned to the prototypes. Empirically validating our claims on the behavior of GAP, the distance between the two representations are bounded by the covering radius of the prototypes. 62

Figure 4.5 The big picture showcasing our analysis on the behavior of GAP. *A method* that is able to control the learning of the prototypes as well as their convex combination weights can yield non-overlapping class convex hulls, meaning that the samples of different classes are mapped apart as desired. 63

Figure 5.1 Effect of global pooling of the feature map and our method. 65

Figure 5.2 10x10x3 feature map (top-left) with 5x5 reddish and bluish features to be pooled and the resultant pooling weights (higher the darker) of different problems with red $(1, 0, 0)$ and blue $(0, 0, 1)$ prototypes, ω. 70

Figure 5.3 Sketch of the method. 74

Figure 5.4 GAP (a) vs GSP (b) in aggregating features, where tokens denote learned embedding vectors and samples are obtained by aggregating them. 76

Figure 5.5 Visualizations for Cifar Collage dataset experiments: (a) Illustration of a sample generation for Cifar Collage dataset. (b) Evaluation on Cifar Collage dataset and on the right; sample train and test images with their attention maps in terms of pooling weights. *Distilled* denotes baseline performance on non-collage dataset (*i.e.*,excluding the shared classes). 77

Figure 5.6 Comparing the distributions of the learned 8 prototypes across classes of Cifar10 dataset with and without $\mathcal{L}_{ZS}$. Pooling weights are coloured according to the dominant prototype at that location. 78

Figure 5.7 Summary of relative improvements recorded in Tab. 5.1. . 86

Figure 5.8 Effect of μ in CUB dataset with C2 loss. Shaded regions represent $\mp std$. 89

Figure 5.9 Comparing closed-form gradient with automatic differentiation through analyzing the effect of k on computation in CUB dataset with C2 loss. Shaded regions represent $\mp std$. 90

Figure 5.10 Searching $m - \varepsilon$ space in CUB dataset for the parameters m: number of prototoypes and ε: entropy smoothing coefficient. We fix $\mu = 0.3$ and $\lambda = 0.1$. 92

Figure 5.11 Effect of m in SOP and InShop datasets once we fix $\varepsilon = 5$ for Contrastive (C2) and $\varepsilon = 0.5$ for Proxy NCA++ (PNCA). . . 92

Figure 5.12 Evaluation of the feature pooling methods on (a) Ciffar Collage and (b) CUB dataset. 93

LIST OF ABBREVIATIONS

BOVW	Bag-of-visual-words
CCP	Chance Constrained Programming
CNN	Convolutional Neural Network
DML	Deep Metric Learning
EMA	Exponential Moving Average
GAP	Global Average Pooling
GMP	Global Max Pooling
GSP	Generalized Sum Pooling
MAP@R	Mean Average Precision at R
MDS	Multi-dimensional Scaling
MMD	Maximum Mean Discrepancy
OT	Optimal Transport
P@1	Precision at 1
P@R	Precision at R
PCC	Prototype Convex Combination
R@1	Recall at 1
SOTA	State-of-the-Art
VLAD	Vector of Locally Aggregated Descriptors
XBM	Cross-batch Memory
ZSL	Zero-shot Learning

CHAPTER 1

INTRODUCTION

Typical approaches of many machine learning applications (*e.g.* visual object tracking [2], image captioning [3], object detection [4], semantic segmentation [5] *etc.*) implicitly or explicitly build on some notion of semantic similarity of the input data. Distance metric learning is the problem of finding a proper function that satisfies metric axioms and assesses the semantic dissimilarity of the data samples from its domain. Being the core of many vision tasks, distance metric learning for images is of ever-growing interest and has been extensively studied [6, 7].

In modern approaches, image distance metric learning is generally realized by learning the parameters of an embedding function so that the semantically similar samples are embedded to the small vicinity in the representation space as the dissimilar ones are placed relatively apart in the Euclidean sense (*i.e.*,$l2$ distance). Such a task is termed *deep metric learning* (DML).

DML problem is attempted mostly by contrastive learning which is built on tailoring a loss function penalizing/encouraging pairwise proximity based on $l2$ distance of the sample pairs. Contrastive learning can be supervised (*i.e.*,with labeled data) or unsupervised. The labelling can be as weak as binary labels indicating whether the pairs are of the same class. The unsupervised setting is typically converted to self-supervised [8] via data augmentation where a positive sample (*i.e.*,sample of the same class) is generated by applying some transformations to the original sample. In supervised setting, the function parameters are learned through minimizing the empirical expected loss with pairs sampled from a class-labelled training dataset.

Once we employ supervised training with class-labelled dataset, DML problem seems to be reduced to a classification problem. With that being said, DML is a more restrictive problem in which we do not only deal with categorizing the images but also capturing the relative orderings of the images from the same category. Moreover, one desired behavior for the embedding function is generalization to unseen classes, namely, *zero-shot* performance. In that manner, DML is well-suited for the tasks that are built on nearest-neighbour classification, *e.g.* content based image retrieval. Therefore, if we solely solve a classification problem to obtain the embedding function, then such a function will probably be a sub-optimal one for DML-related tasks.

In this dissertation, we address DML in a supervised setting. Upon comprehensive study of the literature, we focus on theoretically explaining *how* or *why* the existing approaches succeed. In particular we employ mathematical modeling (*i.e.*,optimization models) to formulate general frameworks whose corner cases are the typical approaches in the literature. Accordingly, we try to discover rooms for improvement for the generalization of DML methods. To this end, we propose a mathematical programming for DML, as well as two methods for the embedding function. In the following sections, we formally introduce DML problem and motive our study upon a short review of state-of-the-art (SOTA) approaches.

1.1 Problem Formulation

In this section, we formally define the problem of DML and set up the base notation for the rest of the thesis.

In typical c-class supervised DML, we consider the data distribution $p_{\mathcal{X}\times\mathcal{Y}}$ over $\mathcal{X}\times\mathcal{Y}$ where $\mathcal{X}$ is the space of data points (*e.g.* images) and $\mathcal{Y} = [c]$ is the space of labels with $[n] \coloneqq \{1, \ldots, n\}$. Given independent and identically distributed (*iid.*) samples from $p_{\mathcal{X}\times\mathcal{Y}}$ as $\{(x_i, y_i)\}$, deep metric learning problem aims to find the parameters θ of an embedding function $e(\cdot;\theta)$: $\mathcal{X} \to \mathbb{R}^d$ such that the Euclidean distance in the space of embeddings is consistent with the label information where d is the embedding dimension. More specifically, the parametric distance $\|x_i - x_j\|_{e_\theta} \coloneqq \|e(x_i;\theta) - e(x_j;\theta)\|_2$ is small, whenever $y_i = y_j$, and large, if $y_i \neq y_j$. In order to enable learning, this requirement is represented via loss function $l((x_i, y_i), (x_j, y_j); \theta)$ of pairwise distance (*e.g. contrastive* [9] $l((x_i, y_i), (x_j, y_j); \theta) = \max\{0, \mathbb{1}_{y_i=y_j}(\|x_i - x_j\|_{e_\theta} - \beta) + \alpha\}$ where $\mathbb{1}_A$ is an indicator function whose components are equal to 1, whenever A is true, and equal to -1 otherwise).

The typical learning mechanism is gradient descent of an empirical risk function defined over a batch of data points $B \sim p_{\mathcal{X}\times\mathcal{Y}}$. To simplify notation throughout the writing, we will use $b = \{b(i) \mid x_i, y_i \in B\}_i$ to index the samples in a batch. Then, the typical empirical risk function is defined as:

$$\mathcal{L}_{DML}(b;\theta) \coloneqq \tfrac{1}{|b|^2} \sum_{i \in b} \sum_{j \in b} l((x_i, y_i), (x_j, y_j); \theta)\,. \tag{1.1.1}$$

In this thesis, we first consider improving the generalization performance in terms of the expected loss:

$$\mathcal{L}_{DML}(\theta) = \mathbb{E}_{p_{\mathcal{X}\times\mathcal{Y}}}\left[\ell((x, y), (x', y'); \theta)\right]. \tag{1.1.2}$$

We then address unseen class generalization of the embedding function $e(\cdot;\theta)$.

1.2 Evaluation of DML Methods

DML methods are typically evaluated on image retrieval task, where there are the *query* and the *reference* images. A query is an image for which similar images are to be retrieved, and the references are the images in the searchable database. The well-accepted evaluation metrics are *precision at 1* (P@1), *precision at R* (P@R), and *mean average precision* (MAP@R) at R, where R is defined for each query and is the total number of true references as the query. P@1 sometimes referred as *recall at 1* (R@1). We next define the aforementioned metrics.

P@1: We consider the nearest reference to the query. The score for that query is 1 if the reference is of the same class, 0 otherwise. Average over all queries gives P@1 metric. Albeit used in common, P@1 is a rather greedy metric to assess the quality of the embedding space geometry, since it only considers the nearest neighbour retrieval performance.

P@R: For a query, i, we find R_i nearest references to the query and let r_i be the number of true references in those R_i-neighbourhood. The score for that query is $\mathrm{P@R}_i = {}^{r_i}/_{R_i}$. Average over all queries gives P@R metric, *i.e.*,$\mathrm{P@R} = \frac{1}{n} \sum_{i\in[n]} \mathrm{P@R}_i$, where n is the number of queries.

MAP@R: For a query, i, we define $\mathrm{MAP@R}_i := \frac{1}{R_i} \sum_{i\in[R_i]} P(i)$, where $P(i) = \mathrm{P@R}_i$ if i^{th} retrieval is correct or 0 otherwise. Average over all queries gives MAP@R metric, *i.e.*,$\mathrm{MAP@R} = \frac{1}{n} \sum_{i\in[n]} \mathrm{MAP@R}_i$, where n is the number of queries. MAP@R is shown to better assess the clustering performance in the embedding space [6]. Hence, it is recently considered as a less noisy evaluation metric to evaluate the methods.

1.3 Summary of SOTA Approaches and Our Motivation

Primary thrusts in DML include *i*) tailoring pairwise loss terms [6] that penalize the violations of the desired intra- and inter-class proximity constraints, *ii*) pair mining [7] to deliberately select mini-batches for gradient updates, *iii*) generating informative samples [10–13] to improve unseen class generalization, and *iv*) augmenting the mini-batches with virtual embeddings, called *proxies* [14, 15] to enhance the gradient updates with more holistic information. To improve generalization; learning theoretic ideas [16,17], intra-batch feature aggregation [1] and further regularization terms [18, 19] are utilized. To go beyond of a single model, ensemble [20–22] and multi-task based approaches [23, 24] are also used.

Such advances report their improvements to previous SOTA with respect to R@1 metric. Recent independent studies [6, 7] reveal that the evaluation protocol of the most methods might fail to reflect the true order of the improvements due to both the experimental setting and the R@1 evaluation metric. Indeed, reevaluation of the existing SOTA using P@R and MAP@R metrics showcase that the relative improvements over the previous SOTA differ from the that of R@1 and the ranking of the methods with respect to MAP@R metric changes.

Proxy-based methods are observed to be SOTA especially in large-scale problems (in terms of $\#classes$).

We observe that the motivation of using proxies is mostly coming from intuition and that intuition is exploiting more pairs in the computation of the empirical loss for the gradient update. In other words, introduction of proxies is based on engineered solutions and the motivation is seemingly lacking a proper theoretical explanation. For instance, increasing the proxies [25] is also exploited and shown to improve R@1 performance while later shown to degrade MAP@R [6]. Nevertheless, there lacks a theoretical answer to a critical question *"How does increasing proxies help?"*.

Our first motivation is attempting the question above and we find that characterizing generalization performance of proxy-based DML can be a decisive step towards theoretically addressing that question. To this end, we approach DML differently by posing it as a feasibility problem of chance constraints. We then propose a mathematical programming method that can be applied with variety of existing DML losses.

Our formulations are based on test and training samples to be drawn from the same distribution. However, the intended behaviour for a trained embedding function is generalization to unseen classes (*i.e.*,*zero-shot* performance). We empirically observe that the generalization in training domain indeed transfer to the test domain. Then, another critical question we want to answer is *"How does embedding function transfer its training?"* In that manner, we study the implications of local feature aggregation which is a common component in embedding functions of DML methods. Building on such implications, we propose a learnable feature aggregation layer with a zero-shot prediction constraint. Again, we propose a method that can be applied with existing DML methods.

1.4 Contributions

As the result of our study on characterizing the generalization performance of proxy-based DML methods, our contributions are as follows:

- We reformulate DML as a chance constrained feasibility problem.
- We expand on the discussions of the works on the generalization bounds to characterize generalization of proxy-based DML as well as to formally write the feasibility problem as a set intersection to be solved by iterative projections.
- We show that solution of a proxy-based DML indeed corresponds to a solution of some chance constraints and thus, we build a bridge between proxy-based methods and the set intersection solution suggested by the theory.
- We introduce a novel way to utilize arbitrary number of proxies per class.

As the result of our study on the implications of local feature aggregation on the global embedding vectors, our contributions are as follows:

- We show that local feature aggregation with global average pooling (GAP) corresponds to a convex combination of particular prototype vectors of some semantic entity.
- We introduce a general formulation for weighted sum pooling.
- We formulate local feature selection as an optimization problem which admits closed form gradient expression without matrix inversion.
- We propose a meta-learning based zero-shot regularization term to explicitly impose unseen class generalization to the DML problem.

1.5 The Outline of the Thesis

We discuss the works that are most related to ours in Chapter 2 as well as how we differ from them. We reformulate DML problem as a chance constrained optimization in Chapter 3 and propose a mathematical programming method to solve it. We provide the related empirical study to validate our claims and compare with the SOTA within the chapter. Chapter 4 is devoted to interpreting the behavior of global average pooling and its role in unseen class generalization. Building on the implications of Chapter 4, we formulate a learnable version of weighted average pooling in Chapter 5 and propose an algorithm to learn it with zero-shot prediction constraint. We also provide the related empirical study to validate our claims and compare our method with the SOTA in Chapter 5. Finally, we summarize our work in Ch. 6 and draw conclusions from the empirical studies.

CHAPTER 2

RELATED WORK

In this chapter we discuss the works that are related to ours. Briefly, we mostly present deep metric learning literature and how we differ from the existing approaches. We also discuss some literature from *zero-shot* and *few-shot learning* owing to their relation to our work. Similarly, we recapitulate feature pooling mechanisms from image retrieval domain and provide a short review for the *optimal transport* based selection operators, since our feature aggregation method shares similar concepts.

2.1 Metric Learning

We restrict ourselves to the distance metric learning problem which is posed as learning the parameters, θ, of an embedding function, $e(\cdot;\theta)$, so that the parametric distance, $\| \cdot - \cdot \|_{e_\theta}$, between the data samples reflects their semantic dissimilarity. Pioneer metric learning methods consider e as a linear mapping [26–28] that later inspire most of state-of-the-art (SOTA) frameworks in which e is a nonlinear mapping realized by deep neural networks. Such methods that use a deep neural network, in particular convolutional neural network (CNN), as the feature extraction backbone are termed deep metric learning (DML) methods. We cover the advances in supervised DML. Unsupervised DML [8] mostly inherits the loss functions and feature extraction strategies from supervised DML. They differ in that unsupervised (*i.e.*,self-supervised) DML exploit data augmentation techniques to generate positive samples which are the transformed version of the original sample. Thus, in the rest of the section, we focus on DML approaches for supervised setting.

2.1.1 Ranking Losses in DML

Learning a proper linear mapping for the parametric distance is initially formulated as a convex optimization problem in [26]. The objective is minimizing the parametric distances among the samples of the same classes. To prevent null mappings, a constraint that enforces mapping of the samples from different classes to be at least separated by some margin is added to the formulation. Moving that constraint to the objective via hinge loss [29] results in the well-known contrastive loss:

$$\ell_{cntrv}((x_i, y_i), (x_j, y_j); \theta) = \max\{0, y_{ij}\, \|x_i \text{ - } x_j\|_{e_\theta}\} + \max\{0, y_{ij}\, (\|x_i \text{ - } x_j\|_{e_\theta} - \varepsilon)\}, \tag{2.1.1}$$

where we let $y_{ij} \in \{\text{-}1, 1\}$ denote $\mathbb{1}_{y_i = y_j}$ and ε is the desired margin. Minimizing the contrastive loss embeds the samples from the same class in a neighborhood as small as possible. This approach ignores the intra-class variations of the classes and results in internally diverse classes to be treated equally as internally similar classes. Triplet loss introduced in [27] and popularized in deep metric learning frameworks [30, 31] alleviates this problem by constraining the distance to any positive sample to be at least some margin smaller than the distance to any negative sample for each sample. In particular, for a sample, x_i, we denote its positive sample as x_{j^+} such that $y_i = y_{j^+}$ and denote its negative sample as x_{j^-} such that $y_i \neq y_{j^-}$. Then, we write the triplet loss as:

$$\ell_{triplet}((x_i, x_{j^+}, x_{j^-}); \theta) = \max\{0, \|x_i \text{ - } x_{j^+}\|_{e_\theta} - \|x_i \text{ - } x_{j^-}\|_{e_\theta} + \varepsilon\}, \tag{2.1.2}$$

Minimizing triplet loss entails deliberately selection of the triplets to have nonzero loss terms. Thus, either large batch size or mining for exemplars violating the triplet constraint is required [30]. Such an effort makes the computation of the triplet loss is less attractive than of the contrastive loss. Margin-based loss is introduced in [9] to provide the flexibility in the distribution of the classes in the embedding space without using triplets as the exemplars. It expresses the margin constraint of the triplet loss as separate loss terms of the distances between positive and negative sample pairs by relaxing the constraint of the contrastive loss on the positive pairs:

$$\ell_{margin}((x_i, y_i), (x_j, y_j); \theta) = \max\{0, y_{ij}(\|x_i \text{ - } x_j\|_{e_\theta} - \beta) + \alpha\}, \tag{2.1.3}$$

where α controls the separation margin and β is a trainable parameter for the boundary between positive and negative pairs. For fixed β (*i.e.*,non-trainable), the margin loss is indeed the generalized version of the original contrastive loss.

The contrastive, triplet and margin-based loss terms are the simplest forms of the pairwise distance ranking based loses. In these losses, only 2 or 3 data samples contribute to the loss terms. On the other hand, all negative samples within the batch can also be integrated to a single triplet loss term in lifted structured loss [32]. It reformulates the triplet loss with the triplets formed by hard negative mining within the batch for each positive pair. Using log-sum-exp expression as the approximation of max operator in the mining operation, the lifted structured loss term for a positive pair gets the contribution of all the negative samples in the batch. Similarly, proceeding approaches utilize smoothed versions of these simple losses by replacing hinge loss with log-sum-exp [33] or soft-max [34, 35] expressions. Similarly, ranking among more samples via soft-batch-mining [34–37] are addressed to exploit in-batch tuples with non-trivial settings.

In a different aspect, angular loss [38] that constraints the local geometry of the samples in the embedding space is proposed to better exploit the relation among triplets. Ranking in the quadruplets is also studied [39–42] to improve the structure of the embedding space by considering relation among more samples.

The motivation of using the relations among more samples leads to softmax-classifier-based losses [34, 43] and clustering-based losses [44–46]. For the latter, optimization of the clustering-based losses deviates from the typical nonlinear programming procedure of the deep metric learning frameworks. They involve additional sophisticated computations which might be quite expensive. Differently, softmax-classifier-based losses are bath based. N-pair loss [34] makes use of half of the negative samples in the batch for a positive pair. The batch for the N-pair loss is constructed by N samples from distinct classes and N corresponding positive samples. Each positive sample is used as a negative sample for the samples from the other classes. Such a batch construction together with the N-pair loss formulation results in a sampled softmax-cross-entropy loss term for a classifier in which the embedding of the positive samples are considered as the sampled class vectors. Similar softmax-based loss is later formulated in [43] within a meta-learning framework.

In our work, we do not consider crafting a loss term. We rather relate minimizing the ranking losses to feasibility of some chance constraints on the probability of observing pairs violating desired intra-/inter-class distances. Namely, the existing loss terms follow a top-down approach to impose pairwise proximity constraints. On the contrary, we use a bottom-up formulation using chance constraints that ensure low probability of proximity violation among the samples and end up with the similar loss forms to that of contrastive loss.

2.1.2 Pair Mining and Sample Generation in DML

Pair mining. Apart from the advances in the loss terms, efficient sampling strategies to provide batch of exemplars to the learning algorithm also have a key role in the success of the DML frameworks. The aim of exemplar mining is to improve the speed of the convergence and the quality of the embeddings by providing informative exemplars which are tuples with non-trivial settings in general. Mining for such informative exemplars brings additional computational burden since vast number of exemplars exist and all the samples should be mapped to the embedding space prior to mining. To avoid such a computational burden, some frameworks performs mining in batches for *semi-hard negative* [30, 36], *hard negative* [47] and distance weighted sampling [9]. Similarly, a random subset of the dataset for semi-hard negative mining is also considered [34]. For global mining, hierarchical triplet loss framework [48] performs triplet mining through tree representation of the dataset, and smart mining method [31] exploits approximate nearest neighbor search. Class-level global mining is addressed in [49]. Global mining is further exploited for contrastive learning in [50]. The main problem with such global approaches is requirement of the mapping of the entire dataset to the embedding space periodically, which prevents scalability to the large datasets.

Sample generation. Different from the search (*i.e.*,mining) based approaches for the informative pairs, generation of synthetic hard negative samples through adversarial [51, 52] and variational [53, 54] models is addressed. The motive is to exploit a generative model to create samples such that high pairwise losses are observed when paired with the original samples. Follow-up works [10–13] address synthesizing samples from the convex combination of the embedding vectors of the samples in the batch. Unlike generative models, such approaches explicitly use the embedding space to generate synthetic samples. Mixing the samples with several strategies and combining such strategies to generate synthetic samples are studied in [13].

Our contributions in this thesis are orthogonal to such sample mining and sample generation based approaches. Hence, our methods can be applied with such methods. That being said, our empirical study show that we can achieve SOTA performance without sample mining or augmentation, implying that our methods seemingly alleviates the need for such enhancements.

2.1.3 Proxy-based DML

Proxy-based methods consider augmenting the mini-batch with more samples for less noisy estimate of the expected loss and circumvent the costly embedding computation to include more samples in the mini-batches. In order to approximately increase the contribution of the negative samples for a positive pair to the global extent, proxies for classes are re-introduced in [55] after its linear metric learning counterpart [28]. Proxies can be considered as the representative vectors for the set of points in the embedding space. Thus, using proxies in the loss terms implicitly takes multiple data samples into account. Learning a proxy vector per class within the deep metric learning framework is quite similar to learning the class vectors of a softmax classifier with the sampled softmax-cross-entropy loss. The slight difference in [55] is the absence of the positive pair similarity term in the normalization of softmax computation. Many later approaches consider the proxies as vectors representing embeddings of the class centers [15, 37, 56, 57] and are trained along with the embedding function parameters.

Non-trainable proxies are also exploited in [14] to gradually augment mini-batch with previously computed embeddings. In proxy-based DML, the pairwise distances are computed between the proxies and the mini-batch samples. Thus, pseudo-global dataset geometry is considered during loss computation. To better represent global geometry, multiple proxies per class are considered in [25, 58, 59] where the former two build on improving P@1[1] by fine-grained clustering of class samples to overlook intra-class variances.

In our analysis, we also align with increasing the proxies. Our work differs in that *i*) we build on reducing the probability of proximity violations (*i.e.*,improving MAP@R [1]) and *ii*) we progressively increase the proxies by relating the proxy-based DML instances. To this end, considering recently computed embeddings [14] within the batch memory can be considered as a *fuzzy* version of our method as we will discuss in the related part.

2.1.4 Enhancing the Embedding Vectors

Ensemble methods. To enhance the diversity of the semantic information embedded to the vector representations, ensemble techniques are also combined with deep metric learning framework [20–22, 60–62]. The general idea is simply concatenating the vectors from multiple embedding functions whose parameters are learned by considering different local features of the samples. Hence, better embedding space can be obtained by integrating the vectors that are specialized to different aspects of the samples.

Enhancing the loss gradient. Several works address augmenting the primary DML loss with the losses corresponding to either some regularization terms [18, 19] or sub-tasks to regularize the learning towards better embedding learning. Implications of $l2$ normalization in gradient updates are studied in [18] and a regularization term bounding the vector magnitudes is proposed to enable better learning. A distance ranking based regularization term is introduced in [19] to augment binary similarity based loss with some auxiliary labels to harden the

---

[1] P@1 (immediate neighbourhood) and MAP@R (global geometry) are explained in § 1.2

learning. In words, each pairwise distance is assigned to a label according to their magnitude and the distances are regularized to keep their initial value after the gradient update. Similarly, multi-task based approaches [23, 24] augments the primary DML loss with the losses corresponding to sub-tasks to regularize and harden the learning towards better embedding vectors. Different from proposing a loss term, a feature aggregation method is proposed in [1] to enable richer representations to be used in distance computing. In particular, each feature is represented as the convex combination of the other similar features within the batch.

All of the aforementioned approaches build on the global image features extracted by the global average pooling (GAP) layer. Different to them, we propose a learnable pooling method for the global feature extraction generalizing GAP, a shared component of all of the mentioned works. Hence, our work is orthogonal to all of these and can be used jointly with any of them.

2.1.5 Disentangling Features

Several methods consider that the samples used in embedding learning share some semantic entity which are not class-discriminative and thus, put particular effort on disentangling class-discriminative and class-shared features [54, 63]. In [54], global feature is decomposed into sum of class-discriminative and intra-class variant parts where the latter is modelled by a variational auto-encoder. Similarly, in [63], global feature is a concatenation of class-discriminative and class-shared features where the latter is learned through auxiliary labels obtained by clustering of the embedding space. Those methods operate on global image representation level and their effect on local features are rather explicit.

In our feature aggregation method, we also consider overlooking shared features among the classes. Different from us, the aforementioned methods design loss functions and algorithms regularizing global representations. We instead focus on interface between local and global representations. Hence, our method is orthogonal and can be combined with them.

In this sense, the methods that exploit local features explicitly through distribution matching [64, 65] are quite related to our feature aggregation method as well. Local feature matching with moments of several orders are used as a regularization in [64] and optimal transport distance [66] to match local features is used as the distance metric instead of $l2$ distance in [65].

Our formulation of feature selection resembles optimal transport formulation. Our method differs in that we stick to $l2$ as the distance metric and do not introduce costly optimal transport computation during retrieval.

2.1.6 Characterizing Generalization Bounds

Notion of robustness in learning algorithms is studied in [67] and generalization error bounds of several techniques are derived accordingly. This study is extended to metric learning setting in [68]. These works study the deviation between the expected loss and the empricial loss over the whole dataset. Differently in [69], deviation between two empirical losses, *core-set loss*, is studied and generalization error is characterized by the core-set loss to formulate an optimization objective for their active learning method. Core-set loss typically measures how much is lost when a subset of the training data is exploited. Generalization bound for metric learning is further studied in [16, 17] to analyze and suggest training strategies.

Our work expands on the theories in the aforementioned works to characterize and improve generalization bound for proxy-based DML.

2.2 Other Related Work

2.2.1 Feature Pooling Techniques

One closely related task to metric learning is query based image retrieval. The de-facto standard to address image retrieval problem has been the bag-of-visual-words (BOVW) [70] until unprecedented success of deep learning methods. The idea is extracting local feature representations of the image and aggregating such features to obtain a global representation vector that is to be used in matching the similarity of the images.

Before deep learning, hand-crafted feature extractor are employed and popular pooling techniques include max pooling, average pooling, Fisher vectors [71], vector of locally aggregated descriptors (VLAD) [72], and structural match kernels (SMK) [73], where VLAD and Fisher can be considered as first and second order moment extension of average pooling BOVW. Later, fully convolutional backbone part of CNNs pretrained on image classification task has replaced the handcrafted features and the follow-up works have focused on transferring the BOVW aggregation techniques to such CNN-based features [74]. Next, task specific fine tuning of CNN backbones is addressed to improve the feature extraction process and SOTA approaches employ pooling layers enabling end-to-end learning.

Global pooling layers. Global average pooling and global max pooling are the simplest parameter-free forms of feature aggregation enabling end-to-end learning. Global average pooling accumulates information of all the local features yet suffers from burstiness problem, *i.e.*,repeated semantic entities dominate the salient ones. Although max pooling on the other hand is more robust to burstiness, it is prone to prevent useful gradient updates from propagating to feature extractor backbone. To facilitate such a weakness of max pooling, fusion of k-max and k-min pooling is proposed in [75]. Compromising between average and max, generalized mean pooling is introduced in [76]. Similarly, generalized max pooling and democratic pooling are proposed in [77] to address contribution of more features in max pooling without suffering from burstiness.

Fisher vectors [71] equipped with correlations among the local features have been a predominant pooling method owing to its highly discriminating power. However, Fisher vectors are not suitable for end-to-end learning. To this end, trainable bilinear pooling [78] is introduced and shortly after, its generalized version compromising between bilinear pooling and mean pooling is developed in [79]. Following the attention era, enhancing the local features with inter-feature relations are as well addressed with attention mechanism in [80]. Such methods consider local feature correlations in the aggregated (pooled) image representation and hence, the resultant global representation equips richer information on the distribution of the local features. That being said, prototype learning based methods [81, 82] also aim to encode local feature distribution to global representation vector. NetVLAD [81] is such an extension of VLAD [72] pooling in trainable form. Similarly, optimal transport is adopted for feature aggregation in [82] to form ensemble of pooled features corresponding to learned prototype vectors.

Foreground-aware pooling. Although bilinear pooling methods [79] implicitly perform salient matching, they are not explicitly tailored to ignore nuisance information. Namely, all the features in the bag constitutes to global feature vector. Excluding the features corresponding to background in pooling is addressed by detection based [83, 84] and attention based [85–89] approaches. The former class builds on training with bounding box annotations and region based pooling representation [90] of the detected region. Resolving bounding box annotations, attention based methods [85–89] build on learning to mask non-discriminative features via class-label annotations. Among those, CroW [86], Trainable-SMK [88] , and CBAM [85] build on feature magnitude based saliency, assuming that the backbone functions must be able to zero-out nuisance information. Yet, such a requirement is restrictive for the parameter space and annihilation of the non-discriminative information might not be feasible in some problems. Similarly, attention-based weighting methods DeLF [87], GSoP [89] do not have explicit control on feature selection behavior and might result in poor models when jointly trained with the feature extractor [87].

Our work is mostly related the prototype based methods [81, 82] and the methods [77,83–89] that reweights the CNN features before pooling. Existing prototype methods have no feature selection mechanism and thus, all features are somehow included in the image representation. Briefly, such methods map a set of features to another set of features without discarding any. Such a representation is useful for distribution matching. On the contrary, our pooling machine effectively enables learning to select discriminative features and maps a set of features to a single feature that is distilled from nuisance information. Moreover, our method unifies attention-based feature masking practices (*e.g.* convolution, correlation, normalization) with an efficient-to-solve optimization framework and lets us do away with engineered heuristics in obtaining the masking weights (*e.g.* normalization, sigmoid, soft-plus) without restricting the solution space unlike magnitude based methods [85, 86, 88].

2.2.2 Optimal Transport Based Operators

Our feature selection and aggregation formulation has close relation to optimal transport [66] based top-k [91], ranking [92] and aggregation [82] operators. In aggregation [82], optimal transport formulation is used to select local features to be pooled according to their similarity to some anchor features. Hence, to each anchor, a vector of aggregated local features are obtained, forming ensemble of representations similar to [62].

In our work, we also perform feature selection according to similarities to some anchor features and then aggregate the features accordingly to obtain single global representation. What makes our method different is the unique way we formulate the feature selection problem. Our formulation allows computationally appealing matrix inversion free gradient computation of the selection operator unlike optimal transport plan based counterparts [93].

2.2.3 Zero-shot and Few-shot Learning

Our zero-shot regularization for local features draws inspirations from attribute based zero-shot learning (ZSL) [94–96] and meta-learning based few-shot learning [97, 98] approaches. Attribute based zero-shot learning approaches express class embeddings as the convex combination of attribute vectors which are either available from text domain [94, 96] or learned within classification framework [95]. The approaches in [95, 96] can provide attribute localization and share a similar spirit with our feature aggregation method which also enables approximate localization. However, attribute annotations must be provided for those methods whereas we exploit only class labels to extract attribute-like features.

Our method can be considered as attribute-unsupervised version of these methods. We learn class embedding and attribute vectors with a meta-learning framework similar to the differentiable convex optimization based approaches [97, 98] for few-shot learning. In meta-learning based few-shot learning, the framework *learns to learn* predictors that are useful for few-shot prediction. In words, fitting a predictor to a set of data is formulated as a differentiable layer through formulating it as a strict convex optimization problem [97, 98] (*e.g.* regression). Then, such a regression operation as a layer is augmented to the feature extraction framework, enabling learning to learn how to regress.

In this thesis, we also propose a similar meta-learning approach to enable transferring the behavior of our feature selection layer to unseen classes. Instead of few-shot setting, we propose meta-learning for zero-shot setting thanks to our feature selection formulation. In short, our feature selection formulation together with meta-learning setting enables us to introduce attribute based zero-shot predictions without explicit attribute annotations.

CHAPTER 3

CHANCE CONSTRAINED PROGRAMMING FOR DEEP METRIC LEARNING

Deep metric learning (DML) poses distance metric problem as learning the parameters of an embedding function so that the semantically similar samples are embedded to the small vicinity in the representation space as the dissimilar ones are placed relatively apart in the Euclidean sense. The typical embedding function is implemented as convolutional neural networks (CNN) for visual tasks and the parameters are learned through minimizing the empirical expected loss with possibly deliberately selected mini-batch gradient updates [6, 7]. The loss terms in the empirical loss penalize violations of the desired intra- and inter-class proximity constrains. Large-scale problems (in terms of #*classes*) suffer from the noisy estimation of the expected loss with mini-batches [6, 14, 30]. Recently, augmenting the mini-batches with virtual embeddings called *proxies* is shown to better approximate empirical loss in large-scale problems [6, 14] owing to pseudo-global consideration of the dataset during loss computation. These advances raise a critical question: *"How does increasing proxies help?"* which is partially addressed empirically with the methods exploiting multiple proxies per class [14, 25, 59].

Characterizing generalization performance of proxy-based DML can be a decisive step towards theoretically addressing that question. To this end, we approach DML differently by posing it as a feasibility problem. In particular, we consider a chance constraint for desired embedding function and relate it to the typical expected loss of DML. Using such a relation, we provide an upper bound to the generalization error of proxy-based DML. Aligned with the literature, the form of the bound suggests possible room for the improvement on the generalization performance if more and diverse proxies are considered per class. However, straightforward increase of the proxies may not help; since, *i*) proxies of the same class tend to merge [25] and *ii*) memory is prohibitive to arbitrarily increase the proxies.

To alleviate these limitations, we relate minimizer of the proxy-based DML to a feasible point of some chance constraints, and reformulate DML as finding a point at the intersection of the sets that the proxies imply. We provide a scalable algorithm using iterative projections to the individual sets to solve the problem. Each projection corresponds to a regularized proxy-based DML. Hence, we inherently increase the number of diverse proxies included in the problem. We empirically study the implications of our formulation. Evaluation of our method on image retrieval shows state-of-the-art (SOTA) performance in improving the baselines.

3.1 Preliminaries

In this section, we formally define the generalization performance of DML and extend the base notation that we set up in § 1.1.

In typical DML, we consider the set $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ with elements $z = (x \in \mathcal{X}, y \in \mathcal{Y})$ where $\mathcal{X}$ is a compact space and $\mathcal{Y} = [c]$ (*i.e.*,$\{1, \dots, c\}$) is a finite label set. We will use x (or y) to denote data (or label) component of z. We have $p_{\mathcal{Z}}$, an unknown probability distribution over $\mathcal{Z}$. Indicator of the two samples, z and z', belonging to the same class is denoted as $\iota_{y,y'} \in \{\text{-}1, 1\}$ where $\iota_{y,y'} = 1$ if $y = y'$. For indexed dataset samples, z_i and z_j, we will simply use y_{ij} in place of ι_{y_i,y_j}

We are interested in finding the parameters, θ, of an embedding function, $f(\cdot;\theta)$: $\mathcal{X} \to \mathbb{R}^{d}$[1], so that the parametric distance, $\|x \text{-} x'\|_{f_\theta} \coloneqq \|f(x;\theta) \text{-} f(x';\theta)\|_2$, between two samples, $(x, x') \sim p_{\mathcal{X}}$, reflects their semantic dissimilarity. For any pair, $(z, z') \sim p_{\mathcal{Z}}$ and embedding function, $f(\cdot;\theta)$, we associate a loss, $\ell(z, z';\theta)$,

---

[1] Different from the notation throughout the thesis, we use f to denote the embedding function instead of e.

penalizing proximity violations in the embedding image. We omit f dependency in the ℓ notation for simplicity. We are to consider minimization of the expected loss:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{z,z' \sim p_{\mathcal{Z}}} \left[\ell(z, z'; \theta) \right] \tag{3.1.1}$$

In practice, we are given a dataset of n instances sampled *i.i.d.* from $\mathcal{Z}$ as $\{z_i\}_{i \in [n]} \sim p_{\mathcal{Z}}$ where $[n] = \{1, \ldots, n\}$, and an algorithm, $\mathcal{A}_{s_1 \times s_2}$, which outputs parameters, θ, minimizing empirical expected loss with a training error, $e(\mathcal{A}_{s_1 \times s_2})$, for a given set of pairs, $\{(z_i, z_j)\}_{i,j \in s_1 \times s_2}$, from the dataset, where $s_k = \{s_k(l) \in [n]\}_{l \in [n_k]} \subseteq [n]$ is a pool of indexes chosen from the dataset, $[n]$. *i.e.*,

$$\mathcal{A}_{s_1 \times s_2} \coloneqq \arg\min_{\theta} \tfrac{1}{n_1\, n_2} \sum_{i \in [n_1]} \sum_{j \in [n_2]} \ell(z_{s_1(i)}, z_{s_2(j)}; \theta) \,, \tag{3.1.2}$$

and we formally define DML as $\mathcal{A}_{[n] \times [n]}$, *i.e.*,minimizing empirical expected loss with all possible pairs. We consider improving the generalization error of $\mathcal{A}_{s_1 \times s_2}$ which is:

$$\mathcal{L}(\mathcal{A}_{s_1 \times s_2}) = \mathbb{E}_{z,z' \sim p_{\mathcal{Z}}} \left[\ell(z, z'; \mathcal{A}_{s_1 \times s_2}) \right] . \tag{3.1.3}$$

3.2 Method

We will iteratively solve multiple proxy-based DML problems. At each problem, we re-initialize the class proxies by samples from the dataset. We relate the problems by regularizing the learned parameters to be in the close vicinity of the previous ones. In the following sections, we provide theoretical foundation behind the motivation of our method. We defer all the proofs to appendix.

We start with reformulating DML with a chance constraint. We will introduce two propositions that allow us to decompose the chance constraint into finite chance constraints. We also show minimizer of proxy-based DML satisfies some chance constraints. Hence, we link DML to finding a point in the intersection of finite sets, which we solve using iterative projections that correspond to regularized proxy-based DML problem instances.

In the formulations throughout the chapter, we rely on Lipschitz continuity of the loss function for which we refer to Lemma 3.2.1. Our method builds on improving the generalization performance on training domain. Granted that the desired behavior is generalization to unseen classes, as we will discuss in Ch. 4, the embedding vector of DML models is typically obtained by global average pooling of local CNN features. In that manner, local features can be considered as visual words. Thus, if we consider better generalization in training domain, the semantics captured by the visual words can be transferred to represent the samples from an unseen domain. Besides, our empirical studies support that the implications suggested by the formulations are quite effective.

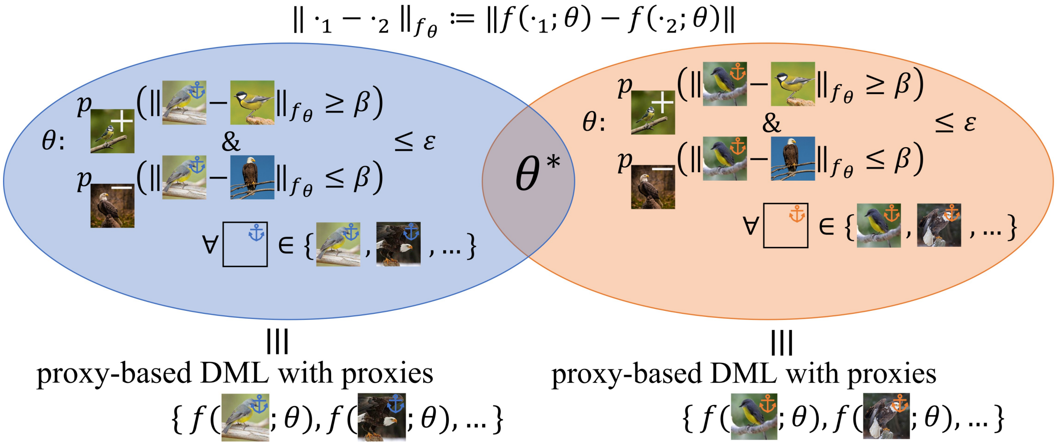

$\| \cdot_1 - \cdot_2 \|_{f_\theta} \coloneqq \|f(\cdot_1;\theta) - f(\cdot_2;\theta)\|$
θ:
p
$(\| \; - \; \|_{f_\theta} \geq \beta)$
&
$\leq \varepsilon$
p
$(\| \; - \; \|_{f_\theta} \leq \beta)$
$\forall \; \in \{ \; , \; , \ldots \}$
θ^*
θ:
p
$(\| \; - \; \|_{f_\theta} \geq \beta)$
&
$\leq \varepsilon$
p
$(\| \; - \; \|_{f_\theta} \leq \beta)$
$\forall \; \in \{ \; , \; , \ldots \}$
|||
proxy-based DML with proxies
$\{ f(\; ;\theta), f(\; ;\theta), \ldots \}$
|||
proxy-based DML with proxies
$\{ f(\; ;\theta), f(\; ;\theta), \ldots \}$

Figure 3.1: Simple illustration of our chance constrained DML formulation over two sets where each set's elements are the embedding function parameters, *i.e.*,θ, that yield an embedding space in which the distances to anchor samples assess the semantic dissimilarity with high probability (*i.e.*,satisfying chance constraints). We consider the parameters of the desired embedding function, *i.e.*,$f(\cdot;\theta^*)$, to lie in the intersection of such sets. We show that to each such set, there corresponds a proxy-based DML solution. Hence, we solve DML as a set intersection problem via solving multiple proxy-based DML problems.

3.2.1 Chance Constrained Formulation of Metric Learning

We consider the solution of the following chance constrained feasibility problem:

$$\min_{\theta} 0^{\intercal}\theta \text{ s. to } p_{z,z'\sim p_{\mathcal{Z}}}(\iota_{y,y'}(\|x - x'\|_{f_\theta} - \beta) \geqslant 0) \leqslant \varepsilon\,, \tag{3.2.1}$$

with some small ε. Namely, we want the probability of observing two samples of the same (different) class being apart (close) more than β in the embedding space being low. We write that probability as expected violation, $\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}\left[\mathbb{1}_{\iota_{y,y'}(\|x\text{-}x'\|_{f_\theta}-\beta)\geqslant 0}\right]$ where $\mathbb{1}_{(\cdot)}$ being indicator function, and bound it for $\beta \geqslant \alpha > 0$ as:

$$\begin{aligned} & p_{z,z'\sim p_{\mathcal{Z}}}(\iota_{y,y'}(\|x \text{ - } x'\|_{f_\theta} - \beta) \geqslant 0) \\ & \quad \leqslant {}^{1}\!/_{\alpha}\, \mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}\left[(\iota_{y,y'}(\|x \text{ - } x'\|_{f_\theta} - \beta) + \alpha)_+\right] \ , \end{aligned} \tag{3.2.2}$$

using Markov's inequality where $(u)_+ = \max\{0, u\}$. Note that to each value of the expectation, $e(\theta)$, there corresponds an $\varepsilon = {}^{e(\theta)}\!/_{\alpha}$ which the chance constraint satisfies. Hence, we use the expectation as the surrogate of the penalty term for the chance constraint and can redefine the aforementioned feasibility problem as the expected loss minimization in Eq. (3.1.1) with $\ell(z, z'; \theta) = (\iota_{y,y'}(\|x \text{ - } x'\|_{f_\theta} - \beta) + \alpha)_+$. In particular, we end up with minimization of the expected *generalized contrastive loss* [9].

We now consider the relaxed feasibility problem in which we consider m chance constraints conditioned on given m samples $\mathcal{S}{=}\{z_i\}_{i\in[m]} \sim p_{\mathcal{Z}}$, say *anchor samples*. In other words, we want to find $\theta \in \mathcal{C}_{\mathcal{S}}$ where:

$$\mathcal{C}_{\mathcal{S}} = \{\theta \mid p_{z\sim p_{\mathcal{Z}}}(\iota_{y_i,y}(\|x_i\text{-}x\|_{f_\theta}{-}\beta){\geqslant}0) \leqslant \varepsilon,\ \forall i{\in}[m]\} \tag{3.2.3}$$

Using expectation bounds as in Eq. (3.2.2), the unconstrained problem becomes:

$$\theta^* = \arg\min_{\theta} \tfrac{1}{m} \sum_{i\in[m]} \mathbb{E}_{z\sim p_{\mathcal{Z}}}\left[\ell(z_i, z; \theta)\right] \ . \tag{3.2.4}$$

We are particularly interested in the problem of the form in Eq. (3.2.4) owing to its relation to proxy-based methods to characterize their generalization. Prior to delving into such a relation, we first bound the deviation from the actual expectation in Eq. (3.1.1) when we solve Eq. (3.2.4) instead.

Proposition 3.2.1 *Given* $\mathcal{S}{=}\{z_i\}_{i\in[m]} \overset{i.i.d.}{\sim} p_{\mathcal{Z}}$ *such that* $\forall k{\in}\mathcal{Y}\ \{x_i | y_i{=}k\}$ *is* $\delta_{\mathcal{S}}$-*cover*[2] *of* $\mathcal{X}$, $\ell(z, z'; \theta)$ *is* ζ-*Lipschitz in* x, x' *for all* y, y' *and* θ, *and bounded by* L; *then with probability at least* $1 - \gamma$,

$$\begin{aligned} & \left| \mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}\left[\ell(z, z'; \theta)\right] - \tfrac{1}{m} \sum_{i\in[m]} \mathbb{E}_{z\sim p_{\mathcal{Z}}}\left[\ell(z_i, z; \theta)\right] \right| \\ & \qquad\qquad \leqslant \mathcal{O}(\zeta\, \delta_{\mathcal{S}}) + \mathcal{O}(L \sqrt{{}^{\log \frac{1}{\gamma}}\!/_{m}}). \end{aligned}$$

---

[2] $\mathcal{S} \subset \mathcal{S}'$ is $\delta_{\mathcal{S}}$-cover of $\mathcal{S}'$ if $\forall z' \in \mathcal{S}'$, $\exists z \in \mathcal{S}:\ \|z - z'\|_2 \leqslant \delta_{\mathcal{S}}$.

Proposition 3.2.1 gives an upper bound which is controlled by the diversity of the anchor samples defining the relaxed problem. Theoretically, such a controlled bound allows DML to be formulated as a feasibility problem of finite sets for some accepted error tolerance. In practice, the best we can do is using all the samples in the dataset as the anchor samples when defining $\mathcal{C}_{\mathcal{S}}$ in Eq. (3.2.3). Granted that the minimization of the empirical loss of Eq. (3.2.4) boils down to the classical DML in Eq. (3.1.2), it has different stochastic optimization procedure. The relaxed problem suggests sampling batch of instances rather than pairs, which yields less noisy gradient estimates with the same batch budget[3]. This intuitively explains the superior performance of the methods using batches augmented by class proxies, past embeddings or more samples (*i.e.*,larger batches), especially in large-scale problems.

3.2.2 Reducing Chance Constraints

The loss terms conditioned on anchor samples in Eq. (3.2.4) are computationally prohibitive in large-scale problems. Thus, we are interested in reducing the chance constraints, *i.e.*,anchor samples. To this end, proxy-based methods are quite related in that a proxy-based DML constitutes a superset of the feasible region of the primary DML problem in Eq. (3.2.1) as we will show shortly.

Proxy-based methods use parametric vectors, $\{\rho_i\}_{i\in[c]}$, to represent embedding of the class centers and minimize the pair losses with respect to those centers. Formally, given a dataset $\{z_i\}_{i\in[n]} \sim p_{\mathcal{Z}}$, proxy-based methods consider the following problem:

$$\min_{\theta,\rho} \tfrac{1}{n\,c} \sum_{i\in[c]} \sum_{j\in[n]} \hat{\ell}(\rho_i, z_j; \theta) \tag{3.2.5}$$

where $\hat{\ell}(\rho_i, z_j; \theta)$ is a loss term in which the pairwise distance is computed as $\|\rho_i - f(x_j;\theta)\|_2$. We can associate an algorithm, $\mathcal{A}_{s\mathsf{x}[n]}$ defined in Eq. (3.1.2), to the minimizer of Eq. (3.2.5) with $e(\mathcal{A}_{s\mathsf{x}[n]})$ training error where $s = \{s(i) \in [n] \mid f(x_{s(i)}; \mathcal{A}_{s\mathsf{x}[n]}) = \rho_i\}_{i\in[c]}$. In other words, to each proxy, we associate a sample whose embedding matches that proxy, assuming such sample exists. Hence, the minimizer of the proxy-based methods can be reformulated as the following

---

[3] Besides intiution, implied by Thm. 3 in [67] and Thm. 1 in [68].

feasibility problem with the abuse of notation $p_z = p_{z \sim p_{\mathcal{Z}}}$:

$$\min_{\theta} 0^{\intercal}\theta \text{ s. to } p_z(\iota_{i,y}(\|x_{s(i)}\text{-}x\|_{f_\theta}-\beta)\geqslant 0) \leqslant \varepsilon,\ \forall i\in[c] \tag{3.2.6}$$

where $\varepsilon = \frac{1}{\alpha}\mathcal{L}(\mathcal{A}_{s\times[n]})$ from Eq. (3.2.2) and $\mathcal{L}(\mathcal{A}_{s\times[n]})$ defined in Eq. (3.1.3) is shown to be bounded in [68]. Reformulation of proxy-based DML defines the feasibility problem in Eq. (3.2.3) with one sample per class.

We now consider more general case where we use m samples per class from the dataset, $\{z_i\}_{i\in[n]} \sim p_{\mathcal{Z}}$, to define the feasibility problem. We have m-many disjoint 1-per-class sets, $s = \cup_{k\in[m]} s_k$, where $s_k = \{s_k(i)\in[n] \mid y_{s_k(i)}=i\}_{i\in[c]}$ with $\cap_{k\in[m]} s_k = \emptyset$. We define the problem as:

$$\begin{gathered}\min_{\theta\in\cap_k \mathcal{C}_{s_k}} 0^{\intercal}\theta \quad \text{where} \quad \mathcal{C}_{s_k} = \{\theta \mid \forall i \in [c], \\ p_{z\sim p_{\mathcal{Z}}}(\iota_{i,y}\|x_{s_k(i)} \text{ - } x\|_{f_\theta} - \beta) \geqslant 0) \leqslant \varepsilon\}\end{gathered} \tag{3.2.7}$$

Solving the problem by minimizing the empirical expectation bounds in Eq. (3.2.4), we end up with an algorithm $\mathcal{A}_{s\times[n]}$ in which we are minimizing expected loss over a subset of all possible pairs. We want to characterize the generalization performance of the algorithm $\mathcal{A}_{s\times[n]}$. We consider the following bound from [69] for the generalization error:

$$\begin{aligned}\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}\left[\ell(z,z';\mathcal{A}_{s\times[n]})\right] \leqslant & \left|\frac{1}{|s|\,n}\sum_{i,j\in s\times[n]}\ell(z_i,z_j;\mathcal{A}_{s\times[n]})\right|_{(\mathcal{L}_1)} \\ + & \left|\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}\left[\ell(z,z';\mathcal{A}_{s\times[n]})\right] - \frac{1}{n^2}\sum_{i,j\in[n]\times[n]}\ell(z_i,z_j;\mathcal{A}_{s\times[n]})\right|_{(\mathcal{L}_2)} \\ + & \left|\frac{1}{n^2}\sum_{i,j\in[n]\times[n]}\ell(z_i,z_j;\mathcal{A}_{s\times[n]}) - \frac{1}{|s|\,n}\sum_{i,j\in s\times[n]}\ell(z_i,z_j;\mathcal{A}_{s\times[n]})\right|_{(\mathcal{L}_3)}\end{aligned} \tag{3.2.8}$$

where the bound is controlled by $(\mathcal{L}_1)$ training loss $(i.e.,e(\mathcal{A}_{s\times[n]}))$, $(\mathcal{L}_2)$ the deviation between expected loss and empirical loss over all possible pairs, and $(\mathcal{L}_3)$ the deviation between empirical loss over all possible pairs and empirical loss over the subset of pairs defining the algorithm, $\mathcal{A}_{s\times[n]}$. It is widely observed that high capacity CNNs can reach very small training error. Moreover, $\mathcal{L}_2$ is proved to be bounded in [68] and is independent of $\mathcal{A}$. Thus, $\mathcal{L}_3$ characterizes the generalization performance of using the subset of pairs over exploiting all possible pairs.

Proposition 3.2.2 *Given* $\{z_i\}_{i\in[n]} \overset{i.i.d.}{\sim} p_{\mathcal{Z}}$ *and a set* $s \subset [n]$*. If* $s = \cup_k s'_k$ *with* s'_k *is the* δ_s*-cover of* $\{i \in [n] \mid y_i = k\}$ *(*i.e.,*the samples in class* k *),* $\ell(z, z'; \theta)$ *is* ζ*-Lipschitz in* x, x' *for all* y, y' *and* θ*, and bounded by* L*,* $e(\mathcal{A}_{s\times[n]})$ *training error; then with probability at least* $1 - \gamma$ *we have:*

$$\left|\frac{1}{n^2}\sum_{i,j\in[n]\times[n]}\ell(z_i, z_j; \mathcal{A}_{s\times[n]}) - \frac{1}{|s|\,n}\sum_{i,j\in s\times[n]}\ell(z_i, z_j; \mathcal{A}_{s\times[n]})\right|$$
$$\leqslant \mathcal{O}(\zeta\,\delta_s) + \mathcal{O}(e(\mathcal{A}_{s\times[n]})) + \mathcal{O}(L\sqrt{\tfrac{\log 1/\gamma}{n}})$$

Corollary 3.2.2.1 *Generalization of the proxy-based methods can be limited by the maximum of distances between the proxies and the corresponding class samples in the dataset.*

Proposition 3.2.2 implies that increasing the number of chance constraints with more anchor samples in the feasible point problem formulation of DML improves the generalization error bound as long as the included samples improve the covering radius of the dataset. In other words, including more anchor samples do not improve the bound unless the covering radius is decreased. Similarly, Corollary 3.2.2.1 informally suggests possible improvement on the generalization error bound of the proxy-based methods if we manage to introduce more proxies which are spread over the dataset once trained. In practice introducing more proxies generally does not help the performance; since, they eventually coalesce into a single point [25]. Besides, the computation resource limits the number of proxies to be included in the formulation. In the next section, we develop an approach to alleviate these problems.

3.2.3 Solving the Feasibility Problem

We now introduce our chance constrained programming (CCP) method, outlined in Algorithm 1, exploiting proxy-based training together with satisfying arbitrarily increased chance constraints. In short, we repeatedly solve a proxy-DML and improve the solution by re-initializing the proxies with the new samples reducing the covering radius.

Algorithm 1 CCP DML

```
initialize $\theta^*$ randomly, given $\{z_i\}_{i\in[n]}\sim p_{\mathcal{Z}}$ dataset
initialize $\rho^*$ with random samples, set budget $b$
repeat
    $\rho \leftarrow GreedyKCenterProxy(\rho^*, b, f(\cdot;\theta^*))$
    repeat
        sample $\{j(i) \in [n]\}_{i\in[m]}\sim[n]$ a batch
        $g_\theta\leftarrow\lambda\,(\theta^*-\theta)+\nabla_\theta \frac{1}{m|\rho|}\sum_{\rho\times[m]} \ell(\rho_i, z_j;\theta)$
        $g_\rho\leftarrow\nabla_\rho \frac{1}{m|\rho|}\sum_{\rho\times[m]} \ell(\rho_i, z_j;\theta)$
        $(\theta, \rho)\leftarrow ApplyGradient(\theta, \rho, g_\theta, g_\rho)$
    until convergence
    $\theta^*\leftarrow\theta$, $\rho^*\leftarrow\rho$
until convergence
```

We consider the problem in Eq. (3.2.7) as finding a point in the intersection of the sets. In particular, given dataset $\{z_i\}_{i\in[n]} \sim p_{\mathcal{Z}}$, we have m many 1-per-class sets, $s_k = \{s_k(i) \in [n] \mid y_{s_k(i)} = i\}_{i\in[C]}$, to define the constraint set as $\mathcal{C}_s = \cap_{k\in[m]}\mathcal{C}_{s_k}$. If the sets were closed and convex, the problem would be solvable by iterative projection methods [99, 100]. Nevertheless, it is not uncommon to perform iterative projection methods to non-convex set intersection problems [101, 102]. Hence, we propose to solve the problem approximately by performing iterative projections onto the feasible sets, $\mathcal{C}_{s_k}$, defined by s_k. At each iteration, k, we solve the following projection problem given $\theta^{(k\text{-}1)}$:

$$\theta^{(k)} = \arg\min_{\theta\in\mathcal{C}_{s_k}} \tfrac{1}{2}\|\theta^{(k\text{-}1)} - \theta\|_2^2 \tag{3.2.9}$$

where C_{s_k} is defined in Eq. (3.2.7). Using expectation bounds as the surrogate of the penalty terms for the chance constraints as we do in § 3.2.1, we have:

$$\theta^{(k)} = \arg\min_{\theta} \tfrac{\lambda}{2}\|\theta^{(k\text{-}1)} - \theta\|_2^2 + \tfrac{1}{c}\sum_{i\in[c]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}\left[\ell(z_{s_k(i)}, z;\theta)\right] \tag{3.2.10}$$

where λ is a hyperparameter for the projection regularization. We can minimize the resultant loss by using batch stochastic gradient approaches. However, the batch should be augmented by c many anchor samples to compute the loss, which becomes prohibitive for large-scale problems. To alleviate costly embedding computation of c many samples, we propose to use proxies, ρ_i, in place of the embedding of the samples, $z_{s_k(i)}$. Namely, at each iteration k, we initialize $\rho_i = f(z_{s_k(i)}; \theta^{(k\text{-}1)})$ and solve:

$$\theta^{(k)}, \rho^{*} = \arg\min_{\theta,\rho} \tfrac{\lambda}{2} \|\theta^{(k\text{-}1)} - \theta\|_2^2 + \tfrac{1}{c} \sum_{i\in[c]} \mathbb{E}_{z\sim p_{\mathcal{Z}}} \left[\ell(\rho_i, z; \theta)\right] \tag{3.2.11}$$

where the resultant problem we solve at each iteration corresponds to a proxy-based DML. Any pairwise distance based loss can replace $\ell(\cdot)$ with anchor samples being class proxies. Namely, we repurpose existing objectives with a regularization term in an iterative manner. Although we set up the formulation using single proxy per class, multiple proxy extension is straightforward.

Theoretically, we should cycle through the sets until convergence to solve $\theta \in \cap_{k\in[m]} \mathcal{C}_{s_k}$. Thus, we must pick anchor samples for each set to initialize proxies. The updates of the proxies are not guaranteed to mimic the actual updates of the corresponding anchor samples. With that being said, we will still have a solution, as Eq. (3.2.6) suggests, to feasibility of some chance constraints as long as the converged proxies, ρ^*, are diverse. We empirically observe that the proxies initialized with diverse samples converge to embedding of distinct samples (Fig. 3.2). Hence, on one hand, we have solutions to different constraint sets as long as we re-initialize the proxies with new samples and solve proxy-based DML problems. On the other hand, Proposition 3.2.2 implies that generalization is improved as long as we end up with converged proxies reducing the covering radius. Therefore, the theory suggests a set intersection mechanism to reduce the covering radius yet allows a greedy algorithm via iterative projections to select (*i.e.*,initialize) the next proxies on the fly instead of explicitly defining the sets we will iterate on. Such a result is useful especially for the cases where the dataset is stochastically extended with random data augmentations which obstruct explicit set forming.

Proxy selection. We can simply use random sampling for anchor samples to

Algorithm 2 Greedy K-Center Proxy

```
input: proxy set $\rho$, sampling budget $b$ and $f(\cdot;\theta)$
repeat for each class $c$
    $s_c \leftarrow \{x_i \mid y_i = c\}_{i\in[b]}$, $b$-sample-per-class
    initialize $r_c \leftarrow \{\}$, $p \leftarrow f(s_c;\theta)$
    repeat
        $q \leftarrow \arg\max_{u\in p\setminus r_c} \min_{v\in \rho_c \cup r_c} \|u - v\|_2$
        $r_c \leftarrow \{q\} \cup r_c$
    until $|r_c| = |\rho_c|$
return $\cup_c r_c$
```

initialize proxies since we eventually observe informative samples reducing the covering radius through the iterations. We can as well explicitly mine samples that possibly help with reducing the covering radius. Thus, we also exploit clever selection of proxies as outlined in Algorithm 2. Given a budget, b, we sample b many instances per class and compute their embeddings to form a pool. We then select the samples that reduce the covering radius most once added to proxy set. This selection is equivalent to k-Center problem as formulated in [69]. Such a selection of proxies helps converged proxies to be diverse. $b = 1$ reduces to random sampling. In both, we inherently increase the number of anchor samples defining the problem and hence reducing the covering radius.

Relation to cross-batch-memory (XBM) [14]. XBM stores past embeddings in a queue based memory which dequeues the oldest ones at each iteration to enqueue the latest batch. If the memory is much larger than the batch size and *slow drift* [14] is assumed, the loss terms are conditioned to particular proxies until they are updated. To this end, XBM can be seen as solving alternating problems of proxy-based DML with *fuzzy* boundaries and $\lambda = 0$.

Relation to exponential moving average (EMA) of weights. θ' denoting $\theta^{(k\text{-}1)}$, we can express the stochastic gradient descent update corresponding to Eq. (3.2.11) as $\theta \leftarrow (1 - \lambda)(\theta - \nabla_\theta) + \lambda\theta' + \varepsilon$ where the error term ε can be made 0 with a proper learning rate. If we perform only 1 step per projection problem, then the form of this update imposes the EMA weight regularization, akin to weight decay. Sharing similar spirit with our method, such EMA update of the weights is employed in aggregating the ensemble of neural networks-produced during the training- to a target neural network [103, 104]. In that setting, an online network gets the θ - ∇_θ update, and a separate target network updates its parameters θ' using EMA of online network parameters. We encounter such a mechanism in self-supervised training [103] and self-teacher-student networks [104].

3.2.4 Implementation Details

Embedding function. For the embedding function, $f(\cdot;\theta)$, we use CNNs with ReLU activation, max- and average-pooling. In particular, we use ResNetV2-20 [105] for MNIST [106] experiments, and ImageNet [107] pretrained BN-Inception [108] for the rest. We exploit architectures until the output of the global average pooling layer. We add a fully connected layer to the output of the global average pooling layer to obtain the embedding vectors of size 2 (ResNetV2-20) and 128 (BN-Inception). We state the following lemma to prove our loss is Lipschitz continuous:

Lemma 3.2.1 *Generalized contrastive loss defined as*

$$\ell(z, z'; \theta) := (\iota_{y,y'}(\|x - x'\|_{f_\theta} - \beta) + \alpha)_+$$

is $\sqrt{2}\omega^L$-Lipschitz in x and x' for all y, y', θ for the embedding function $f(\cdot;\theta)$ being L-layer CNN (with ReLU, max-pool, average-pool) with a fully connected layer at the end, where ω is the maximum sum of the input weights per neuron.

ω can be made arbitrarily small by using weight regularization, which is commonly used. SOTA methods widely use $l2$ normalization on the embeddings. For normalization, we apply $\hat{v} = {}^{v}/_{\|v\|_2}$ if $\|v\|_2 \geqslant 1$ or no normalization otherwise ($i.e.$,$\hat{v} = v$ if $\|v\|_2 \leqslant 1$). Unlike $l2$ normalization, such a transform is Lipschitz continuous, hence so are our loss.

Solving projections. Performing a projection defined in (3.2.11) involves a minimization problem. We monitor MAP@R validation accuracy and use early stopping patience of 3 to pass the next projection.

3.3 Experimental Work

We start our empirical study with *proof of the concept* tests validating the role of proxy-based approaches in learning and the impact of alternating proxies on the feature geometry. We further perform ablation studies for the implications of our formulation as well as the effects of the hyperparameters. We examine the effectiveness of the proposed proxy-based DML framework for the image retrieval task. We use our own framework implemented in Tensorflow [109] library in the experiments. Throughout the section, we use CCP to refer our framework.

3.3.1 Proof of the Concept

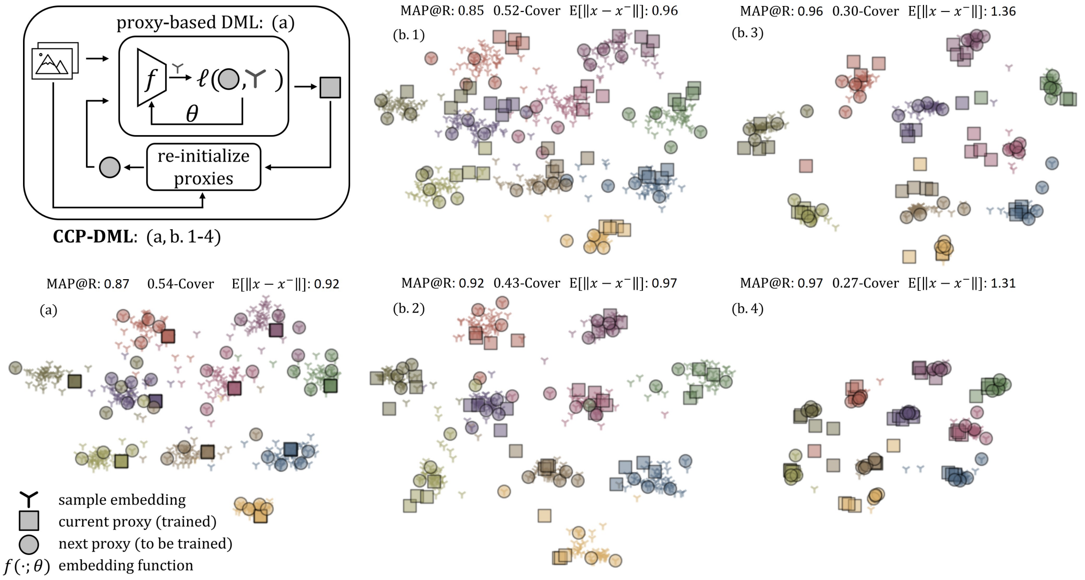


Figure 3.2: Illustration of our method (CCP) and the geometry of the embedding space before, (a), and after, (b), our method (through iterations 1-4), where boxes are the converged proxies and the circles are the next proxies as the result of k-Center. In proxy-based DML, proxies are coalesced into one whereas with CCP, we have diverse proxies, resulting reduced covering radius.

We evaluate our method on MNIST dataset with 2-D embeddings to show the implications of our formulation. In Fig. 3.2, we provide the distribution of the samples in the embedding space. We use 4 proxies per class and pool size b=16.

We observe that when single proxy-based method is converged (Fig. 3.2-(a)), the class proxies collapse to a single point. Once we continue training with proposed approach (Fig. 3.2-(b)), the covering radius decreases, leading to performance improvement. It can also be observed that diverse samples results in diverse proxies. We also experiment the case where we use samples instead of proxies. Though it is not practically applicable to large-scale problems, it is important to see whether our intuitions about alternating proxies in place of samples hold. We obtain 98.06% MAP@R performance with sample-based training against 97.21% MAP@R performance of proxy-based training. This empirical result supports our motivation on using the proxies in place of samples. With that being said, it is important to show how such efforts in the training domain are reflected in the test domain.

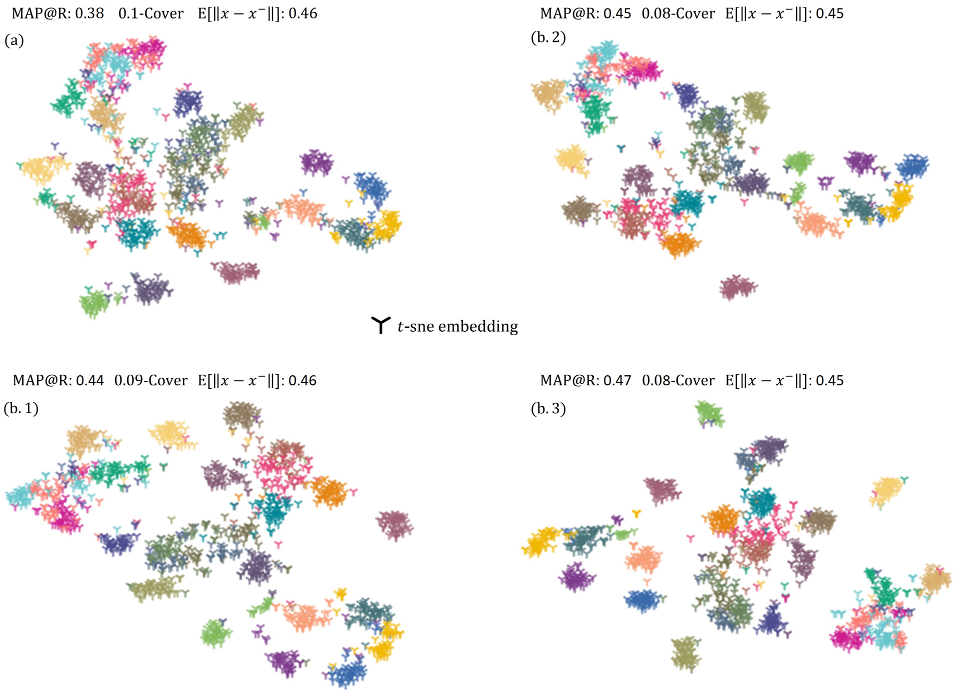


Figure 3.3: The geometry of the embedding space before, (a), and after, (b), our method (through iterations 1-3), relating how the generalization efforts in training domain transfer to the geometry of test domain on CUB dataset with Contrastive+CCP. We use 2-D TSNE embeddings of the validation data in the visualization, in which we report MAP@R, average covering radius and average inter-class pairwise distances.

We further provide the visualization of the validation data in CUB dataset in Fig. 3.3. We compute covering radii for 1 to n sample case in k-Center. Namely, we take k samples with minimum cover for $k \in [n]$ where n is the number of samples per class. We then take the average of these radii to compute a representative metric for the covering radius. We observe that solving single proxy-based DML results in relatively poor generalization in the test domain. On the contrary, solving the problem as the set intersection problem with alternating projections improves the embedding geometry (reduced radius with increased inter-class pairwise distances).

3.3.2 Deep Metric Learning Experiments

3.3.2.1 Setup

Independent works [6, 7, 110] reveal that conventional training and evaluation procedures in DML may fail to properly assess the true order of performance that the methods bring. The consensus for unbiased comparability is evaluation of the methods with their best version under the same experimental settings unless the compared methods demand any particular architecture or experimental setup. Our empirical study is completely aligned with the literature's claims for unbiased evaluation of our method.

We mostly follow the procedures proposed in [6] to provide fair and unbiased evaluation of our method as well as comparisons with the other methods. We provide full detail of our experimental setup for the sake of complete transparency and reproducibility.

Datasets

We perform our experiments on 4 widely-used benchmark datasets: Stanford Online Products (SOP) [32], In-shop [111], Cars196 [112] and, CUB-200-2011 (CUB) [113].

SOP has 22,634 classes with 120,053 product images. The first 11,318 classes

(59,551 images) are split for training and the other 11,316 (60,502 images) classes are used for testing.

In-shop has 7,986 classes with 72,712 images. We use 3,997 classes with 25,882 images as the training set. For the evaluation, we use 14,218 images of 3,985 classes as the query and 12,612 images of 3,985 classes as the gallery set.

Cars196 contains 196 classes with 16,185 images. The first 98 classes (8,054 images) are used for training and remaining 98 classes (8,131 images) are reserved for testing.

CUB-200-2011 dataset consists of 200 classes with 11,788 images. The first 100 classes (5,864 images) are split for training, the rest of 100 classes (5,924 images) are used for testing.

Data augmentation follows [6]. During training, we resize each image so that its shorter side has length 256, then make a random crop between 40 and 256, and aspect ratio between $3/4$ and $4/3$. We resize the resultant image to 227x227 and apply random horizontal flip with 50% probability. During evaluation, images are resized to 256 and then center cropped to 227x227.

Training Splits

Fair evaluation. We split datasets into disjoint training, validation and test sets according to [6]. In particular, we partition $50\%/50\%$ for training and test, and further split training data to 4 partitions where 4 models are to be trained by exploiting $1/4$ as validation while training on $3/4$.

Conventional evaluation. Following relatively *old-fashioned* conventional evaluation [32], we use the whole train split of the dataset for training and we use the test split for evaluation as well as monitoring the training for early stopping.

Ablation studies. For the additional experiments related to the effect of hyperparameters, we split training set into 3 splits and train a single model on the $2/3$ of the set while using $1/3$ for the validation.

Evaluation Metrics

We consider precision at 1 (P@1) and mean average precision (MAP@R) at R where R is defined for each query and is the total number of true references as the query. Among those, MAP@R performance metric is shown to better reflect the geometry of the embedding space and to be less noisy as the evaluation metric [6]. Thus, we use MAP@R to monitor training in our experiments except for conventional evaluation setting where we monitor P@1. We explain the metrics in § 1.2.

Training Procedure

Fair evaluation. We use Adam [114] optimizer with constant 10^{-5} learning rate, 10^{-4} weight decay, and default moment parameters, β_1=.9 and β_2=.99. We use batch size of 32 (4 samples per class). We evaluate validation MAP@R for every 25 steps of training in CUB and Cars196, for 250 steps in SOP and In-shop. We stop training if no improvement is observed for 60 steps. We recover the parameters with the best validation performance. Following [6], we train 4 models for each $^3/_4$ partition of the train set.

Conventional evaluation. We use Adam [114] optimizer with default moment parameters, β_1=.9 and β_2=.99. Following recent works [37], we use *reduce on plateau* learning rate scheduler with patience 4. The initial learning rate is 10^{-5} for CUB, and 10^{-4} for Cars, SOP and In-shop. We use 10^{-4} weight decay for BNInception backbone and $4\,10^{-4}$ wight decay for ResNet50 backbone. We use batch size of 128 (4 samples per class) for BNInception backbone and 112 (4 samples per class) for ResNet backbone (following [7]). We evaluate validation P@1 for every 25 steps of training in CUB and Cars196, for 250 steps in SOP and In-shop. We stop training if no improvement is observed for 15 steps in CUB and Cars196, and 10 steps in SOP and In-shop. We recover the parameters with the best validation performance.

Ablation studies. We use Adam [114] optimizer with constant 10^{-5} learning rate, 10^{-4} weight decay, and default moment parameters, β_1=.9 and β_2=.99. We use batch size of 32 (4 samples per class). We evaluate validation MAP@R for every 25 steps of training in CUB and Cars196, for 250 steps in SOP and In-shop. We stop training if no improvement is observed for 40 steps. We recover the parameters with the best validation performance. We train a single model on the $^2/_3$ of the training set while using $^1/_3$ for the validation.

Embedding vectors

Fair evaluation. Embedding dimension is fixed to 128. During training and evaluation, the embedding vectors are $l2$ normalized. We follow the evaluation method proposed in [6] and produce two results: *i*) Average performance (128 dimensional) of 4-fold models and *ii*) Ensemble performance (concatenated 512 dimensional) of 4-fold models where the embedding vector is obtained by concatenated 128D vectors of the individual models before retrieval.

Conventional evaluation. Embedding dimension is 512 in both BNInception and ResNet50 experiments.

Ablation studies. Embedding dimension is fixed to 128.

Losses with CCP

We evaluate our method with *C1+CCP*: Contrastive [29], *C2+CCP*: Contrastive with positive margin [9], *MS+CCP*: Multi-similarity (MS) [33], *Triplet+CCP*: Triplet [30].

Regarding the other popular losses, ProxyAnchor [37] is indeed proxy-based MS loss except for missing a margin term. Similarly, ProxyNCA [15] is $\log\Sigma\exp$-approximation of proxy-based Triplet with hard-mining and for single proxy case SoftTriple [25] is equivalent to ProxyNCA. Therefore, we should note that our experiments cover wide range of the DML losses.

Hyperparameters

For the hyperparameter selection, we exploit the recent work [6] that has performed parameter search via Bayesian optimization on variety of losses. We further experiment the suggested parameters from the original papers and official implementations. We pick the best performing parameters. We perform no further parameter tuning for the loss parameters when applied to our method to purely examine the effectiveness of our method.

C1: We adopted XBM's official implementation for fair comparison. We use 0.5 margin for all datasets.

C2: C2 has two parameters, (m^+, m^-): positive margin, m^+, and negative margin. We set (m^+, m^-) to $(0, 0.3841), (0.2652, 0.5409), (0.2858, 0.5130), (0.2858, 0.5130)$ for CUB, Cars196, In-shop and SOP, respectively.

Triplet: We set its margin to 0.0961, 0.1190, 0.0451, 0.0451 for CUB, Cars196, In-shop and SOP, respectively.

MS: MS has three parameters (α, β, λ). We set (α, β, λ) to $(2, 40, 0.5)$, $(14.35, 75.83, 0.66)$, $(8.49, 57.38, 0.41)$, $(2, 40, 0.5)$ for CUB, Cars196, In-shop and SOP, respectively.

ProxyAnchor: We set its two paremeters (δ, α) to $(0.1, 32)$ for all datasets. We use 1 sample per class in batch setting (*i.e.*,32 classes with 1 samples per batch), we perform 1 epoch warm-up training of the embedding layer, and we apply learning rate multiplier of 100 for the proxies during training.

ProxyNCA++: We set its temperature parameter to 0.1 for all datasets. We use 1 sample per class in batch setting (*i.e.*,32 classes with 1 samples per batch), we perform 1 epoch warm-up training of the embedding layer, and we apply learning rate multiplier of 100 for the proxies during training.

SoftTriple: SoftTriple has 4 parameters $(\lambda, \gamma, \tau, \delta)$. We set $(\lambda, \gamma, \tau, \delta)$ to $(20, 0.1, 0.2, 0.01)$, $(17.69, 19.18, 0.0669, 0.3588)$, $(20, 0.1, 0.2, 0.01)$, $(100, 47.9, 0.2, 0.3145)$ for CUB, Cars196, In-shop and SOP, respectively. We use

1 sample per class in batch setting (*i.e.*,32 classes with 1 samples per batch), we perform 1 epoch warm-up training of the embedding layer, and we apply learning rate multiplier of 100 for the proxies during training.

XBM: We evaluate XBM with C1 and C2; since, in the original paper, contrastive loss is reported to be the best performing baseline with XBM. We set the memory size of XBM to the total number of proxies (*i.e.*,$proxy\_per\_class \times \#classes$) to compare the methodology by disentangling the effect of proxy number. With that being said, we also evaluate XBM with the memory sizes suggested in the original paper. In this manner we use two memory sizes for XBM for each dataset: (S, L) where S and L denote the number of batches in the memory. For CUB and Cars196, CCP uses 1(8) proxies per class for $S(L)$. Thus, we set (S, L) to $(3, 25)$ for CUB and Cars196. For In-shop and SOP, CCP uses 1(4) proxies per class for $S(L)$. Thus, we set (S, L) to $(100, 400), (400, 1400)$ for In-shop and SOP, respectively. We perform 1K steps of training with the baseline loss prior to integrate XBM loss in order to ensure *slow drift* [14] assumption.

CCP: For the hyperparameters of our method, we use 8 proxies per class and λ=2 10^{-4} for CUB and Cars datasets, as the result of the parameter search; and use pool size, b=12, for greedy k-Center method. We select pool size based on our empirical studies on the effect pool size and number of proxies. Due to computation limitations, we use 4 proxy per class, λ=2 10^{-4} and b= 7 for SOP and In-shop dataset. We perform no warm-up or do not use learning rate multiplier for the proxies.

Compared Methods and Fairness

Compared methods. We compare our method against proxy-based SoftTriple [25], ProxyAnchor [37] and ProxyNCA++ [15] methods as well as XBM [14].

Fairness. We note that like the compared methods (*i.e.*,loss functions, proxy-based methods), our method's improvement claims do not demand any particular architecture or experimental setup. Therefore, to evaluate the improvements purely coming from the proposed ideas, we implemented the best version of the compared methods in our framework and evaluate on the same architecture and experimental settings. In this manner, we stick to BN-Inception with global average pooling architecture to directly compare our method with the benchmarked losses in [6]. To eliminate any framework related performance differences, we re-implemented the methods within our framework and produce the consistent results with [6].

Our experimental setting is fair and unbiased; since,

- The compared methods are either invented loss functions or proxy-based approaches, which do not demand a particular setting to show the effectiveness of the proposed ideas.
- We use the same experimental setting for each method (*e.g.* image size, architecture, embedding size, batch size, data augmentation).
- We implement and re-evaluate all the compared methods on our framework.
- We reproduce consistent results reported in [6] to eliminate any framework related performance bias.
- We use the same train and test split as the conventional methods, but we do not exploit test data during training.

3.3.2.2 Results

Fair Evaluation

We provide quantitative results in Tab. 3.1 for the evaluation of the methods under the same experimental settings (*i.e.*,fair evaluation). We summarize the results in Fig. 3.4 through average 128D and concatenated 512D MAP@R performance of 4-fold models on In-shop and SOP. We use Method-S/L naming convention to denote memory size in XBM, and the proxy per class in SoftTriple and CCP where S denotes 1, and L denotes 4(10) for SoftTriple and 4(8) for CCP in In-shop, SOP (CUB, Cars196). For a fair comparison, we match XBM memory size and the number of proxies in CCP.

Table 3.1: Comparison with the existing methods for the retrieval task on SOP, InShop, CUB, Cars

	SOP				**In-shop**				**CUB**				**Cars196**			
	512D		**128D**		**512D**		**128D**		**512D**		**128D**		**512D**		**128D**	
Method	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**
C1	68.84	40.25	64.96	36.54	80.12	50.15	76.08	46.51	63.67	23.08	56.21	19.06	77.75	23.50	64.17	16.13
C1+XBM-L	78.68	51.82	75.37	47.39	88.39	58.64	**85.75**	55.37	65.40	24.87	57.57	19.84	83.68	27.93	**72.13**	18.83
C1+CCP-L	**79.53**	**52.73**	**76.24**	**48.07**	**88.52**	**59.67**	85.50	**55.56**	**68.11**	**27.11**	**59.56**	**21.27**	**83.76**	**28.32**	72.05	**18.96**
C2	74.87	46.94	71.15	42.66	86.32	59.42	83.04	55.13	67.49	26.47	59.73	21.01	81.04	24.73	69.17	17.22
C2+XBM-L	76.66	49.04	73.47	45.15	87.66	60.64	84.58	56.75	68.62	26.83	60.18	21.41	82.40	25.99	70.01	18.01
C2+CCP-L	**78.95**	**52.19**	**75.92**	**48.18**	**88.52**	**61.07**	**86.11**	**57.24**	**69.73**	**28.02**	**62.39**	**22.67**	**82.89**	**26.27**	**72.16**	**18.52**
MS	72.74	44.10	68.96	40.18	88.37	60.65	85.39	56.61	64.65	24.15	57.24	19.64	80.88	26.23	69.27	18.25
MS+CCP-L	**78.96**	**51.85**	**75.80**	**47.97**	**90.24**	**63.59**	**87.10**	**59.15**	**68.84**	**27.44**	**61.10**	**22.40**	**86.26**	**29.14**	**74.97**	**19.85**
Triplet	75.40	47.03	70.41	41.03	86.71	60.60	82.58	55.25	64.01	23.43	55.51	18.51	78.44	23.11	64.57	15.68
Triplet+CCP-L	**77.09**	**49.33**	**72.21**	**43.11**	**89.44**	**64.28**	**86.00**	**59.04**	**65.36**	**24.53**	**56.65**	**19.31**	**81.84**	**25.21**	**68.75**	**17.43**
ProxyAnchor	77.10	49.01	73.86	44.89	88.08	58.09	85.87	54.95	68.43	26.53	60.61	21.48	85.29	27.73	**75.79**	19.56
ProxyNCA++	76.07	48.20	72.89	44.44	87.33	57.48	84.79	54.42	65.48	24.85	58.49	20.96	82.87	26.34	72.45	19.32
SoftTriple-S/L	78.48	50.77	74.66	45.75	88.37	59.56	85.71	55.68	66.12	26.02	57.94	19.86	84.90	27.80	73.16	19.18

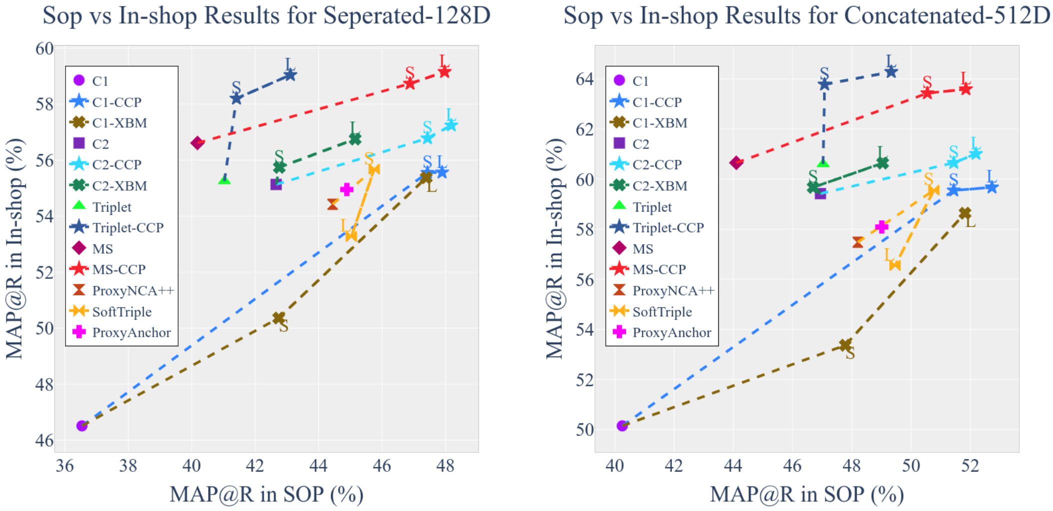


Figure 3.4: Summary of relative improvements

We observe that CCP consistently outperforms the associated baseline methods on each dataset. Contrastive loss' great performance with CCP is important to support the implications of our formulation. Furthermore, performance improvements on the loss functions which does not directly fit in our formulation show the broader applicability of our method to the pairwise distance based loss functions.

Additionally, the proposed CCP framework outperforms not only the related proxy-based methods but also every single benchmarked approaches in [6]. When compared with SoftTriple and XBM (*i.e.*,multiple proxy methods), our method outperforms by large margin especially in the cases where less number of proxies are used (*i.e.*,*method*-S comparisons in Fig. 3.4). We observe especially in large-scale datasets (SOP & In-shop) that even single proxy per class brings substantial performance improvement with our CCP.

Conventional Evaluation

We additionally follow the relatively old-fashioned conventional procedure [32] for the evaluation of our method. We provide R@1 results in Tabs. 3.2a and 3.2b for the comparison with SOTA.

Table 3.2: Comparison with the existing methods for the retrieval task in conventional experimental settings with BN-Inception and ResNet50 backbones where superscripts denote embedding size. Red: the best. Blue: the second best. Bold: previous SOTA. †*Results obtained from* [1].

(a)

Backbone →	**BN-Inception-512D**			
Dataset →	**CUB**	**Cars196**	**SOP**	**In-shop**
Method ↓	**R@1**	**R@1**	**R@1**	**R@1**
SoftTriple-L [25]	65.40	84.50	78.60	-
C1+XBM-L [14]	65.80	82.00	**79.50**	89.90
ProxyAnchor [37]	68.40	**86.10**	79.10	**91.50**
DiVA [23]	66.80	84.10	78.10	-
ProxyFewer [59]	66.60	85.50	78.00	-
Margin-S2SD [24]	**68.50**	**87.30**	79.30	-
C1+CCP-L	67.74	83.74	**79.86**	90.98
C2+CCP-L	**69.87**	83.90	**80.01**	**91.72**
MS+CCP-L	**69.09**	86.01	79.75	**91.84**

(b)

Backbone →	**ResNet50**			
Dataset →	**CUB**	**Cars196**	**SOP**	**In-shop**
Method ↓	**R@1**	**R@1**	**R@1**	**R@1**
C1+XBM^{128} [14]	-	-	80.60	91.30
ProxyAnchor^{512} [37]	69.70	87.70	80.00†	92.10†
DiVA^{512} [23]	69.20	87.60	79.60	**-**
ProxyNCA++^{512} [15]	66.30	85.40	80.20	88.60
Margin-S2SD^{512} [24]	69.00	**89.50**	81.20	-
LIBC^{512} [1]	**70.30**	88.10	**81.40**	**92.80**
ProxyAnchor-DIML^{128} [65]	66.46	86.13	79.22	-
C1+CCP-L^{512}	69.87	87.12	**82.77**	92.07
C2+CCP-L^{512}	**71.04**	85.93	**82.34**	92.46
MS+CCP-L^{512}	**70.37**	**89.02**	82.27	**92.71**

We observe that our method outperforms SOTA in most cases and performs on par with or slightly worse in a few. We should recapitulate that R@1 is a myopic metric to assess the quality of the embedding space geometry and hence, pushing R@1 does not necessarily reflect the true order of the improvements that the methods bring. As we observe from Tab. 3.1 that the methods sharing similar R@1 (*i.e.*,P@1) performances can differ in MAP@R performance relatively more significantly. In that manner, we firmly believe that comparing MAP@R performances instead of R@1 technically sounds more in showing the improvements of our method.

3.3.2.3 Ablations

Effect of Proxy per Class and Projection Regularization

We perform Bayesian search on the λ-$\#proxy$ space to see the effect of two in CUB with C2-CCP. We provide the results in Fig. 3.5. We observe that absence of

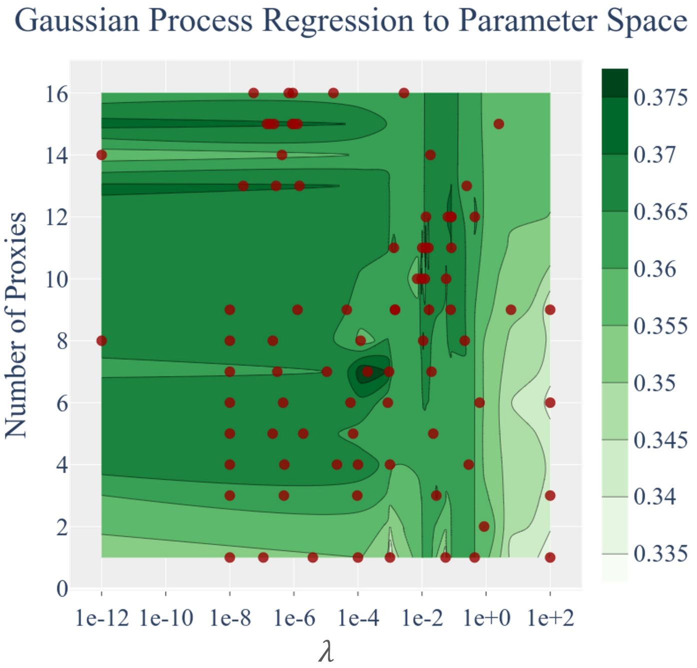


Figure 3.5: Bayesian search on λ-#*proxy* space

λ degrades the performance. Similarly, large values of λ causes over-regularization. We obtain interval of $[10^{-1}, 10^{-5}]$ that works well for λ. For the number of proxies, we observe increasing the proxy per class improves performance. On the other hand, the increase saturates as it can also be observed from Fig. 3.6. As the result of Bayesian parameter search, we take λ=2 10^{-4} and #*proxy*=8 with pool size *b*=12 in our evaluations against other methods for CUB and Cars. For SOP and In-shop, we reduce #*proxy*=4 and *b*=7 owing to relatively less number of samples per class in the dataset.

Effect of Proxy Selection

We perform ablation study with C2-CCP to see the relation between the number of proxies and the pool size used for the proxy selection. We give the corresponding results in Fig. 3.6. We observe that both increasing the number of proxies and the pool size for proxy selection helps performance. We interestingly see that for single proxy case, increasing the pool size gives no better results than random selection. Owing to our greedy proxy selection, we do consider the past

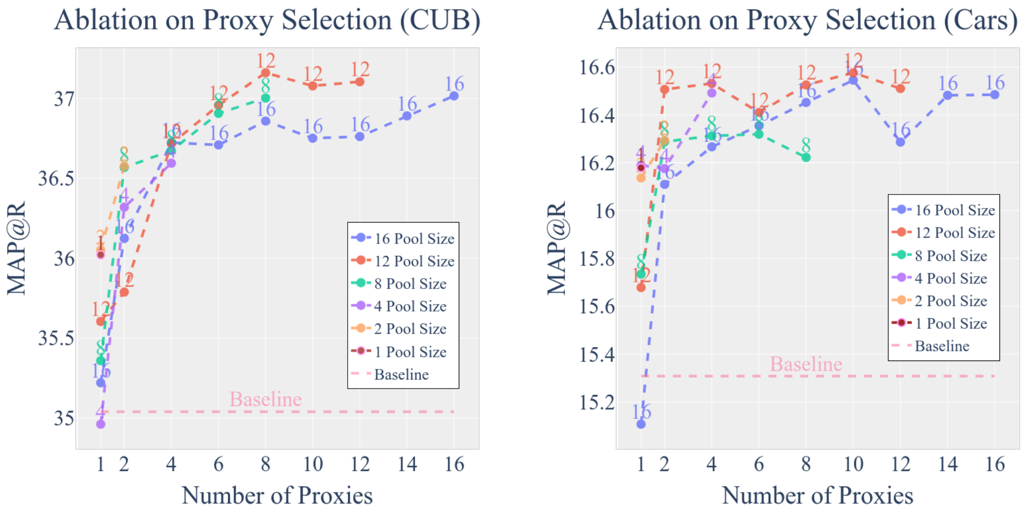
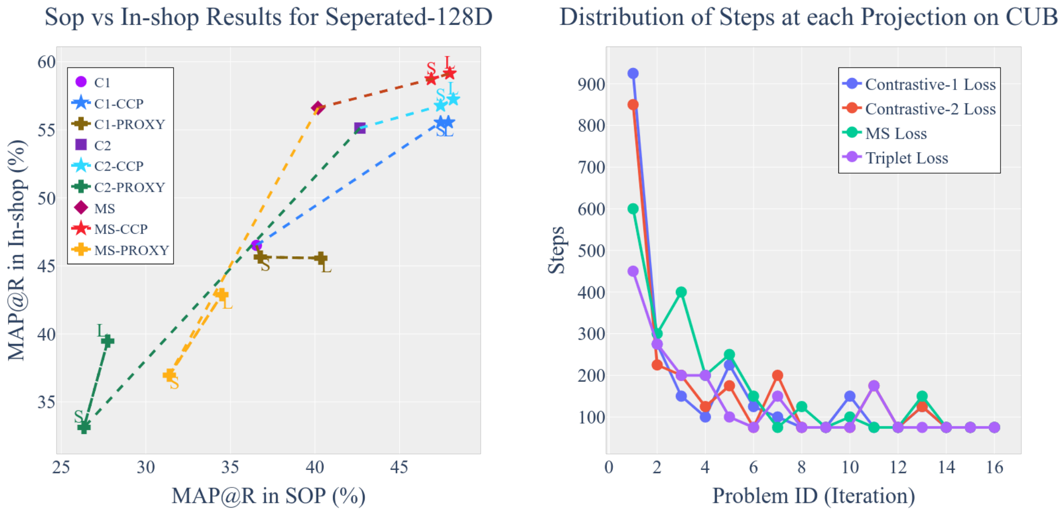


Figure 3.6: Analysis of the relation between the number of proxies and the pool size used for the proxy selection on CUB (left) and Cars (right) dataset with C2-CCP.

geometry no earlier than single step. Thus, in the single proxy case, we are prune to oscillate between similar samples for proxy selection. On the other hand, selecting the samples that reduce the covering radius most brings better generalization over random selection. With that being said, random sampling in proxy selection ($i.e.$,pool size = #proxy) still works well; since, random sampling indeed can provide diversity in the samples as well. Such a result supports that the key to our method is alternating the proxies with new samples. As long as we re-initialize the proxies with new samples, we will have some diverse proxies through the iterations. To this end, we use Greedy K-Center to pick the samples in a clever way to reduce the covering radius as much as we can (*analogous to mining in batch construction*).

Effect of Alternating Problems

We provide results on MNIST in Fig. 3.2 to show the effect of solving alternating problems instead of single proxy-based DML. We additionally evaluate the baseline losses through solving only a single proxy-DML (Loss-Proxy) to show (Fig. 3.7-(a)) that our performance increase is not solely coming from augmentation of proxies in the problem. We clearly observe that alternating proxies helps performance as our formulation suggests. Moreover, we also provide a typical distribution of the steps per proxy-based projection problem in Fig. 3.7-(b) to show that we are not greedy on alternating the proxies just to provide more examples. We do have some steps that are relatively low, implying the selected proxies are not informative enough to change the embedding space geometry.



(a) (b)

Figure 3.7: Impact of alternating proxies (a) and typical distribution of the steps per projection problem (b).

Effect of Batch Size

Batch size plays important role in DML methods to perform well. Therefore, we analyze the robustness to the batch size especially for the cases where increasing the batch size is prohibitive. We train baseline contrastive loss and CCP contrastive loss for the batch sizes of 16, 32, 64 and 128. The training setup is the same as in the *fair evaluation* (§ 3.3.2.1). In each batch we use 4 samples per class. We provide the results in Fig. 3.8. We observe that baseline contrastive loss has increasing performance as the batch size increases whereas our method's performance with small batch size is on par with the large batch size. Thus, our method has reduced batch size complexity.

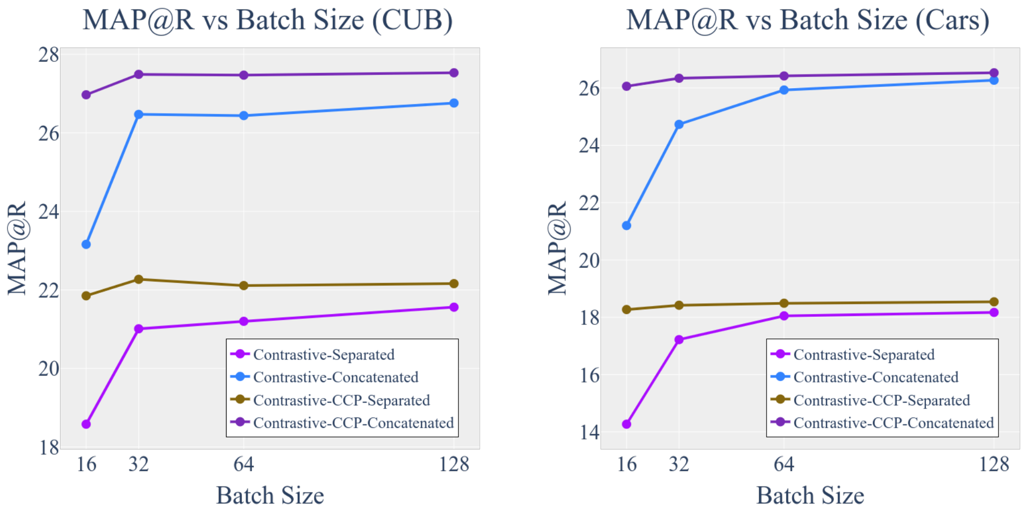


Figure 3.8: Analysis of batch size dependence of the performance on CUB (left) and Cars (right) dataset with C2-CCP.

Computational analysis

Proposed method outlined in Algorithm 1 puts little computation and memory overhead on top of the traditional approaches.

Table 3.3: Total steps of training in SOP and In-shop

Method	**SOP**	**In-shop**
C2	62K	114K
C2+XBM	81K	93K
C2+CCP	69K	127K
MS	67K	98K
MS+CCP	91K	131K
Triplet	93K	73K
Triplet+CCP	124K	115K
ProxyAnchor	54K	87K
ProxyNCA++	88K	103K
SoftTriple	48K	82K

For the computation, we have proxy initialization and weight update steps at the beginning of the each problem instance. In overall, in our system with RTX 2080 Ti GPU and i7 CPU, that additional computation adds on the average 5-10 ms per step (batch update). In particular, for batch size of 32, we typically have rate of 105 ms/batch with Contrastive-CCP whereas vanilla has 97 ms/batch rate. In In-shop and SOP dataset, we have the same rates however for CCP, we have 200 to 400 ms computation overhead due to sampling for proxy initialization. We do not have such overhead in Cars and CUB owing to the much less number of classes. With that being said, we have such 400 ms overhead in In-shop and SOP only at the beginning of new problem instance, which has no significant effect in long run. Due to alternating problems, our method takes more steps to converge than their baseline counterparts. We provide the optimization steps per proxy-based problem instance for several losses in Fig. 4(b) (main paper) from which relative convergence can be compared owing to each problem instance being a proxy-based method. Nevertheless, we also provide Tab. 3.3 to compare the convergence of the methods for SOP and In-shop datasets. The reported numbers are the rounded averages of the 4 models. We observe 10%-35% increase in the optimization steps for the pairwise losses.

For the memory, we store the weighs of the previously converged model in the memory as well as the variables for proxies. For the model, approximately 40-45 mb additional GPU memory is used and for the proxies 16.6 MB and 5.9 MB memory is used in SOP and In-shop dataset (75 kb in CUB and Cars).

In summary, increasing the number of proxies results in $\approx 8\%$ increase in back-propagation computation time and only ≈ 60 MB increase in memory for the largest model. Proxy re-initialization happens rather infrequently; thus, has no significant effect in the long run. Our method takes 10% - 35% more steps to converge than their baseline counterparts due to alternating problems. With that being said, our method with small batch size performs on par with the large batch size owing to alternating proxies. In this manner, marginal increase in computation is seemingly a fair trade-off in improving the performance along with robustness to batch size.

CHAPTER 4

IMPLICATIONS OF GLOBAL AVERAGE POOLING

Recapitulating, distance metric learning (DML) is the problem of finding suitable parameters for an embedding function so that the semantically similar samples are embedded to the small vicinity in the representation space as the dissimilar ones are placed relatively apart in the Euclidean sense. The typical embedding function for visual tasks is implemented as a convolutional neural network (CNN) followed by a global pooling layer. In supervised setting, the function parameters are learned through minimizing the empirical expected pairwise loss with pairs sampled from a class-labelled training dataset. As we discuss in detail at Ch. 2, the efforts to improve the performance includes *i*) tailoring pairwise loss terms [6] that penalize the violations of the desired intra- and inter-class proximity constraints, *ii*) pair mining [7], *iv*) augmenting the mini-batches with virtual embeddings called *proxies* [14, 15], and *iv*) suggesting training strategies upon characterization of the generalization bounds [16, 17, 115]. While such approaches are based on test and training samples to be drawn from the same distribution, the intended behaviour for a trained embedding function is generalization to unseen classes (*i.e.*,*zero-shot* performance) which is commonly achieved by *early-stopping* regularization with validation performance monitoring. That being said, it is widely observed that the generalization in training domain indeed transfer to the test domain. Then a critical question is *"How does embedding function transfer its training?"*

In this chapter, we aim to address that question for the embedding functions having global average pooling (GAP) as the global pooling layer. In other words, we are to discuss the particular family of the embedding functions that yield a single vector for an input image by aggregating the local features extracted by CNN. The findings we present in this chapter are the foundations of our motivation for the method we develop in Ch. 5.

4.1 Preliminaries

In this section we set up the notation that we use throughout the chapter and give some background of which we make use while analyzing the behavior of global average pooling.

We use capital letters (*e.g.* X) to represent the matrices, $X = (x_{ij})_{ij} \in \mathbb{R}^{n\times m}$. The vectors, $x = (x_i)_i \in \mathbb{R}^n$, or scalars, $\alpha \in \mathbb{R}$, are represented by lowercase letters. We express an m by n matrix, X, with its column vectors, $x_i \in \mathbb{R}^d$, as $X = [x_i]_{i\in[n]}$ where $[n] = \{1, \ldots, n\}$.

The simplex in $\mathbb{R}^n$ is denoted as

$$\Sigma_n \coloneqq \{p \in \mathbb{R}^n_{\geqslant 0} \mid \textstyle\sum_i p_i = 1\}. \tag{4.1.1}$$

We use 2-tuple, $(p, X) \in \Sigma_n \times \mathbb{R}^{d\times n}$, to denote a probability mass distribution with masses, $p \in \Sigma_n$, and d-dimensional support, $X = [x_i]_{i\in[n]} \in \mathbb{R}^{d\times n}$.

Definition 4.1.1 (Optimal Transport Distance) *The optimal transport (OT) distance between two probability mass distributions,* (p, X) *and* (q, Y)*, is:*

$$\|(p, X) - (q, Y)\|_{OT} = \min_{\substack{\pi\geqslant 0 \\ \Sigma_i \pi_{ij} = q_j \\ \Sigma_j \pi_{ij} = p_i}} \textstyle\sum_{ij} c_{ij}\pi_{ij} \tag{4.1.2}$$

where $c_{ij} = \|x_i - y_j\|_2$.

Optimal transport distance is a meta distance which is the optimal value of a transportation problem [116]. Optimal transport distance measures the minimum amount of cost to move masses from one support to another to match the masses. Hence, it can measure the dissimilarity between two mass functions.

Definition 4.1.2 (Maximum Mean Discrepancy) *Maximum mean discrepancy (MMD) between two probability mass distributions,* (p, X) *and* (q, Y)*, is:*

$$\|(p, X) \text{ - } (q, Y)\|_{MMD} = \max_{f \in \mathcal{C}(X,Y)} \textstyle\sum_i p_i f(x_i) - \sum_j q_j f(y_j) \tag{4.1.3}$$

where $\mathcal{C}(X, Y)$ *is the set of continuous and bounded functions defined on a set covering the column vectors of* X *and* Y*.*

Similar to OT, MMD is a metric to measure the dissimilarity of two mass functions. OT is indeed an MMD-based metric. Such a relation is more explicit once we write the Lagrangian dual of the OT distance.

Definition 4.1.3 (Optimal Transport Distance Dual) *The Lagrangian dual of the optimal transport distance defined in Definition 4.1.1 reads:*

$$\|(p, X) \text{ - } (q, Y)\|_{OT} = \max_{f_i + g_j \leqslant c_{ij}} \textstyle\sum_i p_i f_i + \sum_j q_j g_j \tag{4.1.4}$$

with the dual variables $\lambda = \{f, g\}$*.*

Note that $x_i = y_j$ implies $f_i = -g_j$ and from the fact that $c_{ij} = c_{ji}$, we can express the problem in Eq. (4.1.4) as:

$$\|(p, X) \text{ - } (q, Y)\|_{OT} = \max_{f \in \mathfrak{L}_1} \textstyle\sum_i p_i f(x_i) - \sum_j q_j f(x_j) \tag{4.1.5}$$

where $\mathfrak{L}_1 = \{f \mid \sup\limits_{x,y} \frac{|f(x) - f(y)|}{\|x-y\|_2} \leqslant 1\}$ is the set of 1-Lipschitz functions.

In the next section, we use OT and MMD to show that global average pooling maps the collection of local features to a point which lies in a convex of hull and that the vertices of such a convex hull are the prototype vectors of some semantic entities.

4.2 Analysis of Global Average Pooling

Given n-many convolutional features, $\{x_i\}_{i \in [n]}$, of an image, global average pooling computes the global representation, x_g, of the image as:

$$x_g = \textstyle\sum_i \frac{1}{n} x_i \tag{4.2.1}$$

4.2.1 Classification Problem

We first consider a typical classification task where x_g is passed through a *classification layer*, *i.e.*,a linear transformation followed by *soft-max* operation. Specifically, for a c-class problem, we have c-many vectors, $[\nu{\in}\mathbb{R}^d]_{i\in[c]}$ corresponding to class representatives (or equivalently class means or class proxies). We obtain similarity to each class as,

$$a_i = \nu_i^\intercal x_g \tag{4.2.2}$$

and then apply *soft-max* normalization on a to obtain predicted class probabilities,

$$\hat{p}(class_i \mid x_g) = \frac{\exp(a_i)}{\sum_j \exp(a_j)}. \tag{4.2.3}$$

The critical desiderata for $\hat{p}(class_i \mid x_g)$ is to have a peak at the index corresponding to the true label and to have a low entropy (*i.e.*,only the true class index has the peak). To achieve such behavior, *cross-entropy* loss is typically used to optimize the feature extractor parameters. We now investigate how global average pooling shapes the learning process.

Obtaining class similarities $a_i = \nu_i^\intercal x_g$ is a linear operation and so is global average pooling. We can interchange the order of the global average pooling and linear transformation. Namely, we can represent a_i as:

$$a_i = \textstyle\sum_j \frac{1}{n} \nu_i^\intercal x_j. \tag{4.2.4}$$

From Eq. (4.2.4), we observe that the local features of an image must be aligned with the direction of the corresponding class feature. In other words, to have large a_i for an image from class i, the dot products with the features, $\nu_i^\intercal x_j$, must be high and such a condition must hold for as much as local features as possible to have a large sum $\sum_j \frac{1}{n} \nu_i^\intercal x_j$. On the other hand, the local features must give low similarity values with the class features of the other classes (*i.e.*,$\nu_{i'}^\intercal x_j$ must be small for $i' \neq i$). Therefore, using global representations obtained by global average pooling results in local features that are concentrated around a particular vector and that particular vector represents the class semantic.

Such a behavior of the global average pooling is empirically studied in [117]. The results support that local features represent semantic entities corresponding to their classes. Nevertheless, the implications on how the learning with global average pooling can transfer to the domain of unseen classes are unclear. We now extend our discussions to pairwise loss based metric learning setting and expand on the empirical studies performed in [117] to provide a more general explanation to success of global average pooling in transferring the learning to unseen classes.

4.2.2 Metric Learning Problem

We see in § 4.2.1 that we should have local features concentrated around some prototype vectors that correspond to class semantics if we are to use global average pooling in a classification problem. Such a behaviour is shaped by the cross-entropy loss serving the classification objective. We now extend our analysis to metric learning setting.

In metric learning, we want the distance between two samples to reflect the semantic dissimilarity. Namely, given two representations, x_g and x'_g, we want $\|x_g - x'_g\|_2$ to be low whenever x_g and x'_g are of the same class and high otherwise. To achieve this, metric learning methods [6] use pairwise distance bases losses (*e.g.* contrastive Eq. (2.1.1)). We want to find out whether we can explain the mapping of the global average pooling in terms of some semantic vectors.

Prior to moving forward, we introduce an operator to compose a histogram representation from the collection of features.

Definition 4.2.1 (Histogram Operator) *Given n-many d-dimensional features, $X = [x_i{\in}\mathbb{R}^d]_{i\in[n]}$, and m-many prototype features of the same dimension, $V = [\nu_i{\in}\mathbb{R}^d]_{i\in[m]}$, the histogram of X on V is denoted as p^* which is computed as the minimizer of the following problem:*

$$(p^*, \pi^*) = \underset{\substack{\Sigma_i \pi_{ij} = 1/n \\ \Sigma_j \pi_{ij} = p_i \\ p \in \Sigma_m, \pi \geqslant 0}}{\arg\min} \textstyle\sum_{ij} c_{ij}\pi_{ij} \tag{4.2.5}$$

where $c_{ij} = \|\nu_i - x_j\|_2$.

Claim 4.2.1 *The solution of the problem in Eq.* (4.2.5) *reads:*

$$\pi^*_{ij} = {}^1\!/\!_n \mathbb{1}(i = \arg\min_k \{c_{kj}\}) \tag{4.2.6}$$

where $\mathbb{1}(A)$ *is 1 whenever* A *is true and 0 otherwise.*

Proof: We prove our claim by contradiction. For any j, we express a solution as $\pi^*_{ij} = \epsilon_i$ with $\epsilon_i \geqslant 0$ and $\sum_i \epsilon_i = {}^1\!/\!_n$. Let $i^* = \arg\min_k\{c_{kj}\}$. We can write $\pi^*_{i^*j} = {}^1\!/\!_n - \sum_{i|i\neq i^*} \epsilon_i$. Our claim states that $\epsilon_i = 0$ for $i \neq i^*$. We assume an optimal solution, π', with $\epsilon_i > 0$ for some $i \neq i^*$. Since π' is optimal, we must have $\sum_{ij} \pi'_{ij} c_{ij} \leqslant \sum_{ij} \pi_{ij} c_{ij}$ for any π. For the j^{th} column we have,

$$\begin{aligned}\textstyle\sum_i \pi'_{ij} c_{ij} &= (\tfrac{1}{n} - \sum_{i'|i'\neq i^*} \epsilon_{i'}) c_{i^*j} + \sum_{i'|i'\neq i^*} \epsilon_{i'} c_{i'j} \\ &= \tfrac{1}{n} c_{i^*j} + \sum_{i'|i'\neq i^*} \epsilon_{i'} (c_{i'j} - c_{i^*j}) \overset{(a)}{>} \textstyle\sum_i \pi^*_{ij} c_{ij}\end{aligned}$$

where in (a) we use the fact that $(c_{i'j} - c_{i^*j}) > 0$ and $\epsilon_{i'} > 0$ for some i' by the assumption. Hence, $\sum_{ij} \pi'_{ij} c_{ij} > \sum_{ij} \pi^*_{ij} c_{ij}$ poses a contradiction. Therefore, $\epsilon_{i'} = 0$ must hold for all $i' \neq i^*$. ■

In words, histogram operator basically assign each feature to their nearest prototype and accumulates ${}^1\!/\!_n$ mass for each assigned feature. Thus, p^* is a probability mass distribution which masses are proportional to the number of features assigned to the corresponding prototypes.

We now consider a set of prototypes in the feature space where the convolutional features, x_i, lie. We consider m-many prototype features, $\mathcal{V} = \{\nu_i\}_{i\in[m]}$, so that the set $\mathcal{V}$ is δ-cover of the feature space, $\mathcal{X}$. Namely, for any $x \in \mathcal{X}$, we have a prototype ν_x such that $\|x - \nu_x\|_2 \leqslant \delta$.

Given n-many convolutional features, $X = [x_i]_{i\in[n]}$, and m-many prototype features of the same dimension, $V = [\nu_i]_{i\in[m]}$, we compute the histogram of X on V (*i.e.*,p^*) using Eq. (4.2.5). Hence, we obtain a probability mass distribution with (p^*, V). We also consider our feature collection $X = [x_i]_{i\in[n]}$ as a uniform mass distribution with (q, X) where $q_i = {}^1\!/\!_n$ for all i. Note that global average pooling performs $\sum_i q_i x_i$. We show that such an operation is *approximately* equivalent to $\sum_i p^*_i \nu_i$, *i.e.*,*convex combination of prototypes.* To show such equivalence, we consider the error between the two representations: $\| \sum_i p^*_i \nu_i - \sum_j q_j x_j \|^2_2$.

Proposition 4.2.1 *Given n-many convolutional features, $X = [x_i \in \mathcal{X}]_{i\in[n]}$, and m-many prototype features, $V = [\nu_i]_{i\in[m]}$, with $\{\nu_i\}_{i\in[m]}$ being δ-cover of $\mathcal{X}$. If p^* is the histogram of X on V, defined in Eq.* (4.2.5)*, then we have:*

$$\| \sum_{i\in[m]} p_i^* \nu_i - \sum_{j\in[n]} \tfrac{1}{n} x_j \|_2^2 \leqslant \delta$$

Proof: We can express

$$\| \sum_{i\in[m]} p_i^* \nu_i - \sum_{j\in[n]} \tfrac{1}{n} x_j \|_2^2 = \sum_{i\in[m]} p_i^* f(\nu_i) - \sum_{j\in[n]} q_j f(x_j)$$

where $f(x) = x^{\intercal}(\sum_i p_i^* \nu_i - \sum_j \frac{1}{n} x_j)$. Note that f is a continuous bounded operator for $\mathcal{X} = \{x \mid \|x\|_2 \leqslant 1\}$ (We can always map the features inside unit sphere without loosing the relative distances). Moreover, the operator norm of f, *i.e.*,$\|f\|$, which is $\| \sum_i p_i^* \nu_i - \sum_j \frac{1}{n} x_j \|_2$ is less than or equal to 1. Thus, f lie in the unit sphere of the continuous bounded functions set. Using the definition of MMD distance, we can bound the error as:

$$\sum_{i\in[m]} p_i^* f(\nu_i) - \sum_{j\in[n]} q_j f(x_j) \leqslant \|(p^*, V) - (q, X)\|_{MMD}$$

where $q_i = {}^1/_n$ for all i. For the continuous and bounded functions of the operator norm less than 1, MMD is lower bound for OT [118]. Namely,

$$\sum_{i\in[m]} p_i^* f(\nu_i) - \sum_{j\in[n]} q_j f(x_j) \leqslant \|(p^*, V) - (q, X)\|_{MMD} \leqslant \|(p^*, V) - (q, X)\|_{OT}.$$

Since columns of V is δ-cover of the set $\mathcal{X}$, the optimal transport distance between the two distributions are bounded by δ, *i.e.*,$\|(p^*, V) - (q, X)\|_{OT} \leqslant \delta$. Thus, we finally have:

$$\| \sum_{i\in[m]} p_i^* \nu_i - \sum_{j\in[n]} \tfrac{1}{n} x_j \|_2^2 \leqslant \delta.$$

■

We visualize the result of Proposition 4.2.1 in Fig. 4.1. Proposition 4.2.1 implies that the representation space defined via output of global average pooling is a convex combination of the prototype features that form a δ-cover of the representation space. Hence, each image is mapped in a convex hull where the vertices are the prototype features. What we desire in class-level metric learning is non-overlapping class convex hulls. Once trained with a metric learning loss, clearly, such prototypes are to correspond to some semantic entity corresponding to classes since CNN assign an embedding vector to a pixel according to the semantic meaning of the spatial extent that the corresponding pixel represents. In the next section, we validate our claims with empirical studies and then bring an explanation on how the representations obtained by global average pooling is transferred to unseen classes.

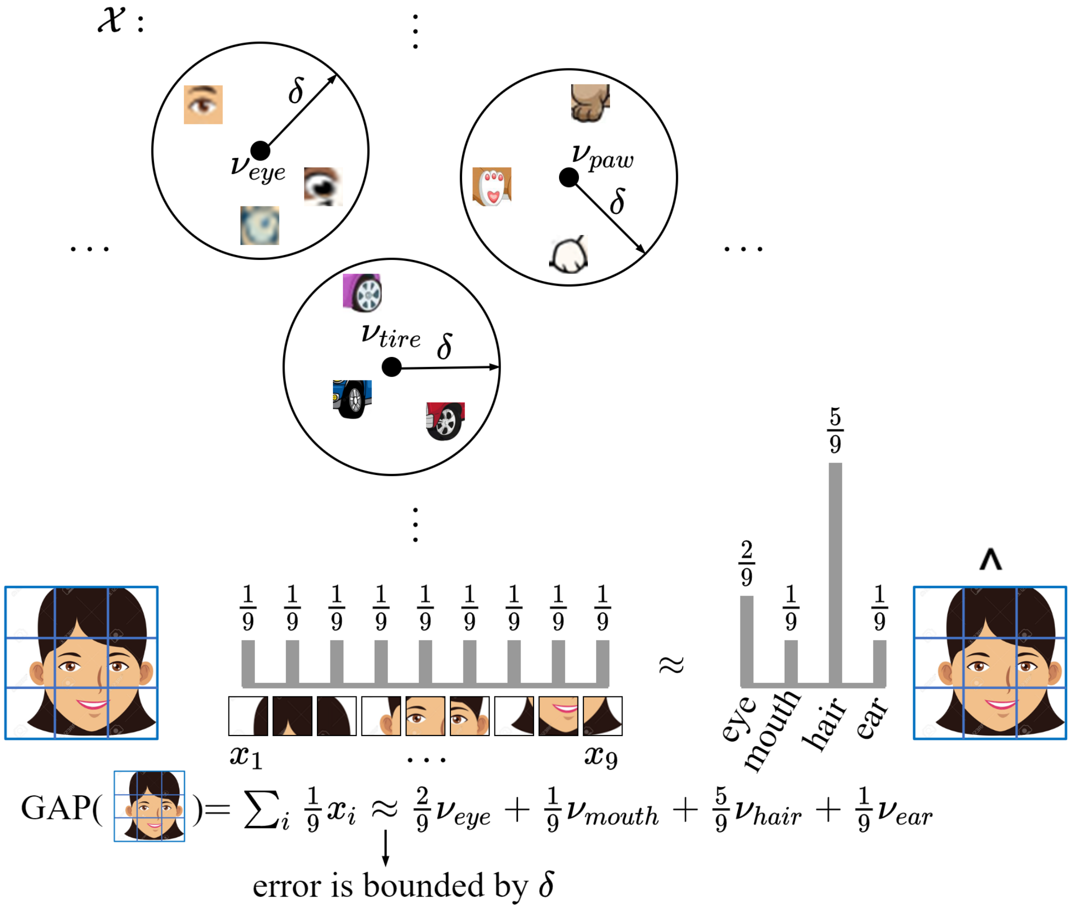

$\mathcal{X}$:
δ
ν_{eye}
ν_{paw}
ν_{tire}
$\frac{1}{9}$ $\frac{1}{9}$ $\frac{1}{9}$ $\frac{1}{9}$ $\frac{1}{9}$ $\frac{1}{9}$ $\frac{1}{9}$ $\frac{1}{9}$ $\frac{1}{9}$
x_1 … x_9
≈
$\frac{2}{9}$ $\frac{1}{9}$ $\frac{5}{9}$ $\frac{1}{9}$
eye
mouth
hair
ear
GAP( )= $\sum_i \frac{1}{9} x_i \approx \frac{2}{9}\nu_{eye} + \frac{1}{9}\nu_{mouth} + \frac{5}{9}\nu_{hair} + \frac{1}{9}\nu_{ear}$
error is bounded by δ

Figure 4.1: Visualization of Proposition 4.2.1: equivalence of global average pooling to convex combination of prototype vectors corresponding to semantic entities. Once local patches are embedded to a vector space according to the semantic meaning, collection of local patches captures the latent semantic representation of the image.

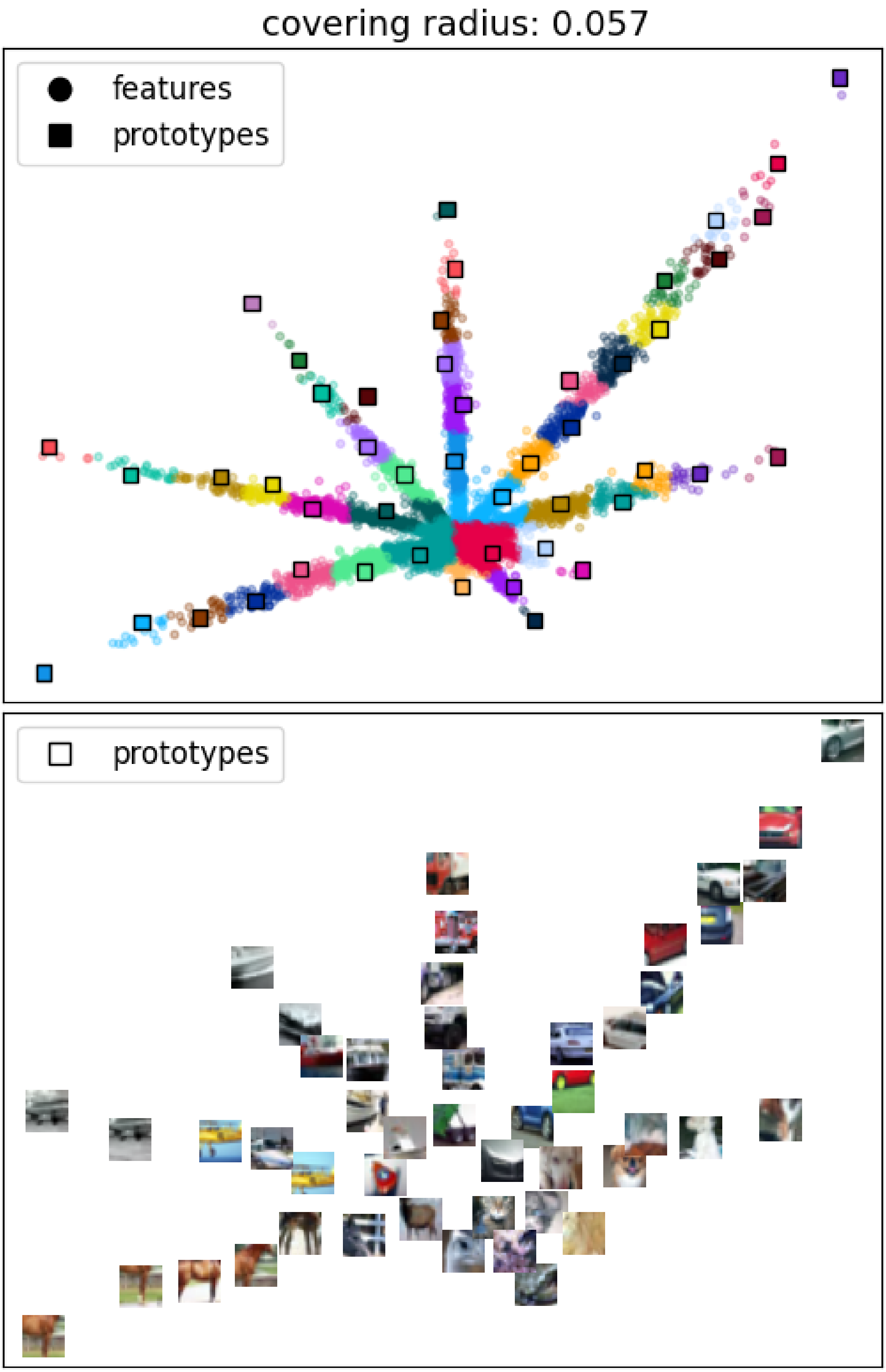


Figure 4.2: Empirical illustration of the local feature distribution and feature prototypes of Cifar10 dataset on the 2D embedding space. We observe that prototypes correspond to some semantic entities.

4.3 Empirical Validation

We perform a metric learning training on Cifar10 [119] dataset using ResNet20 [105] architecture. We use 2D feature embeddings for direct visualization. Namely, the local features of the image is mapped to 2D space and accordingly, so is the global representation obtained by average pooling.

We scale the features so that the magnitude of the features are less than 1. We sample 64 images from each class and obtain the local features as well as the global features. We compute 48-many prototypes among the local features using the *greedy k-center* algorithm outlined in Algorithm 2. We plot the set partition and the prototypes in Fig. 4.2. We also provide the covering radius (*i.e.*,δ in δ-cover) of the prototype set. We observe that prototypes correspond to some semantic entities.

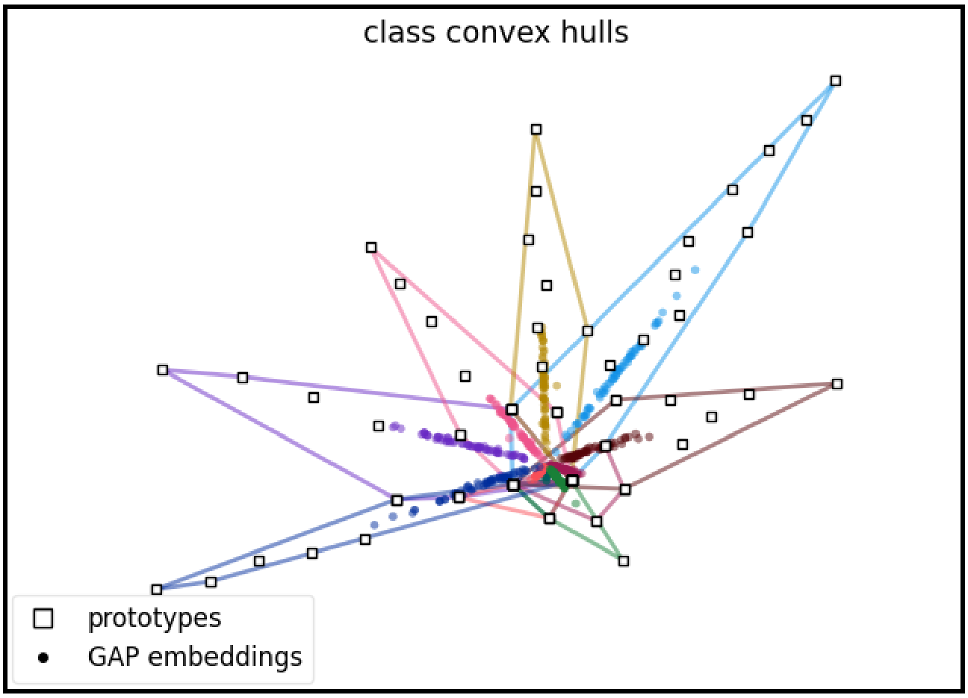


Figure 4.3: Illustration of class convex hulls where the vertices are the active prototypes for the corresponding class. Points are colored with respect to class labels. We observe overlapping class convex hulls, meaning that the samples of different classes are likely to be mapped close.

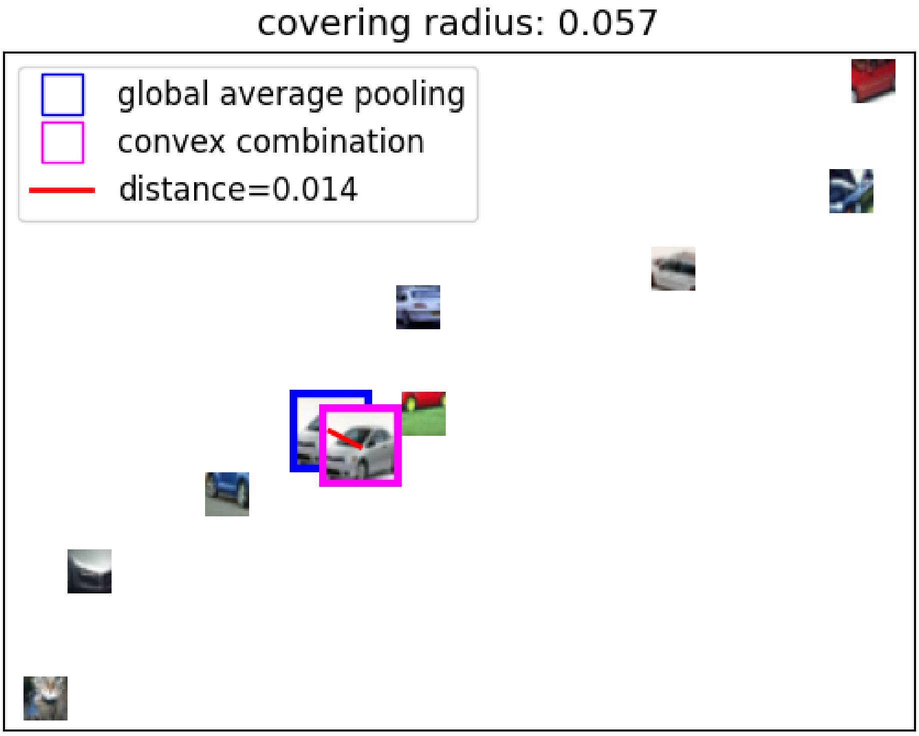


Figure 4.4: GAP and PCC embedding of an image from *car* class.

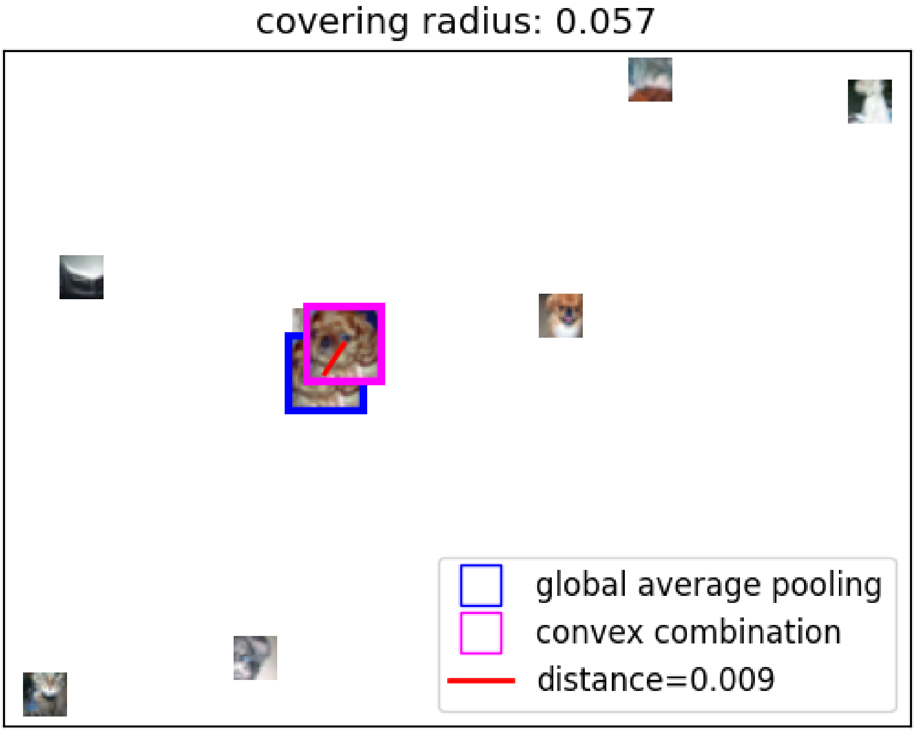


Figure 4.4 (cont'd): GAP and PCC embedding of an image from *dog* class.

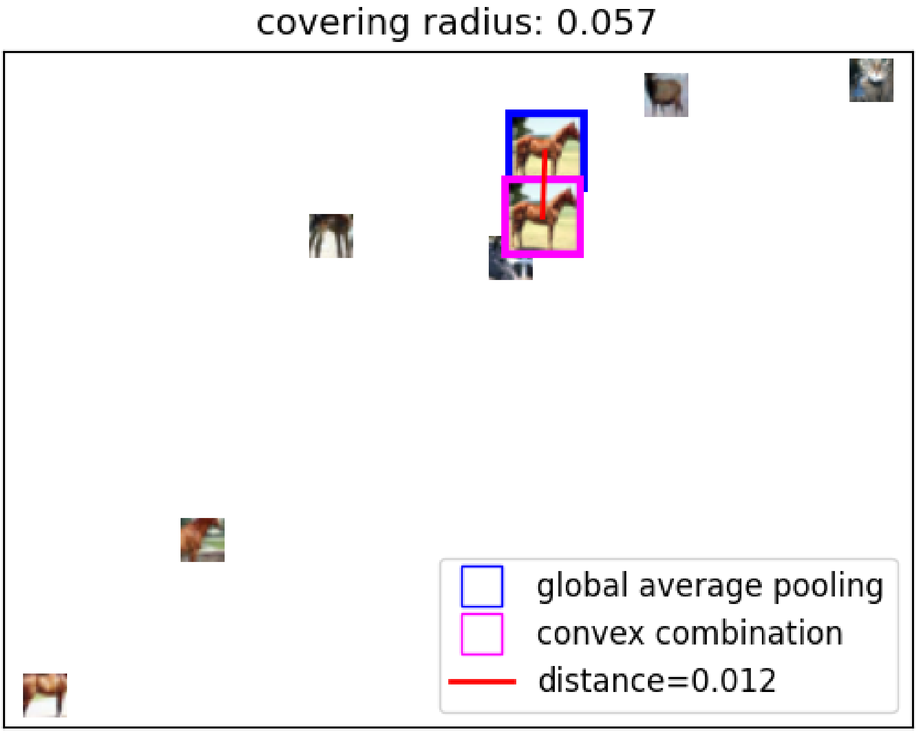


Figure 4.4 (cont'd): GAP and PCC embedding of an image from *car* class. For sample images of several classes (*dog, car, horse*), we plot their GAP embedding as well as the PCC embedding where each local feature is assigned to its closest prototype and convex combination of the prototypes yields the global embedding vector. Combination coefficients are proportional to the amount of local features assigned to the prototypes. Empirically validating our claims on the behavior of GAP, the distance between the two representations are bounded by the covering radius of the prototypes.

To validate our claims on convex combination of the prototypes, we compare the global average pooling (GAP) embeddings and prototype convex combination embeddings (PCC) in Fig. 4.4. In PCC, each local feature is assigned to its closes prototype and the prototype histogram is computed as in Eq. (4.2.5). Then, histogram weights are used as the combination weights. We observe that the error between the two distributions are always smaller than the covering radius, validating the result of Proposition 4.2.1. We also observe that different prototypes are active for different classes while some prototypes are shared among the classes.

We also plot the class convex hulls in Fig. 4.3 to show that each class has its own convex hull. We determine the set of active prototypes for each class and then obtain the convex hull of those prototypes. We observe that the class convex hulls overlaps due to the shared prototypes among the classes. To this end, learning to discriminate local semantics to have non-overlapping class convex hulls can be a decisive step towards improving the generalization of DML.

4.4 Discussion

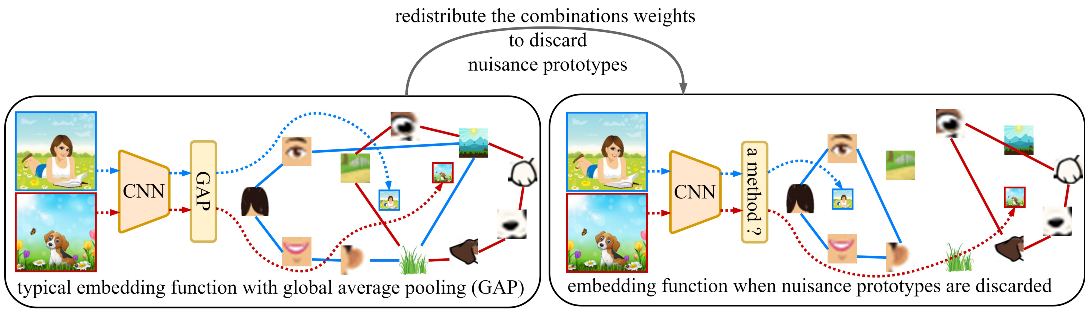

redistribute the combinations weights
to discard
nuisance prototypes
CNN
GAP
typical embedding function with global average pooling (GAP)
CNN
a method ?
embedding function when nuisance prototypes are discarded

Figure 4.5: The big picture showcasing our analysis on the behavior of GAP. *A method* that is able to control the learning of the prototypes as well as their convex combination weights can yield non-overlapping class convex hulls, meaning that the samples of different classes are mapped apart as desired.

We show that the representation space defined via output of GAP is a convex combination of semantically independent representations defined by each pixel in the feature map. Such behaviour of GAP forces CNN to assign an embedding vector to a pixel according to the semantic meaning of the spatial extent that the corresponding pixel represents.

According to our analysis, collection of CNN's output features corresponds to a histogram of some visual vocabulary once the features inherit the semantic meaning of the local regions. Namely, if the local features are mapped in close vicinity of their corresponding *prototype features* of some semantic meanings (*e.g.* all *"tire"* patches in a *"car"* image are mapped very close), then CNN feature map corresponds to a histogram of the prototypes. Averaging the local features can be seen as convex combination of prototypes, where the weights are proportional to the occurrence frequency of the corresponding semantic feature vectors. Hence, each image is mapped in a convex hull where the vertices are the prototype features.

Using our analysis on local feature geometry shaped by GAP, we can explain the effectiveness of GAP in transferring the learning to unseen classes. As long as the local semantics of the training and test samples follow the same distribution, the embedding function can transfer its training. In other words, a metric learning algorithm can generalize to unseen classes if such classes share the same local semantics. As we also empirically observe, the class samples are mapped to a convex hull. Hence, if a new class can be expressed by the learned prototypes, then that class will also have its convex hull.

In our analysis, we conclude that the class convex hulls are shaped by two factors: prototype features and the combination weights. Illustrated in Fig. 4.5, GAP equally considers each prototype and such a behavior results in overlapping convex hulls due to shared prototypes among the classes. To have non-overlapping class convex hulls, we must only use class discriminative features as our prototypes. Nevertheless, such an operation requires local-level label annotations which does not exist in class-supervised metric learning setting. Therefore, with the current DML approaches, none of prototypes nor combination weights is explicitly learned. In the next chapter, we present a learnable and generalized version of GAP which improves GAP with two distinct abilities: *i*) the ability to choose a subset of semantic entities, effectively learning to ignore nuisance information, and *ii*) learning the weights corresponding to the importance of each entity.

CHAPTER 5

GENERALIZED SUM POOLING FOR DEEP METRIC LEARNING

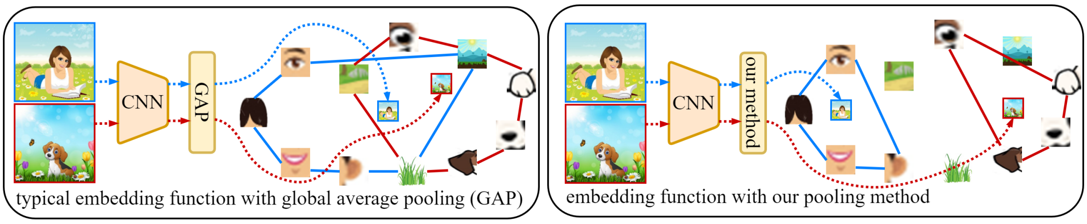

CNN
GAP
typical embedding function with global average pooling (GAP)
CNN
our method
embedding function with our pooling method

Figure 5.1: Effect of global pooling of the feature map and our method.

Distance metric learning (DML) addresses the problem of finding an embedding function such that the semantically similar samples are embedded close to each other while the dissimilar ones are placed relatively apart in the Euclidean sense. We an extensive review of DML literature in Ch. 2. Although the prolific and diverse literature of DML includes various architectural designs [54, 62], loss functions [6], and data-augmentation techniques [7, 13], many of these methods have a shared component: a convolutional neural network (CNN) followed by a global pooling layer, mostly global average pooling (GAP) [6, 7].

As we discuss in Ch. 4, the effectiveness of GAP can be explained by considering each pixel of the CNN feature map as corresponding to a separate semantic entity. For example, spatial extent of one pixel can correspond to a *"tire"* object making the resulting feature a representation for *"tireness"* of the image. If this explanation is correct, the representation space defined via output of GAP is a convex combination of semantically independent representations defined by each pixel in the feature map. Although this folklore is later empirically studied in [117] and further verified for classification in [120], its algorithmic implications are not clear. If each feature is truly representing a different semantic entity, *should we really average over all of them?* Surely, some classes belong to the background and should be discarded as nuisance variables. Moreover, *is uniform average of them the best choice? Aren't some classes more important than others?* In this chapter, we try to answer these questions within the context of metric learning. We propose a learnable and generalized version of GAP which learns to choose the subset of the semantic entities to utilize as well as weights to assign them while averaging.

In order to generalize the GAP operator to be learnable, we re-define it as a solution of an optimization problem. We let the solution space to include 0-weight effectively enabling us to choose subset of the features as well as carefully regularize it to discourage degenerate solution of using all the features. Crucially, we rigorously show that the original GAP is a specific case of our proposed optimization problem for a certain realization. Our proposed optimization problem closely follows optimal transport based *top-k* operators [92] and we utilize its literature to solve it. Moreover, we present an algorithm for an efficient computation of the gradients over this optimization problem enabling learning. A critical desiderata of such an operator is choosing subset of features which are discrimantive and ignoring the background classes corresponding to nuisance variables. Although supervised metric learning losses provide guidance for seen classes, they carry no such information to generalize the behavior to unseen classes. To enable such a behavior, we adopt a *zero-shot prediction loss* as a regularization term which builds on expressing the class label embeddings as a convex combination of attribute embeddings [94–96].

In order to validate the theoretical claims, we design a synthetic empirical study. The results confirm that our pooling method chooses better subsets and improve generalization ability. Moreover, our method can be applied with any DML loss as GAP is a shared component of them. We applied our method on 6 DML losses and test on 4 datasets. Results show consistent improvements with respect to direct application of GAP.

5.1 Preliminaries

In this section, we recapitulate the DML formulation and the related notation that we set up in § 1.1 for the sake of self-completeness of the chapter. We as well extend the base notation to avoid convoluted notation throughout the chapter.

Consider the data distribution $p_{\mathcal{X}\times\mathcal{Y}}$ over $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the space of data points and $\mathcal{Y}$ is the space of labels. Given *iid.* samples from $p_{\mathcal{X}\times\mathcal{Y}}$ as $\{(x_i, y_i)\}$, distance metric learning problem aims to find the parameters θ of an embedding function $e(\cdot;\theta)$: $\mathcal{X} \rightarrow \mathbb{R}^d$ such that the Euclidean distance in the space of embeddings is consistent with the label information where d is the embedding dimension. More specifically, $\|e(x_i;\theta) - e(x_j;\theta)\|_2$ is small whenever $y_i = y_j$, and large whenever $y_i \neq y_j$. In order to enable learning, this requirement is represented via loss function $l((x_i, y_i), (x_j, y_j); \theta)$ (*e.g. contrastive* [9] $l((x_i, y_i), (x_j, y_j); \theta) = \max\{0, \mathbb{1}_{y_i=y_j}(\|e(x_i;\theta) - e(x_j;\theta)\|_2 - \beta) + \alpha\}$ where $\mathbb{1}_A$ is an indicator function which is 1 whenever A is true and -1 otherwise).

The typical learning mechanism is gradient descent of an empirical risk function defined over a batch of data points B. To simplify notation throughout the paper, we will use $b = \{b(i) \mid x_i, y_i \in B\}_i$ to index the samples in a batch. Then, the typical empirical risk function is defined as:

$$\mathcal{L}_{DML}(b;\theta) \coloneqq \tfrac{1}{|b|^2} \sum_{i\in b} \sum_{j\in b} l((x_i, y_i), (x_j, y_j); \theta)\,. \tag{5.1.1}$$

We are interested specific class of embedding functions where a global average pooling is used. Specifically, consider the composite function family $e = g \circ f$ such that g is pooling and f is feature computation. We assume a further structure over the functions g and f. The feature function f maps the input space $\mathcal{X}$ into $\mathbb{R}^{w \times h \times d}$ where w and h are spatial dimensions. Moreover, g performs averaging as;

$$g(f(x;\theta)) = \frac{1}{w \cdot h} \sum_{i\in[w\cdot h]} f_i\,, \tag{5.1.2}$$

where $[n] = 1, \ldots, n$ and we let $f_i \in \mathbb{R}^d$ denote i^{th} spatial feature of $f(x;\theta)$ to avoid convoluted notation. In the rest of the chapter, we generalize the pooling function g into a learnable form and propose an algorithm to learn it.

5.2 Method

Consider the pooling operation in Eq. (5.1.2), it is a simple averaging over pixel-level feature maps (f_i). As we discuss in Ch. 4, one explanation for the effectiveness of this operation is considering each f_i as corresponding to a different semantic entity corresponding to the spatial extend of the pixel, and the averaging as convex combination over these semantic classes. Our method is based on generalizing this averaging such that a specific subset of pixels (correspondingly subset of semantic entities) are selected and their weights are adjusted according to their importance.

We generalize Eq. (5.1.2) in § 5.2.1 by formulating a feature selection problem in which we prioritize a subset of the features that are closest to some trainable prototypes. If a feature is to be selected, its weight will be high. We then formulate our pooling operation as a differentiable layer so that the prototypes can be learned along with the rest of the embedding function parameters in § 5.2.2. We learn the prototypes with class-level supervision, however in metric learning, learned representations should generalize to unseen classes. Thus, we introduce a zero-shot prediction loss to regularize prototype training for zero-shot setting in § 5.2.3.

5.2.1 Generalized Sum Pooling as a Linear Program

Consider the pooling function, g, with adjustable weights as $g(f(x;\theta);\omega) = \sum_{i\in[n]} p_i f_i$ where $n = w\,h$. Note that, $p_i = {}^1\!/_n$ corresponds to average pooling. Informally, we want to control the weights to ease the metric learning problem. Specifically, we want the weights corresponding to background classes to be 0 and the ones corresponding to discriminative features to be high.

If we were given representations of discrimantive semantic entities, we could simply compare them with the features (f_i) and choose the ones with high similarity. Our proposed method is simply learning these representations and using them for weight computations. We first discuss the weight computation part before discussing learning the representations of prototypes.

Assume that there are m discrimantive semantic entities which we call *prototypes* with latent representations $\omega = \{\omega_i\}_{i\in[m]}$ of appropriate dimensions (same as f_i). Since we know that not all features ($\{f_i\}_{i\in[n]}$) are relevant, we need to choose a subset of $\{f_i\}_{i\in[n]}$. We perform this top-k selection process by converting it into an optimal transport (OT) problem.

Consider a cost map $c_{ij} = \|\bar{\omega}_i - \bar{f}_j\|_2$ which is an m (number of prototypes) by n (number of features) matrix representing the closeness of prototypes ω_i and features f_j after some normalization $\bar{u} = {}^{u}\!/_{\max\{1,\|u\|_2\}}$. We would like to find a transport map π which re-distributes the uniform mass from features to prototypes. Since we do not have any prior information over features, we also consider its marginal distribution (importance of each feature to begin with) to be uniform. As we need to choose a subset, we set $\mu\in[0,1]$ ratio of mass to be transported. The resulting OT problem is:

$$\rho^*, \pi^* = \mathop{\arg\min}_{\substack{\rho,\pi\geqslant 0\\ \rho_j+\sum_i\pi_{ij}={}^{1}\!/_{n}\\ \sum_{ij}\pi_{ij}=\mu}} \textstyle\sum_{ij} c_{ij}\pi_{ij}. \tag{P1}$$

Different to typical OT literature, we introduce decision variables, ρ, to represent residual weights to be discarded. Specifically modelling discarded weight instead of enforcing another marginalization constraint is beneficial beyond stylistic choices as it allows us to very efficient compute gradients. When the introduced transport problem is solved, we perform weighting using residual weights as:

$$g(f(x;\theta);\omega) = \textstyle\sum_i p_i f_i = \sum_i \frac{{}^{1}\!/_{n}-\rho_i^*}{\mu} f_i \tag{5.2.1}$$

Given set of prototypes $\{\omega_i\}_{i\in[m]}$, solving the problem in (P1) is a strict generalization of GAP since setting $\mu = 1$ recovers the original GAP. We formalize this equivalence in the following claim.

Claim 5.2.1 *If* $\mu = 1$*, the operation in Eq.* (5.2.1) *reduces to global average pooling in Eq.* (5.1.2)*.*

We defer the proof to Appendix. Having generalized GAP to a learnable form, we introduce a method to learn the prototypes $\{\omega_i\}_{i\in[m]}$ in the next section.

5.2.2 Generalized Sum Pooling as a Differentiable Layer

Consider the generalized form of pooling, defined as solution of (P1), as a layer of a neural network. The input is the feature vectors $\{f_i\}_{i\in[n]}$, the learnable parameters are prototype representations $\{\omega_i\}_{i\in[m]}$, and the output is residual weights ρ^*. To enable learning, we need partial derivatives of ρ^* with respect to $\{\omega_i\}_{i\in[m]}$. However, this function is not smooth. More importantly it requires the μ parameter to be known a priori.

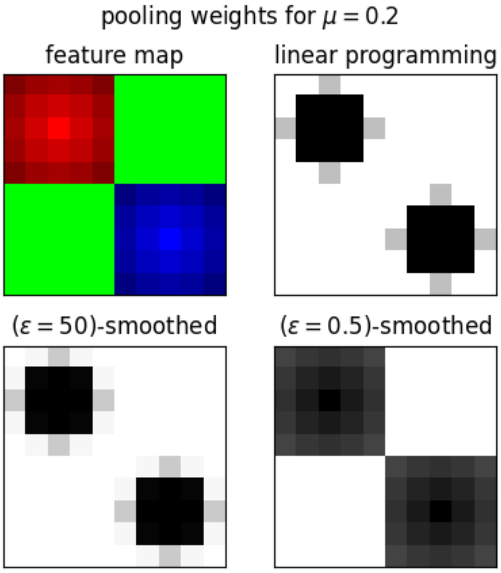


Figure 5.2: 10x10x3 feature map (top-left) with 5x5 reddish and bluish features to be pooled and the resultant pooling weights (higher the darker) of different problems with red $(1, 0, 0)$ and blue $(0, 0, 1)$ prototypes, ω.

We use a toy example to set the stage for rest of the formulation. Consider a feature map of dimension 10x10x3 visualized as RGB-image and corresponding two prototypes with representations $(1, 0, 0)$ (red) and $(0, 0, 1)$ (blue). The true

$\mu = 0.5$ since the half of the image corresponds to red and blue, and other half is background class of green. Consider an under-estimation of $\mu = 0.2$, the global solution (shown as linear programming) is explicitly ignoring informative pixels (part of red and blue region). To solve this issue, we use entropy smoothing which is first introduced in [66] to enable fast computation of optimal transport. Consider the entropy smoothed version of the original problem in (P1) as:

$$\rho^{(\varepsilon)}, \pi^{(\varepsilon)} = \underset{\substack{\rho,\pi \geqslant 0 \\ \rho_j + \Sigma_i \pi_{ij} = {}^1\!/_n \\ \Sigma_{ij}\pi_{ij} = \mu}}{\arg\min} \textstyle\sum_{ij} c_{ij}\pi_{ij} + \frac{1}{\varepsilon}\left(\sum_{ij} \pi_{ij} \log \pi_{ij} + \sum_j \rho_j \log \rho_j\right), \tag{P2}$$

and obtain pooling weights by replacing ρ^* with $\rho^{(\varepsilon)}$ in Eq. (5.2.1). When smoothing is high ($\varepsilon \rightarrow 0$), the resulting solution is uniform over features similar to GAP. When it is low, the result is similar to top-k like behavior. For us, ε controls the trade-off between picking μ portion of the features that are closest to the prototypes and including as much features as possible for weight transfer.

We further visualize the solution of the entropy smoothed problem in Fig. 5.2 showing desirable behavior even with underestimated μ.

Beyond alleviating the under-estimation of μ problem, entropy smoothing also makes the problem strictly convex and smooth. Thus, the solution of the problem enables differentiation and in fact, admits closed-form gradient expression. We state the solution of (P2) and their corresponding gradients in the following propositions and defer their proofs to Appendix.

Proposition 5.2.1 *Given initialization* $t^{(0)} = 1$, *consider the following iteration:*

$$\rho^{(k+1)} = {}^1\!/_n \,(1 + t^{(k)} \exp(\text{-}\varepsilon c)^\intercal \mathbf{1}_m)^{\text{-}1}, \;\; t^{(k+1)} = \mu \,(\mathbf{1}_m^\intercal \exp(\text{-}\varepsilon c)\rho^{(k+1)})^{\text{-}1}$$

where exp *and* $(\cdot)^{\text{-}1}$ *are element-wise and* $\mathbf{1}_m$ *is m-dimensional vector of ones. Then,* $(\rho^{(k)}, t^{(k)})$ *converges to the solution of* (P2) *defining transport map via* $\pi^{(k)} = t^{(k)} \exp(\text{-}\varepsilon c) Diag(\rho^{(k)})$.

Proposition 5.2.2 *Given gradients* $\frac{\partial \mathcal{L}}{\partial \rho^{(\varepsilon)}}$ *and* $\frac{\partial \mathcal{L}}{\partial \pi(\varepsilon)}$, *with* $q = \rho^{(\varepsilon)} \odot \frac{\partial \mathcal{L}}{\partial \rho^{(\varepsilon)}} + (\pi^{(\varepsilon)} \odot$

$\frac{\partial \mathcal{L}}{\partial \pi^{(\varepsilon)}})^{\intercal}\mathbf{1}_n$ and $\eta = (\rho^{(\varepsilon)} \odot \frac{\partial \mathcal{L}}{\partial \rho^{(\varepsilon)}})^{\intercal}\mathbf{1}_n$ - $n\, q^{\intercal}\rho^{(\varepsilon)}$, the gradient with respect to c reads:

$$\frac{\partial \mathcal{L}}{\partial c} = \text{-}\gamma\left(\pi^{(\varepsilon)} \odot \frac{\partial \mathcal{L}}{\partial \pi^{(\varepsilon)}} - n\pi^{(\varepsilon)} Diag\Big(q \text{ - } \eta(1-\mu-n\rho^{(\varepsilon)\intercal}\rho^{(\varepsilon)})^{\text{-}1}\Big)\rho^{(\varepsilon)}\right) \quad , \tag{5.2.2}$$

where $\odot$ denotes element-wise multiplication.

Proposition 5.2.1 and 5.2.2 suggest that our feature selective pooling can be implemented as a differentiable layer. Moreover, Proposition 5.2.2 gives a matrix inversion free computation of the gradient with respect to the costs unlike optimal transport based operators [93]. Thus, the prototypes, ω, can be jointly learned with the feature extraction efficiently.

5.2.3 Cross-batch Zero-shot Regularization

Our feature selection layer should learn discriminative feature prototypes, ω, using top-down label information. Consider two randomly selected batches, (b_1, b_2), of data sampled from the distribution. If the prototypes are corresponding to discrimantive entities, the weights transferred to them (*i.e.*,marginal distribution of prototypes) should be useful in predicting the classes and such behavior should be consistent between batches for zero-shot prediction. Formally, if one class in b_2 does not exist in b_1, a predictor on class labels based on marginal distribution of prototypes for each class of b_1 should still be useful for b_2. Unfortunately, DML losses do not carry such information. We thus formulate a zero-shot prediction loss to enforce such zero-shot transfer.

We consider that we are given a semantic embedding vector for each of c-many class labels, $\Upsilon = [\upsilon_i]_{i\in[c]}$. We are to predict such embeddings from the marginal distribution of the prototypes. In particular, we use linear predictor, A, to predict label embeddings as $\hat{\upsilon} = A\,z$ where z is the normalized distribution of the weighs on the prototypes;

$$z = \tfrac{1}{\mu}\textstyle\sum_i \pi_i^{(\varepsilon)} \quad \text{where} \quad \pi^{(\varepsilon)} = [\pi_i^{(\varepsilon)}]_{i\in[n]}\,. \tag{5.2.3}$$

If we consider the prototypes as semantic vectors of some auxiliary labels such as attributes commonly used in ZSL [94], then we can interpret z as *pseudo-attribute* predictions. Given pseudo-attribute predictions, $\{z_i\}_{i\in b}$, and corresponding class embeddings for a batch, b, we fit the predictor as;

$$A_b = \underset{A=[a_i]_{i\in[m]}}{\arg\min}\; \textstyle\sum_{i\in b} \|A\, z_i \text{ - } \upsilon_{y_i}\|_2^2 + \epsilon \textstyle\sum_{i\in[m]} \|a_i\|_2^2\, . \tag{P3}$$

which admits a closed form expression enabling back propagation $A_b = \Upsilon_b\,(Z_b^{\intercal} Z_b + \epsilon I)^{-1} Z_b^{\intercal}$ where $\Upsilon_b = [\upsilon_{y_i}]_{i\in b}$, $Z_b = [z_i]_{i\in b}$. In practice, we are not provided with the label embeddings, $\Upsilon = [\upsilon_i]_{i\in[c]}$. Nevertheless, having a closed-form expression for A_b enables us to exploit a meta-learning scheme like [97] to formulate a zero-shot prediction loss to learn them jointly with the rest of the parameters.

Specifically, we split a batch, b, into two as b_1 and b_2 such that classes are disjoint. We then estimate attribute embeddings using one set and use that estimate to predict the label embeddings of the other set to form zero-shot prediction loss. Formally, our loss becomes:

$$\mathcal{L}_{ZS}(b;\theta) = \tfrac{1}{|b_2|} \sum_{i\in b_2} \log\left(1 + \sum_{j\in[c]} \mathrm{e}^{(\upsilon_j \text{-} \upsilon_{y_i})^{\intercal} A_1\, z_i}\right) + \tfrac{1}{|b_1|} \sum_{i\in b_1} \log\left(1 + \sum_{j\in[c]} \mathrm{e}^{(\upsilon_j \text{-} \upsilon_{y_i})^{\intercal} A_2\, z_i}\right) \tag{5.2.4}$$

i.e.,rearranged *soft-max cross-entropy* where $A_k{=}A_{b_k}$ with the abuse of notation, and $\theta = \{\theta_f, \omega, \Upsilon\}$ (*i.e.*,CNN parameters, prototype vectors, label embeddings).

We learn attribute embeddings (*i.e.*,columns of A) as sub-task and can define such learning as a differentiable operation. Thus, our cross-batch zero-shot prediction loss, $\mathcal{L}_{ZS}$, is to achieve *learning to learn attribute embeddings for zero-shot prediction.* Intuitively, such a regularization should be useful in better generalization of our pooling operation to unseen classes since pseudo-attribute predictions are connected to prototypes and the local features. We combine this loss with the metric learning loss using λ mixing (*i.e.*,$(1\text{-}\lambda)\mathcal{L}_{DML} + \lambda\mathcal{L}_{ZS}$) and jointly optimize. Then, our final batch loss becomes:

$$\mathcal{L}(b;\theta) = (1 \text{ - } \lambda)\, \mathcal{L}_{DML}(b;\theta) + \lambda\, \mathcal{L}_{ZS}(b;\theta) \tag{5.2.5}$$

where $\mathcal{L}_{DML}(b;\theta)$ defined in Eq. (5.1.1) can be any pair- or proxy-based DML loss.

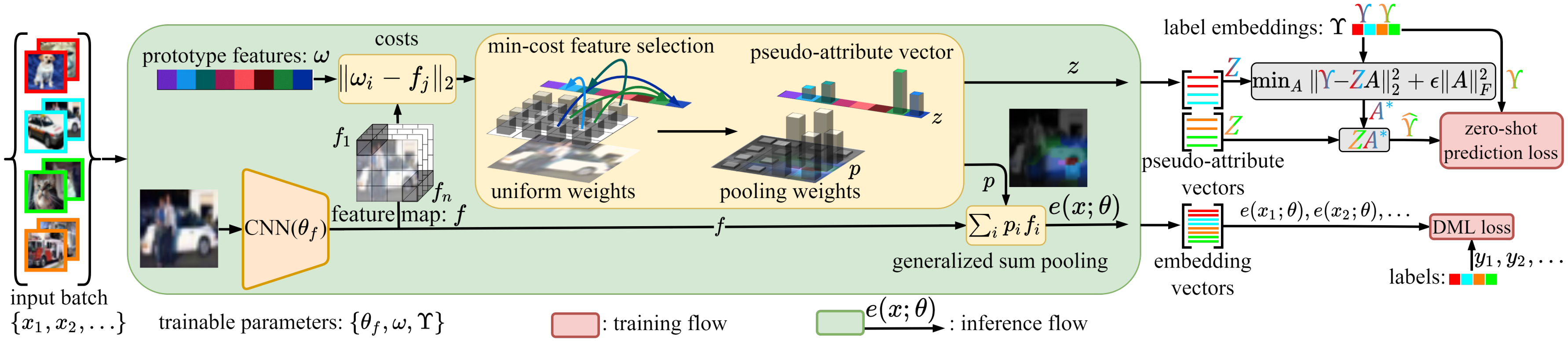


Figure 5.3: Sketch of the method.

5.2.4 Implementation Details

Embedding function. For the embedding function, $f(\cdot;\theta)$, we use ResNet20 [105] for Cifar [119] experiments, and ImageNet [107] pretrained BN-Inception [108] for the rest. We exploit architectures until the output before the global average pooling layer. We add a per-pixel linear transform (*i.e.*,1x1 convolution), to the output to obtain the local embedding vectors of size 128.

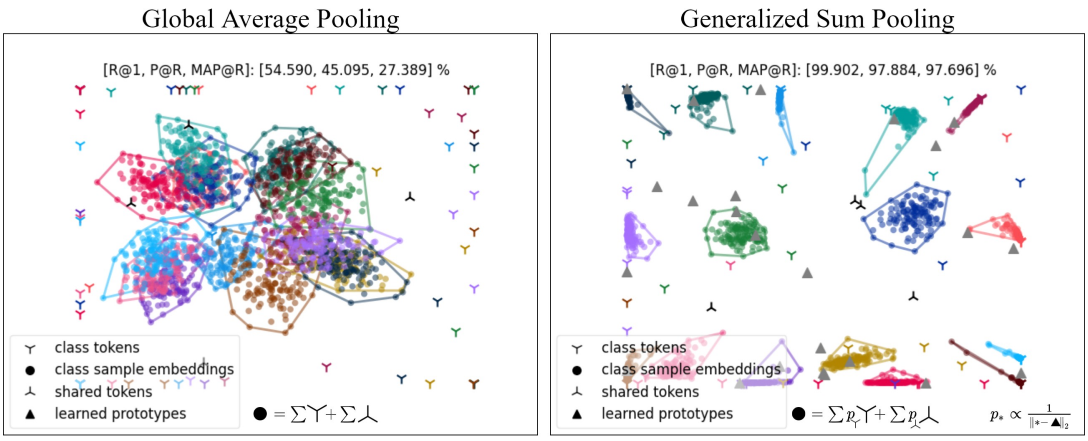
Pooling layer. For baseline methods, we use global average pooling. For our method, we perform parameter search and set the hyperparameters accordingly. Specifically, we use 64- or 128-many prototypes depending on the dataset. We use ε=0.5 for proxy-based losses and ε=5.0 for non-proxy losses. For the rest, we set μ=0.3, ϵ=0.05, λ=0.1 and we iterate until k=100 in Proposition 5.2.1. The embedding vectors upon global pooling are $l2$ normalized to have unit norm.

5.3 Experiments

We start our empirical study with a synthetic study validating the role of GAP in learning and the impact of GSP on the feature geometry. We further examine the effectiveness of our generalized sum pooling in metric learning for various models and datasets. We further perform ablation studies for the implications of our formulation as well as effects of the hyperparameters.

5.3.1 Synthetic Study

We design a synthetic empirical study to evaluate GSP in a fully controlled manner. We consider 16-class problem such that classes are defined over trainable tokens. In this setting, tokens correspond to semantic entities but we choose to give a specific working to emphasize that they are trained as part of the learning. Each class is defined with 4 distinct tokens and there are also 4 background tokens shared by all classes. For example, a *"car"* class would have tokens like *"tire"* and *"window"* as well as background tokens of *"tree"* and *"road"*.



(a) (b)

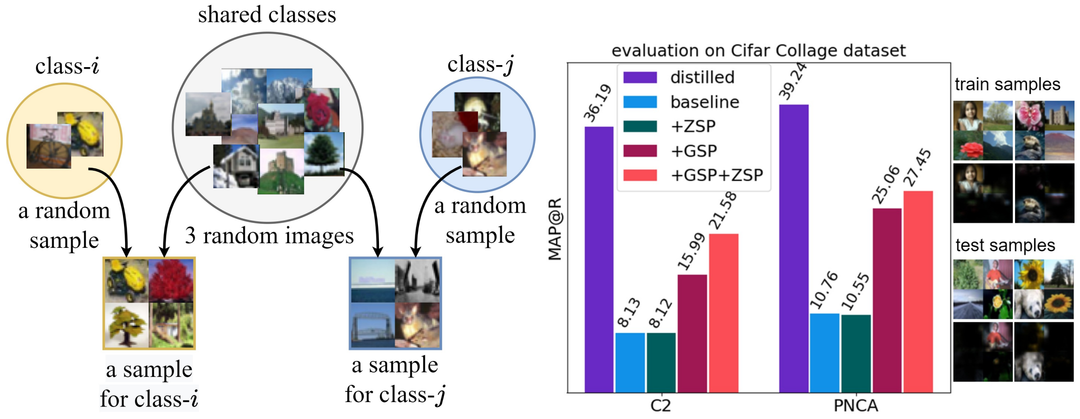
Figure 5.4: GAP (a) vs GSP (b) in aggregating features, where tokens denote learned embedding vectors and samples are obtained by aggregating them.

We sample class representations from both class specific and background tokens according to a mixing ratio $\tilde{\mu} \sim \mathcal{N}(0.5, 0.1)$. We sample a total of 50 tokens and such a 50-many feature collection will correspond to a training sample ($i.e.$,we are mimicking CNN's output with trainable tokens). For instance, given class tokens for class-c, $\nu^{(c)} = \{\nu_1^{(c)}, \nu_2^{(c)}, \nu_3^{(c)}, \nu_4^{(c)}\}$ and shared tokens, $\nu^{(b)} = \{\nu_1^{(b)}, \nu_2^{(b)}, \nu_3^{(b)}, \nu_4^{(b)}\}$; we first sample $\mu = 0.4$ and then sample 20 tokens from $\nu^{(c)}$ with replacement, and 30 tokens from $\nu^{(b)}$, forming a feature collection for a class-c, $i.e.$,$f^{(c)} = \{\nu_3^{(c)}, \nu_1^{(c)}, \nu_1^{(c)}, \nu_3^{(c)}, \ldots, \nu_4^{(b)}, \nu_3^{(b)}, \nu_4^{(b)}, \nu_1^{(b)}, \ldots\}$ We then obtain global representations using GAP and GSP.

We do not apply $l2$ normalization on the global representations. We also constrain the range of the token vectors to be in between $[-0.3, 0.3]$ to bound the magnitude of the learned vectors. We use default Adam optimizer with 10^{-4} learning rate and perform early stopping with 30 epoch patience by monitoring MAP@R. In each batch, we use 4 samples from 16 classes.

We visualize the geometry of the embedding space in Fig. 5.4. With GAP, we observe overlapping class convex hulls hence classes are not well discriminated. In other other hand, GSP gives well separated class convex hulls further validation that it learns to ignore background tokens.

5.3.2 Cifar Collage Experiments



(a) (b)

Figure 5.5: Visualizations for Cifar Collage dataset experiments: (a) Illustration of a sample generation for Cifar Collage dataset. (b) Evaluation on Cifar Collage dataset and on the right; sample train and test images with their attention maps in terms of pooling weights. *Distilled* denotes baseline performance on non-collage dataset (*i.e.*,excluding the shared classes).

We further extend the synthetic study (§ 5.3.1) to image domain. We consider the 20 *super-classes* of Cifar100 dataset [119] where each has 5 sub-classes. For each super-class, we split the sub-classes for train (2), validation (1), and test (2). We consider 4 super-classes as the shared classes and compose 4x4-stitched collage images for the rest 16 classes. In particular, we sample an image from a class and then sample 3 images from shared classes. We illustrate a sample formation process in § 5.3.2.

We should note that the classes exploited in training, validation and test are disjoint. For instance, if a *tree* class is used as a shared class in training, then that *tree* class does not exist in validation or test set as a shared feature. Namely, in our problem setting, both the background and the foreground classes are disjoint across training, validation and test sets. Such a setting is useful to analyze zero-shot transfer capability of our method.

We use ResNet20 (*i.e.*,3 stages, 3 blocks) backbone pretrained on Cifar100 classification task and follow the implementation explained in § 5.2.4. We use $l2$

normalization on global representations. We use default Adam optimizer with initial 0.001 learning rate. We use *reduce on plateau* with 0.5 decay factor and 5 epochs patience. For GSP, we set $m = 64, \mu = 0.2, \varepsilon = 10, \lambda = 0.5$. We use 4 samples from 16 classes in a batch.

We provide the evaluation results in § 5.3.2. GSP and the proposed zero shot loss effectively increase MAP@R. We also provide sample train and test images to showcase that our pooling can transfer well to unseen domain.

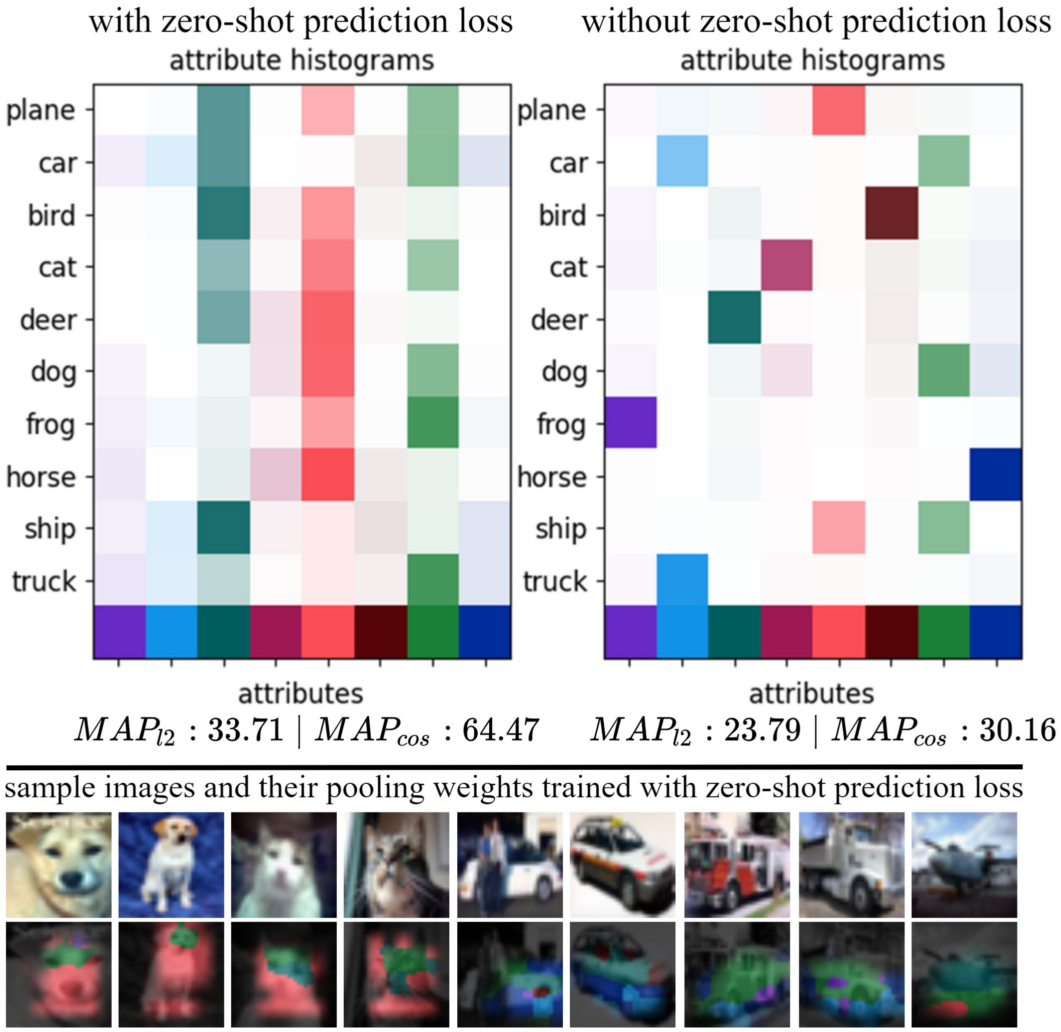


Figure 5.6: Comparing the distributions of the learned 8 prototypes across classes of Cifar10 dataset with and without $\mathcal{L}_{ZS}$. Pooling weights are coloured according to the dominant prototype at that location.

5.3.3 Evaluation of Zero-shot Prediction Loss

We also evaluate the zero-shot prediction performance of the pseudo-attribute vectors. We train on Cifar10 [119] dataset with 8 prototypes using ProxyNCA++ [15] (PNCA) loss with and without $\mathcal{L}_{ZS}$. We then use test set to compute pseudo-attribute histograms for each class. Namely, we aggregate the marginal transport plans of each sample in a class to obtain the histogram.

We visualize the class-attribute vectors in Fig. 5.6. We observe transferable representations with $\mathcal{L}_{ZS}$ and we visually show in Fig. 5.6 that the semantic entities represented by the prototypes transfer across classes.

We quantitatively evaluate such behavior by randomly splitting the classes into half and apply cross-batch zero-shot prediction explained in § 5.2.3. For each class, we compute the mean embedding vector (*i.e.*,we average embedding vectors of the samples of a class). Our aim is to fit a linear predictor to map attribute vectors to the mean embeddings. Namely, we fit A in (P3) for one subset and use it to predict the class embeddings for the other set.

Specifically, we fit a linear predictor using 5 classes and then use that transformation to map the other 5 classes to their mean embeddings. We then compute pairwise distance between the predicted means and the true means. We then evaluate the nearest neighbour classification performance. We use both $l2$ distance and *cosine* distance while computing the pairwise distances. We repeat the experiment 1000 times with different class splits. We observe in Fig. 5.6 that zero-shot performance of the prototypes learned with $\mathcal{L}_{ZS}$ is substantially superior.

5.3.4 Deep Metric Learning Experiments

5.3.4.1 Setup

In order to minimize the confounding of factors other than our proposed method, we keep the comparisons as fair as possible following the suggestions of recent work explicitly studying the fair evaluation strategies for metric learning [6, 7, 110]. Specifically, we mostly follow the procedures proposed in [6] to provide fair and unbiased evaluation of our method as well as comparisons with the other methods. We provide full detail of our experimental setup for the sake of complete transparency and reproducibility.

Datasets

We perform our experiments on 4 widely-used benchmark datasets: Stanford Online Products (SOP) [32], In-shop [111], Cars196 [112] and, CUB-200-2011 (CUB) [113].

SOP has 22,634 classes with 120,053 product images. The first 11,318 classes (59,551 images) are split for training and the other 11,316 (60,502 images) classes are used for testing.

In-shop has 7,986 classes with 72,712 images. We use 3,997 classes with 25,882 images as the training set. For the evaluation, we use 14,218 images of 3,985 classes as the query and 12,612 images of 3,985 classes as the gallery set.

Cars196 contains 196 classes with 16,185 images. The first 98 classes (8,054 images) are used for training and remaining 98 classes (8,131 images) are reserved for testing.

CUB-200-2011 dataset consists of 200 classes with 11,788 images. The first 100 classes (5,864 images) are split for training, the rest of 100 classes (5,924 images) are used for testing.

Data augmentation follows [6]. During training, we resize each image so that its shorter side has length 256, then make a random crop between 40 and 256, and aspect ratio between $^3/_4$ and $^4/_3$. We resize the resultant image to 227x227 and apply random horizontal flip with 50% probability. During evaluation, images are resized to 256 and then center cropped to 227x227.

Training Splits

Fair evaluation. We split datasets into disjoint training, validation and test sets according to [6]. In particular, we partition 50%/50% for training and test, and further split training data to 4 partitions where 4 models are to be trained by exploiting 1/4 as validation while training on 3/4.

Conventional evaluation. Following relatively *old-fashioned* conventional evaluation [32], we use the whole train split of the dataset for training and we use the test split for evaluation as well as monitoring the training for early stopping.

Hyperparameter tuning. For the additional experiments related to the effect of hyperparameters, we split training set into 5 splits and train a single model on the 4/5 of the set while using 1/5 for the validation.

Evaluation Metrics

We consider precision at 1 (P@1) and mean average precision (MAP@R) at R where R is defined for each query and is the total number of true references as the query. Among those, MAP@R performance metric is shown to better reflect the geometry of the embedding space and to be less noisy as the evaluation metric [6]. Thus, we use MAP@R to monitor training in our experiments except for conventional evaluation setting where we monitor P@1. We explain the metrics in § 1.2.

Training Procedure

Fair evaluation. We use Adam [114] optimizer with constant 10^{-5} learning rate, 10^{-4} weight decay, and default moment parameters, β_1=.9 and β_2=.99. We use batch size of 32 (4 samples per class). We evaluate validation MAP@R for every 100 steps of training in CUB and Cars196, for 1000 steps in SOP and In-shop. We stop training if no improvement is observed for 15 steps in CUB and Cars196, and 10 steps in SOP and In-shop. We recover the parameters with the best validation performance. Following [6], we train 4 models for each 3/4 partition of the train set. Each model is trained 3 times. Hence, we have $3^4 = 81$ many realizations of 4-model collections. We present the average performance of such 81 models as well as standard deviation (*std*) of 81 evaluations.

Conventional evaluation. We use Adam [114] optimizer with default moment parameters, β_1=.9 and β_2=.99. Following recent works [37], we use *reduce on plateau* learning rate scheduler with patience 4. The initial learning rate is 10^{-5} for CUB, and 10^{-4} for Cars, SOP and In-shop. We use 10^{-4} weight decay for BNInception backbone and $4\,10^{-4}$ wight decay for ResNet50 backbone. We use batch size of 128 (4 samples per class) for BNInception backbone and 112 (4 samples per class) for ResNet backbone (following [7]). We evaluate validation P@1 for every 25 steps of training in CUB and Cars196, for 250 steps in SOP and In-shop. We stop training if no improvement is observed for 15 steps in CUB and Cars196, and 10 steps in SOP and In-shop. We recover the parameters with the best validation performance. We repeat each experiment 3 times and report the best result.

Hyperparameter tuning. We use Adam [114] optimizer with constant 10^{-5} learning rate, 10^{-4} weight decay, and default moment parameters, β_1=.9 and β_2=.99. We use batch size of 32 (4 samples per class). We evaluate validation MAP@R for every 100 steps of training in CUB and Cars196, for 1000 steps in SOP and In-shop. We stop training if no improvement is observed for 10 steps in CUB and Cars196, and 7 steps in SOP and In-shop. We recover the parameters with the best validation performance. We train a single model on the $^4/_5$ of the training set while using $^1/_5$ for the validation. We repeat each experiment 3 times and report the averaged result.

Embedding Vectors

Fair evaluation. Embedding dimension is fixed to 128. During training and evaluation, the embedding vectors are $l2$ normalized. We follow the evaluation method proposed in [6] and produce two results: *i*) Average performance (128 dimensional) of 4-fold models and *ii*) Ensemble performance (concatenated 512 dimensional) of 4-fold models where the embedding vector is obtained by concatenated 128D vectors of the individual models before retrieval.

Conventional evaluation. Embedding dimension is 512 in BNInception experiments. For ResNet50, we both evaluate our method with 128 and 512 dimensional embeddings.

Hyperparameter tuning. Embedding dimension is fixed to 128.

Baselines with GSP

We evaluate our method with *C1+XBM+GSP*: Cross-batch memory (XBM) [14] with original Contrastive loss (C1) [29], *C2+GSP*: Contrastive loss with positive margin [9], *MS+GSP*: Multi-similarity (MS) loss [33], *Triplet+GSP*: Triplet loss [30], *PNCA+GSP*: ProxyNCA++ loss [15], *PAnchor+GSP*: ProxyAnchor loss [37].

Hyperparameters

For the hyperparameter selection, we exploit the recent work [6] that has performed parameter search via Bayesian optimization on variety of losses. We further experiment the suggested parameters from the original papers and official implementations. We pick the best performing parameters. We perform no further parameter tuning for the baseline methods' parameters when applied with our method to purely examine the effectiveness of our method.

C1: We adopted XBM's official implementation for fair comparison. We use 0.5 margin for all datasets.

C2: C2 has two parameters, (m^+, m^-): positive margin, m^+, and negative margin. We set (m^+, m^-) to $(0, 0.3841), (0.2652, 0.5409), (0.2858, 0.5130), (0.2858, 0.5130)$ for CUB, Cars196, In-shop and SOP, respectively.

Triplet: We set its margin to 0.0961, 0.1190, 0.0451, 0.0451 for CUB, Cars196, In-shop and SOP, respectively.

MS: MS has three parameters (α, β, λ). We set (α, β, λ) to $(2, 40, 0.5)$,

$(14.35, 75.83, 0.66)$, $(8.49, 57.38, 0.41)$, $(2, 40, 0.5)$ for CUB, Cars196, In-shop and SOP, respectively.

ProxyAnchor: We set its two paremeters (δ, α) to $(0.1, 32)$ for all datasets. We use 1 sample per class in batch setting (*i.e.*,32 classes with 1 samples per batch), we perform 1 epoch warm-up training of the embedding layer, and we apply learning rate multiplier of 100 for the proxies during training. For SOP dataset, we use $5\,10^{-6}$ learning rate.

ProxyNCA++: We set its temperature parameter to 0.11 for all datasets. We use 1 sample per class in batch setting (*i.e.*,32 classes with 1 samples per batch), we perform 1 epoch warm-up training of the embedding layer, and we apply learning rate multiplier of 100 for the proxies during training.

XBM: We evaluate XBM with C1 since in the original paper, contrastive loss is reported to be the best performing baseline with XBM. We set the memory size of XBM according to the dataset. For CUB and Cars196, we use memory size of 25 batches. For In-shop, we use 400 batches and for SOP we use 1400 batches. We perform 1K steps of training with the baseline loss prior to integrate XBM loss in order to ensure XBM's *slow drift* assumption.

GSP: For the hyperparameters of our method, we perform parameters search, details of which are provided in § 5.3.4.5. We use 64-many prototypes in CUB and Cars, and 128-many prototypes in SOP and In-shop. We use ε=0.5 for proxy-based losses and ε=5.0 for non-proxy losses. For the rest, we set μ=0.3, ϵ=0.05, and we iterate until k=100 in Proposition 4.1. For zero-shot prediction loss coefficient (*i.e.*,$(1\text{-}\lambda)\mathcal{L}_{DML} + \lambda\mathcal{L}_{ZS}$), we set λ=0.1.

5.3.4.2 Results

Fair Evaluation

We compare GSP against direct application of GAP with 6 DML methods in 4 datasets. The results are provided in Tab. 5.1 and summarized in Fig. 5.7. We observe consistent improvements upon direct application of GAP in all datasets.

Table 5.1: Comparison with the existing methods for the retrieval task on SOP, InShop, CUB, Cars.

Dataset →	**SOP**				**In-shop**				**CUB**				**Cars196**			
Dim. →	512D		128D		512D		128D		512D		128D		512D		128D	
Method↓	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**
C1	69.29$_{\mp0.11}$	40.40$_{\mp0.15}$	65.15$_{\mp0.10}$	36.50$_{\mp0.11}$	80.11$_{\mp0.19}$	50.32$_{\mp0.14}$	75.83$_{\mp0.14}$	46.42$_{\mp0.13}$	63.32$_{\mp0.57}$	23.49$_{\mp0.31}$	56.34$_{\mp0.35}$	19.37$_{\mp0.29}$	78.01$_{\mp0.38}$	22.87$_{\mp0.33}$	65.61$_{\mp0.59}$	16.06$_{\mp0.14}$
+XBM	76.54$_{\mp0.32}$	48.58$_{\mp0.47}$	73.22$_{\mp0.48}$	44.55$_{\mp0.57}$	87.76$_{\mp0.26}$	57.53$_{\mp0.41}$	85.26$_{\mp0.37}$	54.40$_{\mp0.45}$	65.56$_{\mp0.48}$	25.65$_{\mp0.24}$	57.48$_{\mp0.41}$	20.27$_{\mp0.19}$	**83.55**$_{\mp0.35}$	27.53$_{\mp0.22}$	72.17$_{\mp0.30}$	18.98$_{\mp0.17}$
+XBM+GSP	**77.88**$_{\mp0.18}$	**50.65**$_{\mp0.28}$	**74.84**$_{\mp0.19}$	**46.69**$_{\mp0.28}$	**88.33**$_{\mp0.19}$	**58.55**$_{\mp0.29}$	**85.95**$_{\mp0.21}$	**55.30**$_{\mp0.21}$	**67.00**$_{\mp0.49}$	**26.05**$_{\mp0.15}$	**58.89**$_{\mp0.49}$	**20.60**$_{\mp0.16}$	83.31$_{\mp0.22}$	**27.88**$_{\mp0.23}$	**73.04**$_{\mp0.39}$	**19.26**$_{\mp0.19}$
+CCP	78.49$_{\mp0.08}$	51.22$_{\mp0.11}$	75.48$_{\mp0.07}$	47.30$_{\mp0.09}$	88.75$_{\mp0.13}$	60.23$_{\mp0.15}$	85.70$_{\mp0.11}$	55.87$_{\mp0.16}$	67.45$_{\mp0.42}$	26.35$_{\mp0.30}$	59.01$_{\mp0.41}$	20.76$_{\mp0.26}$	83.74$_{\mp0.32}$	27.29$_{\mp0.28}$	72.13$_{\mp0.31}$	18.44$_{\mp0.12}$
+CCP+GSP	**78.64**$_{\mp0.09}$	**51.59**$_{\mp0.11}$	**75.67**$_{\mp0.04}$	**47.63**$_{\mp0.06}$	**88.84**$_{\mp0.11}$	**60.43**$_{\mp0.12}$	**86.12**$_{\mp0.12}$	**56.27**$_{\mp0.12}$	**68.03**$_{\mp0.47}$	**26.98**$_{\mp0.38}$	**59.42**$_{\mp0.35}$	**21.19**$_{\mp0.27}$	**83.95**$_{\mp0.26}$	**27.93**$_{\mp0.27}$	**72.46**$_{\mp0.36}$	**18.98**$_{\mp0.18}$
C2	74.20$_{\mp0.23}$	45.85$_{\mp0.31}$	70.54$_{\mp0.19}$	41.79$_{\mp0.26}$	86.47$_{\mp0.15}$	59.07$_{\mp0.21}$	83.42$_{\mp0.12}$	55.38$_{\mp0.13}$	67.35$_{\mp0.50}$	25.95$_{\mp0.21}$	58.87$_{\mp0.36}$	20.58$_{\mp0.13}$	80.96$_{\mp0.48}$	24.38$_{\mp0.58}$	69.55$_{\mp0.42}$	17.02$_{\mp0.31}$
+GSP	**74.91**$_{\mp0.12}$	**46.81**$_{\mp0.17}$	**71.43**$_{\mp0.11}$	**42.84**$_{\mp0.14}$	**86.90**$_{\mp0.17}$	**60.01**$_{\mp0.29}$	**83.57**$_{\mp0.18}$	**55.94**$_{\mp0.17}$	**68.85**$_{\mp0.41}$	**27.12**$_{\mp0.27}$	**60.42**$_{\mp0.36}$	**21.53**$_{\mp0.24}$	**82.83**$_{\mp0.27}$	**26.25**$_{\mp0.34}$	**71.40**$_{\mp0.27}$	**18.31**$_{\mp0.22}$
MS	72.81$_{\mp0.14}$	44.19$_{\mp0.21}$	69.09$_{\mp0.10}$	40.34$_{\mp0.16}$	87.01$_{\mp0.20}$	58.79$_{\mp0.37}$	83.87$_{\mp0.21}$	54.85$_{\mp0.34}$	65.43$_{\mp0.46}$	24.95$_{\mp0.15}$	**57.57**$_{\mp0.27}$	20.13$_{\mp0.12}$	83.73$_{\mp0.34}$	27.16$_{\mp0.43}$	72.54$_{\mp0.43}$	18.73$_{\mp0.31}$
+GSP	**73.05**$_{\mp0.11}$	**44.72**$_{\mp0.17}$	**69.44**$_{\mp0.15}$	**40.87**$_{\mp0.19}$	**88.28**$_{\mp0.21}$	**60.49**$_{\mp0.24}$	**85.28**$_{\mp0.19}$	**56.62**$_{\mp0.26}$	**65.50**$_{\mp0.33}$	**25.09**$_{\mp0.21}$	57.39$_{\mp0.15}$	**20.34**$_{\mp0.22}$	**84.27**$_{\mp0.35}$	**28.58**$_{\mp0.40}$	**73.74**$_{\mp0.32}$	**19.91**$_{\mp0.31}$
Triplet	74.54$_{\mp0.24}$	45.88$_{\mp0.3}$	69.41$_{\mp0.38}$	40.01$_{\mp0.39}$	85.99$_{\mp0.36}$	59.67$_{\mp0.46}$	81.75$_{\mp0.38}$	54.25$_{\mp0.45}$	64.11$_{\mp0.66}$	23.65$_{\mp0.4}$	55.62$_{\mp0.46}$	18.54$_{\mp0.31}$	77.58$_{\mp0.60}$	22.67$_{\mp0.58}$	64.61$_{\mp0.59}$	15.74$_{\mp0.34}$
+GSP	**75.59**$_{\mp0.23}$	**47.35**$_{\mp0.32}$	**70.65**$_{\mp0.20}$	**41.38**$_{\mp0.22}$	**86.75**$_{\mp0.27}$	**60.85**$_{\mp0.47}$	**82.74**$_{\mp0.33}$	**55.54**$_{\mp0.46}$	**66.09**$_{\mp0.52}$	**24.80**$_{\mp0.33}$	**57.12**$_{\mp0.42}$	**19.38**$_{\mp0.25}$	**78.93**$_{\mp0.30}$	**23.44**$_{\mp0.29}$	**65.81**$_{\mp0.35}$	**16.14**$_{\mp0.21}$
PNCA	75.18$_{\mp0.15}$	47.11$_{\mp0.16}$	72.15$_{\mp0.06}$	43.57$_{\mp0.08}$	**87.38**$_{\mp0.12}$	57.58$_{\mp0.16}$	85.04$_{\mp0.08}$	**54.72**$_{\mp0.06}$	65.74$_{\mp0.51}$	25.27$_{\mp0.23}$	58.19$_{\mp0.36}$	20.63$_{\mp0.20}$	82.33$_{\mp0.25}$	26.21$_{\mp0.22}$	70.75$_{\mp0.18}$	18.61$_{\mp0.08}$
+GSP	**75.68**$_{\mp0.11}$	**47.74**$_{\mp0.14}$	**72.37**$_{\mp0.06}$	**43.95**$_{\mp0.06}$	87.35$_{\mp0.10}$	**57.65**$_{\mp0.12}$	**85.13**$_{\mp0.1}$	54.68$_{\mp0.08}$	**65.80**$_{\mp0.38}$	**25.48**$_{\mp0.25}$	**58.20**$_{\mp0.22}$	**20.75**$_{\mp0.19}$	**82.70**$_{\mp0.27}$	**26.93**$_{\mp0.18}$	**71.55**$_{\mp0.32}$	**19.20**$_{\mp0.17}$
PAnchor	76.48$_{\mp0.19}$	48.08$_{\mp0.26}$	73.50$_{\mp0.14}$	44.33$_{\mp0.20}$	88.02$_{\mp0.21}$	58.02$_{\mp0.25}$	85.83$_{\mp0.18}$	54.98$_{\mp0.22}$	68.04$_{\mp0.41}$	26.20$_{\mp0.21}$	59.91$_{\mp0.34}$	20.94$_{\mp0.15}$	85.26$_{\mp0.31}$	27.14$_{\mp0.20}$	75.08$_{\mp0.23}$	19.15$_{\mp0.13}$
+GSP	**77.13**$_{\mp0.16}$	**49.05**$_{\mp0.22}$	**74.07**$_{\mp0.13}$	**45.07**$_{\mp0.17}$	**88.10**$_{\mp0.11}$	**58.44**$_{\mp0.14}$	**85.97**$_{\mp0.06}$	**55.34**$_{\mp0.13}$	**68.40**$_{\mp0.45}$	**26.59**$_{\mp0.25}$	**60.80**$_{\mp0.31}$	**21.44**$_{\mp0.17}$	**86.46**$_{\mp0.39}$	**28.43**$_{\mp0.33}$	**75.88**$_{\mp0.25}$	**19.90**$_{\mp0.20}$

We achieve more improvement in MAP@R metric than R@1 which is shown to be a noisy measure for DML evaluation [6]. On the average, we consistently improve the baselines ≈1% points in MAP@R except for a single case (In-shop: PNCA) where we perform on par. We improve state-of-the-art XBM method up to 2% points, which is a good evidence that application of GSP is not limited to loss terms but can be combined with different DML approaches. Moreover, GSP also improves our CCP method. Again, a supporting result for wide applicability of GSP method.

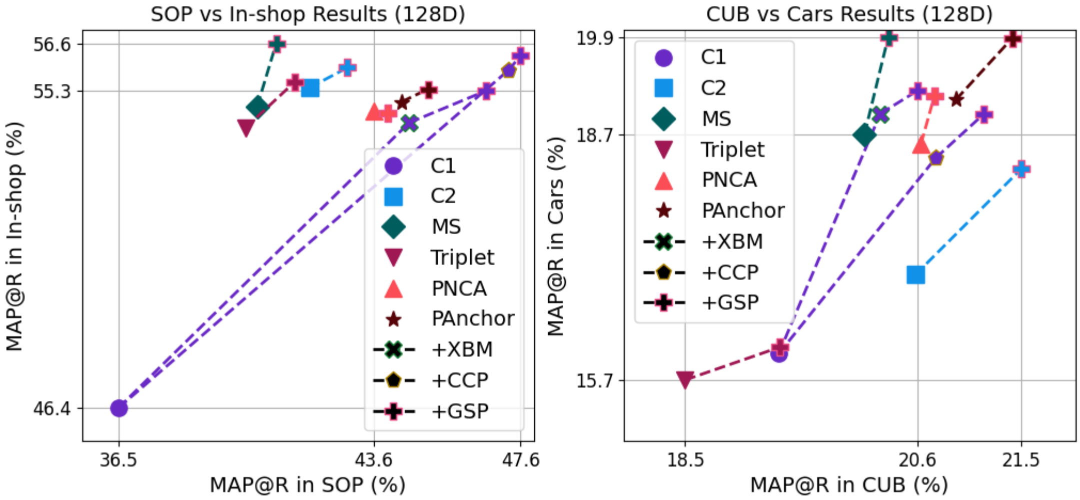


Figure 5.7: Summary of relative improvements recorded in Tab. 5.1.

Conventional Evaluation

We additionally follow the relatively old-fashioned conventional procedure [32] for the evaluation of our method. We use BN-Inception [108] and ResNet50 [105] architectures as the backbones. We obtain 128D (ResNet50) and 512D (BN-Inception and ResNet50) embeddings through linear transformation after global pooling layer. Aligned with the recent approaches [13–15, 37], we use global max pooling as well as global average pooling. The rest of the settings are disclosed in § 5.3.4.1.

We evaluate our method with XBM. We provide R@1 results in Tab. 5.2 for the comparison with SOTA. In our evaluations, we also provide MAP@R scores in parenthesis under R@1 scores. We also provide baseline XBM evaluation in our framework. The results are mostly consistent with the ones reported in the original paper [14] except for CUB and Cars datasets. In XBM [14], the authors use proxy-based trainable memory for CUB and Cars datasets. On the other hand, we use the official implementation provided by the authors, which does not include such proxy-based extensions.

We observe that our method improves XBM and XBM+GSP reaches SOTA performance in large scale datasets. With that being said, the improvement margins are less substantial than the ones in *fair evaluation*. Such a result is expected since training is terminated by *early-stopping* which is a common practice to regularize the generalization of training [16, 17]. In *conventional evaluation*, early-stopping is achieved by monitoring the test data performance, enabling good generalization to test data. Therefore, observing less improvement in generalization with GSP is something we expect owing to generalization boost that test data based early-stopping already provides.

Table 5.2: Comparison with the existing methods for the retrieval task in conventional experimental settings with BN-Inception and ResNet50 backbones where superscripts denote embedding size. Red: the best. Blue: the second best. Bold: previous SOTA. †*Results obtained from* [1].

(a)

Backbone →	**BN-Inception-512D**			
Dataset →	**CUB**	**Cars196**	**SOP**	**In-shop**
Method ↓	**R@1**	**R@1**	**R@1**	**R@1**
C1+XBM [14]	65.80	82.00	**79.50**	89.90
ProxyAnchor [37]	68.40	86.10	79.10	**91.50**
DiVA [23]	66.80	84.10	78.10	-
ProxyFewer [59]	66.60	85.50	78.00	-
Margin+S2SD [24]	**68.50**	**87.30**	79.30	-
C1+XBM	64.32 (23.59)	77.63 (21.67)	79.29 (52.59)	90.16 (61.39)
C1+XBM+GSP	64.99 (25.35)	79.07 (36.11)	**79.59** (52.70)	**90.92** (63.25)

(b)

Backbone →	**ResNet50**			
Dataset →	**CUB**	**Cars196**	**SOP**	**In-shop**
Method ↓	**R@1**	**R@1**	**R@1**	**R@1**
C1+XBM^{128} [14]	-	-	80.60	91.30
ProxyAnchor^{512} [37]	**69.70**	87.70	80.00†	92.10†
DiVA^{512} [23]	69.20	87.60	79.60	-
ProxyNCA++^{512} [15]	66.30	85.40	80.20	88.60
Margin+S2SD^{512} [24]	69.00	**89.50**	81.20	-
LIBC^{512} [1]	70.30	88.10	**81.40**	**92.80**
MS+Metrix^{512} [13]	**71.40**	**89.60**	81.00	**92.20**
PAnchor+DIML^{128} [65]	66.46 (25.58)	86.13 (28.11)	79.22 (43.04)	-
C1+XBM^{128}	62.65 (23.70)	78.26 (22.95)	79.92 (53.45)	91.06 (62.64)
C1+XBM+GSP^{128}	63.44 (24.04)	79.60 (23.87)	80.24 (53.81)	91.03 (62.74)
C1+XBM^{512}	66.68 (25.38)	82.83 (25.34)	**81.44** (55.66)	91.56 (64.00)
C1+XBM+GSP^{512}	66.63 (25.51)	82.60 (25.76)	**81.54** (55.91)	91.75 (64.43)

We also observe that in a few cases, the R@1 performance of GSP is slightly worse than the baseline. Nevertheless, once we compare the MAP@R performances, GSP consistently brings improvement with no exception. We should recapitulate that R@1 is a myopic metric to assess the quality of the embedding space geometry [6] and hence, pushing R@1 does not necessarily reflect the true order of the improvements that the methods bring. As we observe from MAP@R comparisons, the methods sharing similar R@1 (*i.e.*,P@1) performances can differ in MAP@R performance relatively more significantly. In that manner, we firmly believe that comparing MAP@R performances instead of R@1 technically sounds more in showing the improvements of our method.

5.3.4.3 Ablations

Effect of $\mathcal{L}_{ZS}$

Table 5.3: Effects of the two components: Zero-shot prediction loss (ZSP) and Generalized Sum Pooling (GSP)

Base Method: C2		**MAP@R**							
Component		**SOP**		**In-shop**		**CUB**		**Cars196**	
ZSP	GSP	512D	128D	512D	128D	512D	128D	512D	128D
		45.85	41.79	59.07	55.38	25.95	20.58	24.38	17.02
	✓	46.78	42.66	59.46	55.50	26.25	20.85	25.54	17.88
✓		46.60	42.55	59.38	55.43	26.49	21.08	25.54	17.67
✓	✓	**46.81**	**42.84**	**60.01**	**55.94**	**27.12**	**21.52**	**26.25**	**18.31**

We empirically show the effect of $\mathcal{L}_{ZS}$ on learned representations in § 5.3.1. We further examine the effect of $\mathcal{L}_{ZS}$ quantitatively by enabling/disabling it in 4 datasets. We also evaluate its effect without GSP by setting μ=1 where we use GAP with pseudo-attribute vectors. The results are summarized in Tab. 5.3 showing that both components improves the baseline and their combination brings the best improvement. We observe similar behavior in *Cifar Collage* experiment (Fig. 5.4-(b)) where the effect of $\mathcal{L}_{ZS}$ is more substantial.

Effect of μ

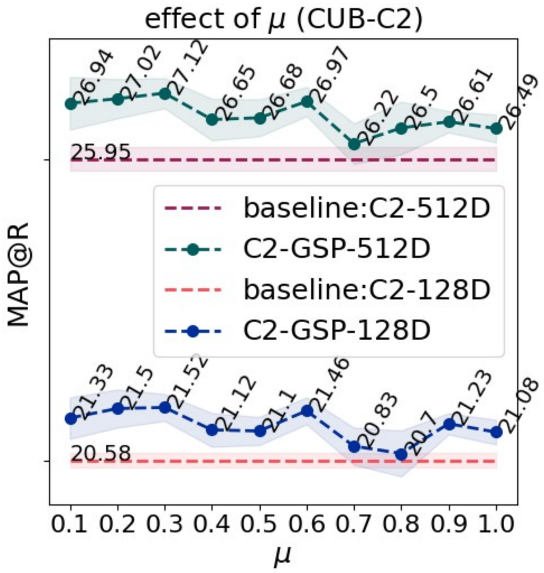


Figure 5.8: Effect of μ in CUB dataset with C2 loss. Shaded regions represent $\mp std$.

As we discuss in § 5.2.2, GSP is similar to top-k operator with an adaptive k thanks to entropy smoothing. We empirically validate such behavior in CUB dataset with C2 loss by sweeping μ parameter controlling top-k behavior. We plot the performances in Fig. 5.8. Relatively lower values of μ performs similarly. As we increase μ, the performance drops towards GAP due to possibly overestimating the foreground ratio.

5.3.4.4 Computational Analysis

Forward and backward computation of proposed GSP method can be implemented using only matrix-vector products. Moreover, having closed-form matrix-inversion-free expression for the loss gradient enables memory efficient back propagation since the output of each iteration must be stored otherwise.

We perform k iterations to obtain the pooling weights and at each iteration, we only perform matrix-vector products. In this sense, the back propagation can be achieved using *automatic-differentiation*. One problem with automatic differentiation is that the computation load increases with increasing k. On the other hand, with closed-form gradient expression, we do not have such issue and in fact we have constant back propagation complexity. Granted that the closed-form expression demands exact solution of the problem (*i.e.*,$k \to \infty$), forward computation puts a little computation overhead and is memory efficient since we discard the intermediate outputs. Moreover, our initial empirical study show that our problems typically converges for $k > 50$ and we observe similar performances with $k \geqslant 25$.

The choice of k is indeed problem dependent (*i.e.*,size of the feature map and number of prototypes). Thus, it is important to see the effect of k on computation load. We analyze the effect of k with automatic differentiation and with our closed-form gradient expression. We provide the related plots in Fig. 5.9. We observe that with closed-form gradients, our method puts a little computation overhead and increasing k has marginal effect. On the contrary, with automatic differentiation, the computational complexity substantially increases.

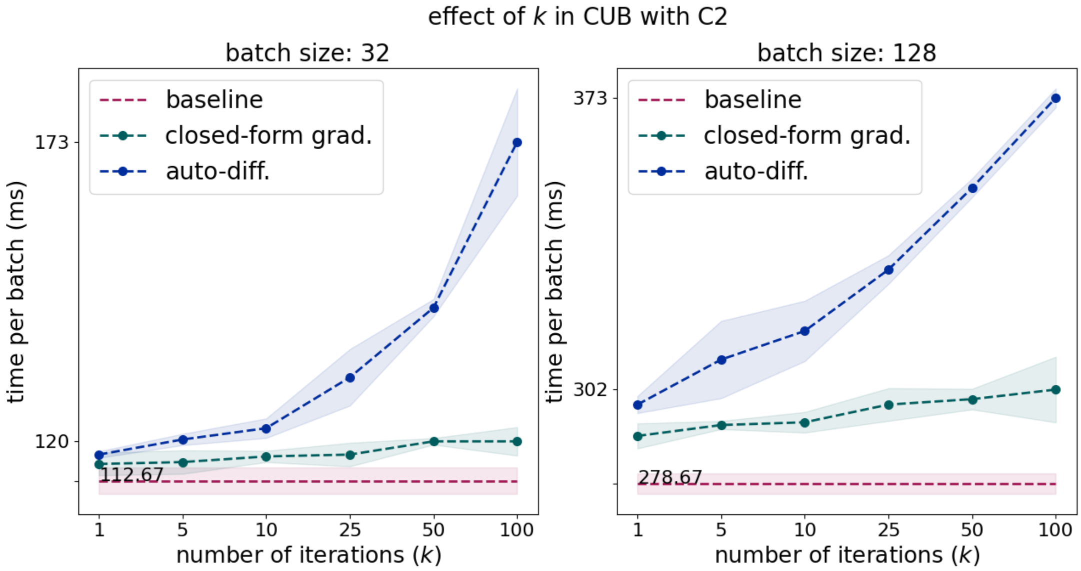


Figure 5.9: Comparing closed-form gradient with automatic differentiation through analyzing the effect of k on computation in CUB dataset with C2 loss. Shaded regions represent $\mp std$.

5.3.4.5 Hyperparameter Selection

We first perform a screening experiment to see the effect of the parameters. We design a 2-level fractional factorial (*i.e.*,a subset of the all possible combinations) experiment. We provide the results in Tab. 5.4. In our analysis, we find that *lower the better* for λ and μ. Thus, we set $\mu = 0.3$ and $\lambda = 0.1$. ε is observed to have the most effect and number of prototypes, m, seems to have no significant effect. Nevertheless, we jointly search for m and ε.

Table 5.4: Initial 2-level fractional factorial screening experiments for hyperparameter tuning (conducted in CUB).

Setting				**MAP@R**	
m	μ	ε	λ	C2	PNCA
16	0.3	0.5	0.1	40.63	40.59
16	0.7	0.5	0.5	40.41	40.34
128	0.3	0.5	0.5	40.22	40.35
128	0.7	0.5	0.1	40.07	40.85
16	0.3	20	0.5	36.11	40.51
16	0.7	20	0.1	39.11	39.88
128	0.3	20	0.1	39.61	39.12
128	0.7	20	0.5	35.36	39.92
Baseline				39.77	39.90

To this end, we perform grid search in CUB dataset with Contrastive (C2) and Proxy NCA++ (PNCA) losses. We provide the results in Fig. 5.10. We see that both losses have their best performance when $m = 64$. On the other hand, $\varepsilon = 5.0$ works better for C2 while $\varepsilon = 0.5$ works better for PNCA. We additionally perform a small experiment to see whether $\varepsilon = 0.5$ is the case for Proxy Anchor loss as well and observe that $\varepsilon = 0.5$ is a better choice over $\varepsilon = 5.0$. As the result of m-ε search, we set $\varepsilon = 5.0$ for non-proxy based losses and $\varepsilon = 0.5$ for proxy-based losses.

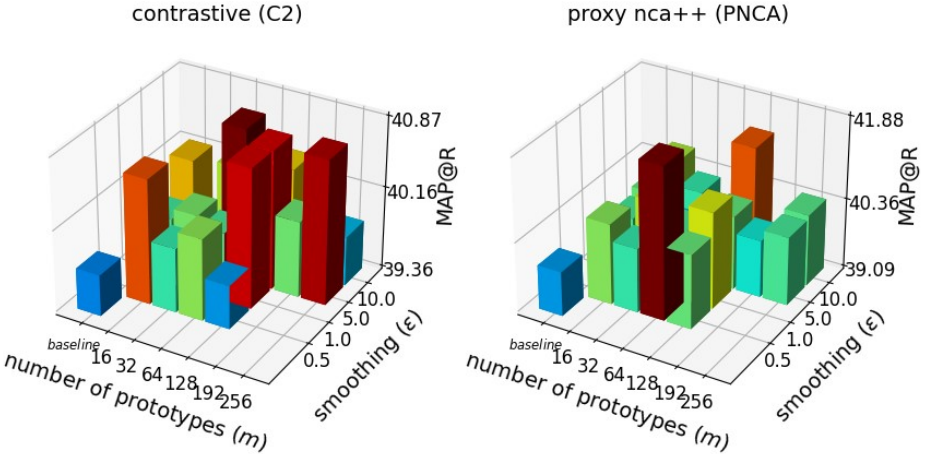


Figure 5.10: Searching $m-\varepsilon$ space in CUB dataset for the parameters m: number of prototoypes and ε: entropy smoothing coefficient. We fix $\mu = 0.3$ and $\lambda = 0.1$.

Fixing $\mu = 0.3, \lambda = 0.1, \varepsilon = 0.5$(or 5.0), we further experiment the effect of number of prototypes, m, in large datasets (*i.e.*,SOP and In-shop). We provide the corresponding performance plots in Fig. 5.11. Supporting our initial analysis, m seemingly does not have a significant effect once it is not small (*e.g.* $m \geqslant 64$). We observe that any choice of $m \geqslant 64$ provides performance improvement. With that being said, increasing m does not bring substantial improvement over relatively smaller values. Considering the results of the experiment, we set $m = 128$ for SOP and In-shop datasets since both C2 and PNCA losses perform slightly better with $m = 128$ than the other choices of m.

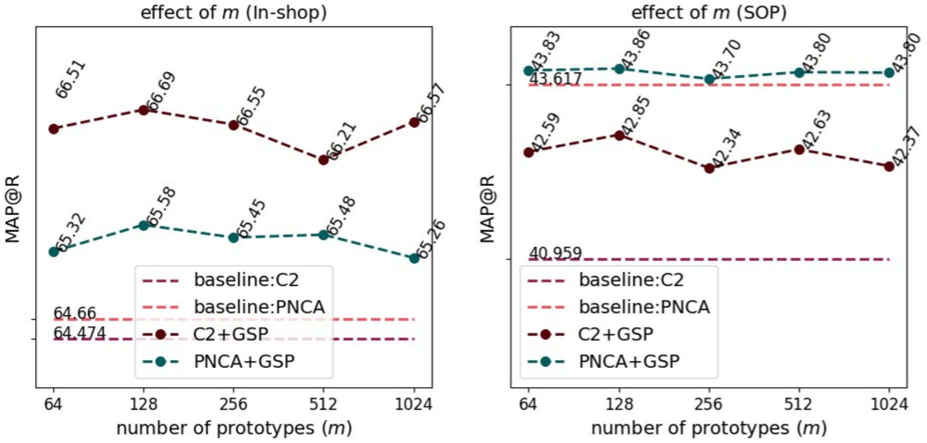
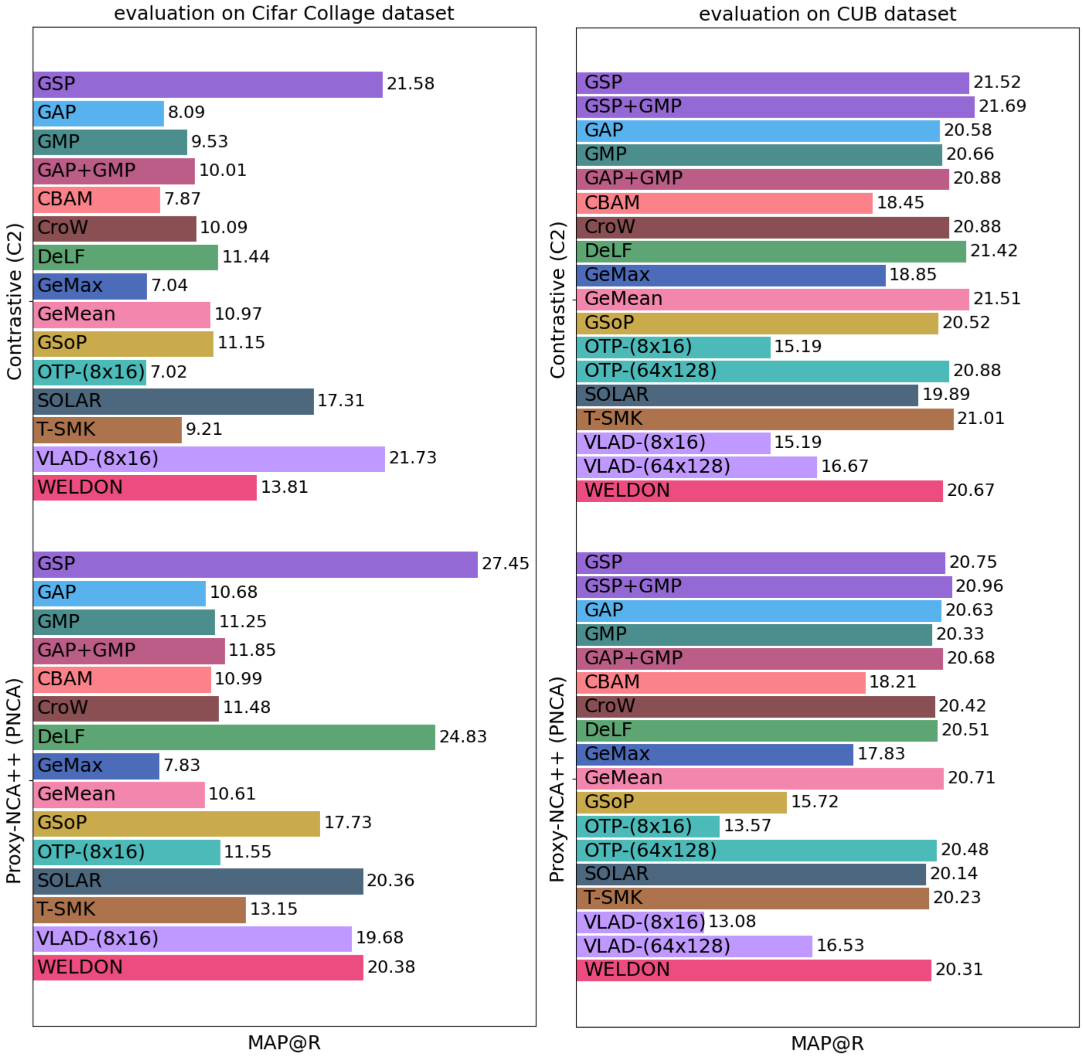


Figure 5.11: Effect of m in SOP and InShop datasets once we fix $\varepsilon = 5$ for Contrastive (C2) and $\varepsilon = 0.5$ for Proxy NCA++ (PNCA).

5.3.5 Comparison to Other Pooling Alternatives



(a) (b)

Figure 5.12: Evaluation of the feature pooling methods on (a) Ciffar Collage and (b) CUB dataset.

We evaluate 13 additional pooling alternatives on Ciffar Collage and CUB datasets with *contrastive* (C2) and *Proxy-NCA++* (PNCA) losses since they are the best performing sample-based and proxy-based losses, respectively, across datasets. We particularly consider Cifar Collage dataset since the images of different classes share a considerable amount of semantic entities, enabling us to assess the methods with respect to their ability to discard the nuisance information.

In addition to our method (GSP) and global average pooling (GAP), we consider:

i) global max pooling (GMP), *ii*) GAP+GMP [37], *iii* CBAM [85], *iv*) CroW [86], *v*) DeLF [87], *vi*) generalized max pooling (GeMax) [77], *vii*) generalized mean pooling (GeMean) [76], *viii*) GSoP [89], *ix*) optimal transport based aggregation (OTP) [82], *x*) SOLAR [80], *xi*) trainable SMK (T-SMK) [88], *xii*) NetVLAD [81], and *xiii*) WELDON [75]. Among those, OTP and VLAD are ensemble based methods and typically necessitate large embedding dimensions. Thus, we both experimented their 128 dimensional version -(8x16) (8 prototypes of 16 dimensional vectors) and 8192 dimensional version -(64x128) (64 prototypes of 128 dimensional vectors). We also evaluate our method with GMP (GAP+GMP) in CUB dataset to show that per channel selection is orthogonal to our approach and thus, GMP can marginally improve our method as well.

For CUB dataset, the experimental setting follows § 5.3.4.1 and we report MAP@R performance of the 4-model average at 128 dimensional embeddings each. For Cifar Collage dataset, the experimental setting follows § 5.3.2 and we report MAP@R performance. We provide the results in Fig. 5.12-(a) and Fig. 5.12-(b) for evaluations on Cifar Collage and CUB, respectively.

Evaluations show that our method is superior to other pooling alternatives including prototype based VLAD and OTP. Predominantly, for 128 dimensional embeddings, our method outperforms prototype based methods by large margin. In CUB dataset, the pooling methods either are inferior to or perform on par with GAP. The performance improvements of the superior methods are less than 1%, implying that our improvements in the order of 1-2% reported in Tab. 5.1 is substantial. On the other hand, the methods that mask the feature map outperform GAP by large margin in Cifar Collage dataset. That being said, our method outperforms all the methods except for Contrastive+VLAD by large margin in Cifar Collage dataset, yet another evidence for better feature selection mechanism of our method. For instance in CUB dataset, DeLF and GeMean are on par with our method which has slightly better performance. Yet, our method outperforms both methods by large margin in Cifar Collage dataset.

Comparing to CroW , T-SMK and CBAM, our method outperforms those methods by large margin. Those methods are the built on feature magnitude based saliency, assuming that the backbone functions must be able to zero-out nuisance information. Yet, such a requirement is restrictive for the parameter space and annihilation of the non-discriminative information might not be feasible in some problems. We experimentally observe such a weakness of CroW , T-SMK and CBAM in Cifar Collage dataset where the nuisance information cannot be zeroed-out by the backbone. Our formulation do not have such a restrictive assumption and thus substantially superior to those methods.

Similarly, attention-based weighting methods, DeLF and GSoP, do not have explicit control on feature selection behavior and might result in poor models when jointly trained with the feature extractor [87], which we also observe in Cifar Collage experiments. On the contrary, we have explicit control on the pooling behavior with μ parameter and the behavior of our method is stable and consistent across datasets and with different loss functions.

Moreover, attention-based methods DeLF, GSoP, and SOLAR typically introduce several convolution layers to compute the feature weights. We only introduce an m-kernel 1x1 convolution layer (*i.e.*,m-many trainable prototypes) and obtain better results. We should note that our pooling operation is as simple as performing a convolution (*i.e.*,distance computation) and alternating normalization of a vector and a scalar.

Other pooling methods, *i.e.*,GAP, GMP, GAP+GMP, GeMax, GeMean, WELDON, VLAD, OTP, do not build on discriminative feature selection. Thus, our method substantially outperforms those.

CHAPTER 6

SUMMARY AND CONCLUSIONS

6.1 Summary

This dissertation addresses the problem of deep metric learning (DML) with particular focus on improving the generalization performance of the existing efforts. Upon a comprehensive review of the existing work in the literature, we attempt to provide theoretical insights into the success of the existing methods. We then follow the mechanisms suggested by the theory to develop novel DML methods. Accordingly, we propose three novel approaches as well as a theoretically founded bridge explaining the mechanism to transfer the DML efforts to unseen classes.

Our first approach brings a different perspective to DML formulation in terms of chance constraints. Specifically, we formulate the deep metric problem as a chance constrained optimization problem. The initial formulation gives no clear algorithmic implications but the existing works. Thus, we rigorously convert the initial problem formulation into another form enabling expressing the deep metric problem as a set intersection problem. The theory suggests a set intersection mechanism yet allows a greedy algorithm via iterative projections. To this end, we also relate the solution of a proxy-based DML approach to one of the supersets to be intersected to obtain the desired solution. Hence, we form the sets to be intersected greedily through solving proxy-based DML problems on the fly. In short, the mechanism suggested by the theory is realized by a simple, yet effective, algorithm. We can summarize our key contribution as building a bridge between what the theory suggests and the actual practice applied by the method. In the end, we provide a mathematical programming to solve a chance constrained problem defining DML through our derivations and the proposed method. Notably, the proposed framework is applicable with the pairwise distance based state-of-the-art (SOTA) DML loss functions without introducing any additional computational cost. We support our claims with comprehensive empirical studies. Extensive evaluations on the benchmark datasets show the efficiency of our method.

Our next contribution is explaining the efficiency of global average pooling (GAP) in zero-shot prediction. We show that averaging the collection of features in the CNN feature map is equivalent to convex combination of latent prototype vectors of some semantic entities corresponding to classes. Hence, we interpret the representation space defined via output of GAP as a convex combination of semantically independent representations defined by each pixel in the feature map. We conclude that as long as the local semantic entities of the training and test classes follow the same distribution, metric learning task can be generalized to unseen classes. In that manner, we also show that class samples are mapped to some convex hulls defined by certain prototypes active for the class. The critical desiderata for metric learning is non-overlapping class convex hulls. With our analysis, we show that such a desired metric learning behaviour can be achieved by controlling the prototypes and the convex combination weights.

Building on perspective explaining the success of GAP, we propose a learnable and generalized version. Our proposed generalization is a trainable pooling layer that selects the feature subset and re-weight it during pooling. Formally, we propose an entropy-smoothed optimal transport problem and show that it is a strict generalization of GAP, *i.e.*,a specific realization of the problem gives back GAP. We show that this optimization problem enjoys analytical gradients enabling us to use it as a direct learnable replacement for GAP. To enable effective learning of the proposed layer, we further propose a regularization loss to improve zero-shot transfer. With extensive empirical studies, we validate the effectiveness of the proposed pooling layer in various metric learning benchmarks.

6.2 Conclusion

Existing evaluations on several benchmark datasets in the literature showed that proxy-based methods are state-of-the art, especially in large scale problems in terms of classes. We also observed from the results of such benchmarks that increasing the number of proxies does not proportionally improve the performance. To this end, we proposed a mathematical programming framework to arbitrarily increase the number of proxies arbitrarily while improving the generalization performance.

We theoretically showed that a contrastive loss based deep metric learning objective is a surrogate for the chance constraints that read the probability of proximity violations in the representation space is less than some small constant. We then theoretically showed that chance constraints can be decomposed into finite chance constraints conditioned on particular samples. We proved that the discrepancy between such alternative decomposed formulation and the original formulation is bounded above by the covering radius of the samples used to define the problem. Equivalently, we showed that the generalization performance is proportional to the covering radius of such samples. We also showed that the solution of a proxy-based DML corresponds to a superset of the feasible region of the primary DML problem and that solving multiple proxy-based problems can be related to solving a DML problem with better generalization properties. As a result, we formally developed a proxy-based method that inherently considers arbitrary number of proxies for better generalization.

We empirically validated that as we initialize proxies with diverse samples and solve proxy-based DML problems, we inherently increase the number of samples used to define the problem and hence reducing the covering radius. Our theoretical and empirical analysis showed that alternating the proxies is crucial to improve the generalization rather than exploiting multiple proxies at once. In particular, we observed more than 10% point MAP@R performance improvement in SOP and In-shop datasets when we altered the proxies rather than fixing them. Besides validating implications of our theoretical derivations, we showed that our chance constrained programming based method is state-of-the-art. In fact, we beat the previous state-of-the-art proxy-based XBM method by at least 1% point MAP@R performance margin. We also showed that we consistently increase the performance of the previous DML methods at least 1% and up to 11% points MAP@R with our method.

Moving forward, we theoretically analyzed how global average pooling shapes the embeddings space and how the metric learning with particular classes can be transferred to unseen classes. We showed that the learning can be transferred as long as the seen and the unseen classes share the same local semantics. We showed that global average pooling corresponds to convex combination of prototype vectors and that vectors are of some semantic entities. We showed the equivalence by bounding the error between the two representations. We validated our claims with experiments showcasing that the representation space can be partitioned using prototype vectors corresponding to semantic entities and that the covering radius of the prototype set is an upper bound for the error between the prototype convex combinations and the global average pooling representations.

Building on our theoretic analysis of GAP, we formulated a generalized sum pooling and proved that GAP is a corner case. We empirically showed that our pooling layer effectively selects discriminative features. We also developed a meta-learning approach to generalize the behaviour of our pooling method to unseen classes. Namely, we proposed a zero-shot prediction loss to enable learning transfer to unseen classes. We empirically showed that the proposed zero-shot prediction loss has more than 30% point MAP@R zero-shot generalization performance capability than that of a regular learning without our loss. We also showed in benchmark datasets that the standalone performance of our pooling method can be boosted by zero-shot prediction loss by 0.5% to 1% points MAP@R. With regard to discriminative feature selection capability of our pooling method, we showed in controlled datasets that inclusion of our pooling method improves the baseline performance by 70% points MAP@R in *synthetic dataset* and by 17% points MAP@R in *Cifar Collage dataset.* We as well showed in several benchmark datasets that our pooling method with zero-shot prediction loss consistently improves the baseline methods including the state-of-the-art XBM and CCP methods. In particular, we boosted the performance of XBM and CCP up to 2% and 0.5% points MAP@R, respectively. Finally, the results of the empirical study to evaluate the proposed pooling method validated our claims and theoretical findings introduced during the formulation of the pooling approach. Specifically, proposed feature selection resulted in non-overlapping convex hulls, the learned prototypes of the feature selection layer were to correspond some semantic entities, and proposed zero-shot prediction loss yielded learned prototypes that have superior generalization capability.

6.3 Future Directions

Among the methods we propose withing the scope of this dissertation, chance constrained programming (CCP) framework demands relatively more frequent computation of validation performance, which becomes displeasing in large scale problems. One approach to control the computation load that the validation performance computation brings can be selecting a representative subset of the validation set. To select such a set, our findings on how global average pooling shapes the local features as well as the on the generalization bounds can be further studied to develop an algorithm to select a subset that possesses δ-cover of the local features.

Our CCP approach aims to reduce the covering radius of the proxies that define the chance constraints. We build on a set intersection formulation in our approach to increase the number of diverse proxies. An alternative research direction to reduce the covering radius can be transforming the embedding space to map each sample to very close vicinity of their associated proxy. To this end, distances between the proxies of the different classes can be considered as a measure for label ordering and multi-dimensional scaling (MDS) can be used to map the samples according to such orderings. An MDS based ordinal metric learning approach has been already proposed in [121], yet the label orderings are essential in that study. Expanding on [121] by replacing label orderings with the distances among the proxies can be an interesting future direction.

Our generalized sum pooling enables localization of the discriminative content in the images. We only deal with the images containing singe objects within the scope of the thesis. Exploiting such a localization byproduct of our pooling operation in object detection and instance segmentation tasks can be an interesting future work direction.

Moreover, we learn feature prototypes with our pooling layer. Such feature prototypes can be used to generate synthetic class samples. Specifically, an optimal transport barycenter of some input images can be computed using the prototypes as the support of the barycenter distribution. Current synthetic sample generation approaches operate on global feature level or use local features with simple weighted averaging. In this manner, extending such works using optimal transport barycenters thanks to our learned prototypes efficiently can be an interesting research direction.

To sum, our theoretical insights can inspire many future works that somehow exploit a metric learning task. Examples of such works include but not limited to synthetic data generation, feature selection, object detection, instance segmentation, subset selection, many-to-one feature mapping.

REFERENCES

[1] J. Seidenschwarz, I. Elezi, and L. Leal-Taixé, “Learning intra-batch connections for deep metric learning,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139 of *Proceedings of Machine Learning Research*, pp. 9410–9421, PMLR, 2021.

[2] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, “A survey of appearance models in visual object tracking,” *ACM transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, pp. 1–48, 2013.

[3] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, “A comprehensive survey of deep learning for image captioning,” *ACM Computing Surveys (CsUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[4] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, “Deep learning for generic object detection: A survey,” *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[5] S. Hao, Y. Zhou, and Y. Guo, “A brief survey on semantic segmentation with deep learning,” *Neurocomputing*, vol. 406, pp. 302–321, 2020.

[6] K. Musgrave, S. Belongie, and S.-N. Lim, “A metric learning reality check,” in *European Conference on Computer Vision*, pp. 681–699, Springer, 2020.

[7] K. Roth, T. Milbich, S. Sinha, P. Gupta, B. Ommer, and J. P. Cohen, “Revisiting training strategies and generalization performance in deep metric learning,” in *International Conference on Machine Learning*, pp. 8242–8252, PMLR, 2020.

[8] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, “A survey on contrastive self-supervised learning,” *Technologies*, vol. 9, no. 1, p. 2, 2020.

[9] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, “Sampling matters in deep embedding learning,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2840–2848, 2017.

[10] B. Ko and G. Gu, “Embedding expansion: Augmentation in embedding space for deep metric learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7255–7264, 2020.

[11] C. Liu, H. Yu, B. Li, Z. Shen, Z. Gao, P. Ren, X. Xie, L. Cui, and C. Miao, “Noise-resistant deep metric learning with ranking-based instance selection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6811–6820, 2021.

[12] G. Gu, B. Ko, and H.-G. Kim, “Proxy synthesis: Learning with synthetic classes for deep metric learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[13] S. Venkataramanan, B. Psomas, E. Kijak, laurent amsaleg, K. Karantzalos, and Y. Avrithis, “It takes two to tango: Mixup for deep metric learning,” in *International Conference on Learning Representations*, 2022.

[14] X. Wang, H. Zhang, W. Huang, and M. R. Scott, “Cross-batch memory for embedding learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6388–6397, 2020.

[15] E. W. Teh, T. DeVries, and G. W. Taylor, “Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis,” in *European Conference on Computer Vision (ECCV)*, Springer, 2020.

[16] M. Dong, X. Yang, R. Zhu, Y. Wang, and J. Xue, “Generalization bound of gradient descent for non-convex metric learning,” in *Advances in Neural Information Processing Systems*, Neural Information Processing Systems Foundation, 2020.

[17] Y. Lei, M. Liu, and Y. Ying, “Generalization guarantee of sgd for pairwise learning,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[18] D. Zhang, Y. Li, and Z. Zhang, “Deep metric learning with spherical embedding,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[19] Y. Kim and W. Park, “Multi-level distance regularization for deep metric learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1827–1835, 2021.

[20] H. Xuan, R. Souvenir, and R. Pless, “Deep randomized ensembles for metric learning,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 723–734, 2018.

[21] A. Sanakoyeu, V. Tschernezki, U. Buchler, and B. Ommer, “Divide and conquer the embedding space for metric learning,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[22] W. Zheng, B. Zhang, J. Lu, and J. Zhou, “Deep relational metric learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12065–12074, 2021.

[23] T. Milbich, K. Roth, H. Bharadhwaj, S. Sinha, Y. Bengio, B. Ommer, and J. P. Cohen, “Diva: Diverse visual feature aggregation for deep metric learning,” in *European Conference on Computer Vision*, pp. 590–607, Springer, 2020.

[24] K. Roth, T. Milbich, B. Ommer, J. P. Cohen, and M. Ghassemi, “S2sd: Simultaneous similarity-based self-distillation for deep metric learning,” in *ICML 2021: 38th International Conference on Machine Learning*, pp. 9095–9106, 2021.

[25] Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, and R. Jin, “Softtriple loss: Deep metric learning without triplet sampling,” in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[26] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, “Distance metric learning with application to clustering with side-information,” in *Advances in neural information processing systems*, pp. 521–528, 2003.

[27] K. Q. Weinberger, J. Blitzer, and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification,” in *Advances in neural information processing systems*, pp. 1473–1480, 2006.

[28] M. Perrot and A. Habrard, “Regressive virtual metric learning,” in *Advances in Neural Information Processing Systems*, pp. 1810–1818, 2015.

[29] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, vol. 2, pp. 1735–1742, IEEE, 2006.

[30] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[31] B. Harwood, B. Kumar, G. Carneiro, I. Reid, T. Drummond, *et al.*, “Smart mining for deep metric learning,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2821–2829, 2017.

[32] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, “Deep metric learning via lifted structured feature embedding,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4004–4012, 2016.

[33] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, “Multi-similarity loss with general pair weighting for deep metric learning,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[34] K. Sohn, “Improved deep metric learning with multi-class n-pair loss objective,” in *Advances in Neural Information Processing Systems*, pp. 1857–1865, 2016.

[35] B. Yu and D. Tao, “Deep metric learning with tuplet margin loss,” in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[36] X. Wang, Y. Hua, E. Kodirov, G. Hu, R. Garnier, and N. M. Robertson, "Ranked list loss for deep metric learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[37] S. Kim, D. Kim, M. Cho, and S. Kwak, "Proxy anchor loss for deep metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3238–3247, 2020.

[38] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, "Deep metric learning with angular loss," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2612–2620, IEEE, 2017.

[39] E. Ustinova and V. Lempitsky, "Learning deep embeddings with histogram loss," in *Advances in Neural Information Processing Systems*, pp. 4170–4178, 2016.

[40] M. T. Law, N. Thome, and M. Cord, "Quadruplet-wise image similarity learning," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 249–256, 2013.

[41] C. Huang, C. C. Loy, and X. Tang, "Local similarity-aware deep feature embedding," in *Advances in neural information processing systems*, pp. 1262–1270, 2016.

[42] W. Chen, X. Chen, J. Zhang, and K. Huang, "Beyond triplet loss: a deep quadruplet network for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 403–412, 2017.

[43] G. Chen, T. Zhang, J. Lu, and J. Zhou, "Deep meta metric learning," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[44] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy, "Deep metric learning via facility location," in *Computer Vision and Pattern Recognition (CVPR)*, vol. 8, 2017.

[45] M. T. Law, R. Urtasun, and R. S. Zemel, “Deep spectral clustering learning,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1985–1994, JMLR. org, 2017.

[46] Y. Duan, J. Lu, J. Feng, and J. Zhou, “Deep localized metric learning,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2644–2656, 2017.

[47] Y. Yuan, K. Yang, and C. Zhang, “Hard-aware deeply cascaded embedding,” in *Proceedings of the IEEE international conference on computer vision*, pp. 814–823, 2017.

[48] W. Ge, “Deep metric learning with hierarchical triplet loss,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 269–285, 2018.

[49] Y. Suh, B. Han, W. Kim, and K. M. Lee, “Stochastic class-based hard example mining for deep metric learning,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[50] J. D. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka, “Contrastive learning with hard negative samples,” in *International Conference on Learning Representations*, 2020.

[51] Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou, “Deep adversarial metric learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2780–2789, 2018.

[52] Y. Zhao, Z. Jin, G.-j. Qi, H. Lu, and X.-s. Hua, “An adversarial approach to hard triplet generation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 501–517, 2018.

[53] W. Zheng, Z. Chen, J. Lu, and J. Zhou, “Hardness-aware deep metric learning,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[54] X. Lin, Y. Duan, Q. Dong, J. Lu, and J. Zhou, “Deep variational metric learning,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 689–704, 2018.

[55] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 360–368, 2017.

[56] B. Chen, W. Deng, and H. Shen, "Virtual class enhanced discriminative embedding learning," *Advances in Neural Information Processing Systems*, vol. 31, pp. 1942–1952, 2018.

[57] T.-T. Do, T. Tran, I. Reid, V. Kumar, T. Hoang, and G. Carneiro, "A theoretically sound upper bound on the triplet loss for improving the efficiency of deep distance metric learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[58] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev, "Metric learning with adaptive density discrimination," *International Conference on Learning Representations*, 2016.

[59] Y. Zhu, M. Yang, C. Deng, and W. Liu, "Fewer is more: A deep graph metric learning perspective using fewer proxies," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17792–17803, 2020.

[60] M. Opitz, G. Waltner, H. Possegger, and H. Bischof, "Bier-boosting independent embeddings robustly," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5189–5198, 2017.

[61] J. Lu, J. Hu, and Y.-P. Tan, "Discriminative deep metric learning for face and kinship verification," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4269–4282, 2017.

[62] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon, "Attention-based ensemble for deep metric learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 736–751, 2018.

[63] K. Roth, B. Brattoli, and B. Ommer, "Mic: Mining interclass characteristics for improved metric learning," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[64] P. Jacob, D. Picard, A. Histace, and E. Klein, “Metric learning with horde: High-order regularizer for deep embeddings,” in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[65] W. Zhao, Y. Rao, Z. Wang, J. Lu, and J. Zhou, “Towards interpretable deep metric learning with structural matching,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9887–9896, 2021.

[66] M. Cuturi, “Sinkhorn distances: Lightspeed computation of optimal transport,” *Advances in neural information processing systems*, vol. 26, 2013.

[67] H. Xu and S. Mannor, “Robustness and generalization,” *Machine learning*, vol. 86, no. 3, pp. 391–423, 2012.

[68] A. Bellet and A. Habrard, “Robustness and generalization for metric learning,” *Neurocomputing*, vol. 151, pp. 259–267, 2015.

[69] O. Sener and S. Savarese, “Active learning for convolutional neural networks: A core-set approach,” in *International Conference on Learning Representations*, 2018.

[70] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” in *Computer Vision, IEEE International Conference on*, vol. 3, pp. 1470–1470, IEEE Computer Society, 2003.

[71] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, “Large-scale image retrieval with compressed fisher vectors,” in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3384–3391, IEEE, 2010.

[72] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3304–3311, IEEE, 2010.

[73] G. Tolias, Y. Avrithis, and H. Jégou, “To aggregate or not to aggregate: Selective match kernels for image search,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1401–1408, 2013.

[74] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proceedings of the IEEE international conference on computer vision*, pp. 1269–1277, 2015.

[75] T. Durand, N. Thome, and M. Cord, "Weldon: Weakly supervised learning of deep convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4743–4752, 2016.

[76] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning cnn image retrieval with no human annotation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1655–1668, 2018.

[77] N. Murray, H. Jégou, F. Perronnin, and A. Zisserman, "Interferences in match kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 9, pp. 1797–1810, 2016.

[78] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, "Compact bilinear pooling," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 317–326, 2016.

[79] M. Simon, Y. Gao, T. Darrell, J. Denzler, and E. Rodner, "Generalized orderless pooling performs implicit salient matching," in *Proceedings of the IEEE international conference on computer vision*, pp. 4960–4969, 2017.

[80] T. Ng, V. Balntas, Y. Tian, and K. Mikolajczyk, "Solar: second-order loss and attention for image retrieval," in *European conference on computer vision*, pp. 253–270, Springer, 2020.

[81] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5297–5307, 2016.

[82] G. Mialon, D. Chen, A. d'Aspremont, and J. Mairal, "A trainable optimal transport embedding for feature aggregation and its relationship to attention," in *ICLR 2021-The Ninth International Conference on Learning Representations*, 2021.

[83] M. Teichmann, A. Araujo, M. Zhu, and J. Sim, "Detect-to-retrieve: Efficient regional aggregation for image search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5109–5118, 2019.

[84] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *International Journal of Computer Vision*, vol. 124, no. 2, pp. 237–254, 2017.

[85] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

[86] Y. Kalantidis, C. Mellina, and S. Osindero, "Cross-dimensional weighting for aggregated deep convolutional features," in *European conference on computer vision*, pp. 685–701, Springer, 2016.

[87] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-scale image retrieval with attentive deep local features," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[88] G. Tolias, T. Jenicek, and O. Chum, "Learning and aggregating deep local descriptors for instance-level recognition," in *European Conference on Computer Vision*, pp. 460–477, Springer, 2020.

[89] Z. Gao, J. Xie, Q. Wang, and P. Li, "Global second-order pooling convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3024–3033, 2019.

[90] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of cnn activations," in *ICLR 2016-International Conference on Learning Representations*, pp. 1–12, 2016.

[91] Y. Xie, H. Dai, M. Chen, B. Dai, T. Zhao, H. Zha, W. Wei, and T. Pfister, "Differentiable top-k with optimal transport," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20520–20531, 2020.

[92] M. Cuturi, O. Teboul, and J.-P. Vert, “Differentiable ranking and sorting using optimal transport,” *Advances in neural information processing systems*, vol. 32, 2019.

[93] G. Luise, A. Rudi, M. Pontil, and C. Ciliberto, “Differential properties of sinkhorn approximation for learning with wasserstein distance,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[94] B. Demirel, R. Gokberk Cinbis, and N. Ikizler-Cinbis, “Attributes2classname: A discriminative model for attribute-based unsupervised zero-shot learning,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1232–1241, 2017.

[95] W. Xu, Y. Xian, J. Wang, B. Schiele, and Z. Akata, “Attribute prototype network for zero-shot learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 21969–21980, 2020.

[96] D. Huynh and E. Elhamifar, “Fine-grained generalized zero-shot learning via dense attribute-based attention,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4483–4493, 2020.

[97] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi, “Meta-learning with differentiable closed-form solvers,” in *International Conference on Learning Representations*, 2018.

[98] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, “Meta-learning with differentiable convex optimization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

[99] L. M. Bregman, “The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming,” *USSR computational mathematics and mathematical physics*, vol. 7, no. 3, pp. 200–217, 1967.

[100] H. H. Bauschke and A. S. Lewis, “Dykstras algorithm with bregman

projections: A convergence proof," *Optimization*, vol. 48, no. 4, pp. 409–427, 2000.

[101] C. Pang, "Nonconvex set intersection problems: From projection methods to the newton method for super-regular sets," *arXiv preprint arXiv:1506.08246*, 2015.

[102] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas, "Convolutional wasserstein distances: Efficient optimal transportation on geometric domains," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 66, 2015.

[103] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *Advances in neural information processing systems*, vol. 30, 2017.

[104] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21271–21284, 2020.

[105] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, pp. 630–645, Springer, 2016.

[106] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[107] 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," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[108] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[109] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: a system for large-scale machine learning.," in *OSDI*, vol. 16, pp. 265–283, 2016.

[110] I. Fehervari, A. Ravichandran, and S. Appalaraju, “Unbiased evaluation of deep metric learning algorithms,” *arXiv preprint arXiv:1911.12528*, 2019.

[111] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, “Deepfashion: Powering robust clothes recognition and retrieval with rich annotations,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1096–1104, 2016.

[112] A. Krause and D. Golovin, “Submodular function maximization,” in *Tractability: Practical Approaches to Hard Problems*, pp. 71–104, Cambridge University Press, 2014.

[113] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, “The caltech-ucsd birds-200-2011 dataset,” 2011.

[114] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[115] Y. Z. Gürbüz, O. Can, and A. A. Alatan, “Asap dml: Deep metric learning with alternating sets of alternating proxies,” 2021.

[116] C. Villani, *Optimal transport: old and new*, vol. 338. Springer Science & Business Media, 2008.

[117] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.

[118] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. Lanckriet, “Hilbert space embeddings and metrics on probability measures,” *The Journal of Machine Learning Research*, vol. 11, pp. 1517–1561, 2010.

[119] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” tech. rep., Citeseer, 2009.

[120] Y. Z. Gürbüz and A. A. Alatan, “A novel bovw mimicking end-to-end trainable cnn classification framework using optimal transport theory,” in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 3053–3057, IEEE, 2019.

[121] P. Yu and Q. Li, “Ordinal distance metric learning with mds for image ranking,” *Asia-Pacific Journal of Operational Research*, vol. 35, no. 01, p. 1850007, 2018.

[122] A. W. Van Der Vaart, A. van der Vaart, A. W. van der Vaart, and J. Wellner, *Weak convergence and empirical processes: with applications to statistics.* Springer Science & Business Media, 2013.

[123] T.-T. Lu and S.-H. Shiou, “Inverses of 2× 2 block matrices,” *Computers & Mathematics with Applications*, vol. 43, no. 1-2, pp. 119–129, 2002.

Appendix A

PROOFS FOR CHAPTER 3

A.1 Proof for Lemma 3.2.1

Lemma 3.2.1 *Generalized contrastive loss defined as*

$$\ell(z, z'; \theta) := (\iota_{y,y'}(\|x - x'\|_{f_\theta} - \beta) + \alpha)_+$$

is $\sqrt{2}\omega^L$*-Lipschitz in* x *and* x' *for all* y, y', θ *for the embedding function* $f(\cdot;\theta)$ *being L-layer CNN (with ReLU, max-pool, average-pool) with a fully connected layer at the end, where* ω *is the maximum sum of the input weights per neuron.*

Proof: We first show that $f(x;\theta)$ is Lipschitz continuous.

We consider $x\in\mathbb{R}^d$ as an input to a layer and $\hat{x}\in\mathbb{R}^{d'}$ as the corresponding output. We express i^{th} component of $\hat{x}$ as $\hat{x}_i = \sum_j w_{i,j} x_{s_i(j)}$ where $s_i = \{s_i(j) \in [d]\}$ is the set of components contributing to $\hat{x}_i$ and $w_{i,j}\in\theta$ is the layer weights. For instance, for a fully connected layer $s_i(j) = j$; for a 3x3 convolutional layer, s_i corresponds to 3x3 window of depth *#channels* centered at i. We now consider two inputs x, x' and their outputs $\hat{x}, \hat{x}'$. We write:

$$\begin{aligned}\frac{\|\hat{x} - \hat{x}'\|_2^2}{\|x - x'\|_2^2} &= \frac{\sum_{i\in[d']} |\hat{x}_i - \hat{x}'_i|^2}{\|x - x'\|_2^2} = \frac{\sum_{i\in[d']} |\sum_j w_{i,j} x_{s_i(j)} - \sum_j w_{i,j} x'_{s_i(j)}|^2}{\|x - x'\|_2^2} \\ &\leqslant \frac{\sum_{i\in[d']} \sum_j |w_{i,j}|^2 |x_{s_i(j)} - x'_{s_i(j)}|^2}{\|x - x'\|_2^2}\end{aligned}$$

Rearranging terms, we express:

$$\textstyle\sum_{i\in[d']} \sum_j |w_{i,j}|^2 |x_{s_i(j)} - x'_{s_i(j)}|^2 = \sum\limits_{k\in[d]} \sum\limits_{i,j:s_i(j)=k} |w_{i,j}|^2 |x_k - x'_k|^2$$

If $\sum\limits_{i,j:s_i(j)=k}|w_{i,j}| \leqslant \omega$ for all k and for all layers, *i.e.*,the absolute sum of the input weights per neuron is bounded by ω, we can write $\sum\limits_{k\in[d]}\sum\limits_{i,j:s_i(j)=k}|w_{i,j}|^2|x_k - x'_k|^2 \leqslant \omega^2\sum\limits_{k\in[d]}|x_k - x'_k|^2 \leqslant \omega^2\|x - x'\|_2^2$, hence,

$$\frac{\|\hat{x} - \hat{x}'\|_2}{\|x - x'\|_2} \leqslant \omega.$$

For max-pooling and average-pooling layers, the inequality holds with $\omega = 1$; since, we can express max-pooling as a convolution where only one weight is 1 and the rest is 0; and similarly, we can express average-pooling as a convolution where the weights sum up to 1.

For ReLU activation, we consider the fact that $|\max\{0,u\} - \max\{0,v\}| \leqslant |u - v|$ to write:

$$\frac{\|ReLU(x) - ReLU(x')\|_2}{\|x - x'\|_2} \leqslant 1.$$

Therefore, L-layer CNN $f(x;\theta)$ is ω^L-Lipschitz.

We now consider $\ell(z,z';\theta) = \max\{0, \iota_{y,y'}(\|f(x;\theta) - f(x';\theta)\|_2 - \beta) + \alpha\}$ as $g(h(f(x;\theta), f(x';\theta)))$ where $g(h) = \max\{0, \iota_{y,y'}(h - \beta) + \alpha\}$ is 1-Lipschitz, and $h(f,f') = \|f - f'\|_2$ is $\sqrt{2}$-Lipschitz and 1-Lipschitz in f for fixed f'. Thus, for y, y', θ fixed, $\ell(z,z';\theta) := (\iota_{y,y'}(d_f(x,x';\theta) - \beta) + \alpha)_+$ is ω^L-Lipschitz in x and in x'; and $\sqrt{2}\omega^L$-Lipschitz in both, for all y, y', θ. ■

Note that it is easy to show that the normalization proposed in Section 4.4:

$$\hat{v} = \begin{cases} v & \text{for } \|v\|_2 \leqslant 1 \\ {}^{v}\!/\!_{\|v\|_2} & \text{for } \|v\|_2 \geqslant 1 \end{cases}$$

is 2-Lipschitz. Therefore, our loss is still Lipschitz continuous with normalized embeddings in our framework.

A.2 Proof for Proposition 3.2.1

Proposition 3.2.1 *Given* $\mathcal{S}{=}\{z_i\}_{i\in[m]} \overset{i.i.d.}{\sim} p_{\mathcal{Z}}$ *such that* $\forall k{\in}\mathcal{Y}$ $\{x_i|y_i{=}k\}$ *is* $\delta_{\mathcal{S}}$-*cover*[1] *of* $\mathcal{X}$, $\ell(z,z';\theta)$ *is* ζ-*Lipschitz in* x, x' *for all* y, y' *and* θ, *and bounded by*

---

[1] $\mathcal{S} \subset \mathcal{S}'$ is $\delta_{\mathcal{S}}$-cover of $\mathcal{S}'$ if $\forall z' \in \mathcal{S}'$, $\exists z \in \mathcal{S}$ such that $\|z - z'\|_2 \leqslant \delta_{\mathcal{S}}$.

L; then with probability at least $1-\gamma$,

$$\left|\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}\left[\ell(z,z';\theta)\right] - \tfrac{1}{m}\sum_{i\in[m]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}\left[\ell(z_i,z;\theta)\right]\right| \leqslant \mathcal{O}(\zeta\,\delta_{\mathcal{S}}) + \mathcal{O}(L\sqrt{\log \tfrac{1}{\gamma}/m}).$$

Proof: We start with defining $\hat{\mathcal{L}}(z;\theta) := \mathbb{E}_{z'\sim p_{\mathcal{Z}}}\left[\ell(z,z';\theta)\right]$. Note that

$$\begin{aligned}\|\hat{\mathcal{L}}(z_1;\theta) - \hat{\mathcal{L}}(z_2;\theta)\|_2 &= |\,\mathbb{E}_{z'\sim p_{\mathcal{Z}}}\left[\ell(z_1,z';\theta)\right] - \mathbb{E}_{z'\sim p_{\mathcal{Z}}}\left[\ell(z_2,z';\theta)\right]|\\ &\leqslant \mathbb{E}_{z'\sim p_{\mathcal{Z}}}\left[|\ell(z_1,z';\theta) - \ell(z_2,z';\theta)|\right].\end{aligned}$$

Therefore, $\ell(z,z';\theta)$ being ζ-Lipschitz in x for fixed x', y, y' and θ, and bounded by L implies $\hat{\mathcal{L}}(z;\theta)$ is also ζ-Lipschitz in x for all y, θ and bounded by L. Hence, we have

$$|\hat{\mathcal{L}}(z_i;\theta) - \hat{\mathcal{L}}(z;\theta)| \leqslant \zeta\,\delta_{\mathcal{S}} \quad \forall z_i, z{:}z_i \in \mathcal{S}, z \in \mathcal{Z}, \|z_i - z\|_2 \leqslant \delta_{\mathcal{S}}$$

From Theorem 14 of [67], we can partition $\mathcal{Z}$ into $K = \min_t\{|t|{:}t \text{ is } \frac{\delta_{\mathcal{S}}}{2}\text{-cover of } \mathcal{Z}\}$ disjoint sets, denoted as $\{\mathcal{R}_i\}_{i\in[K]}$, such that $\forall i{:}z_i \in \delta_{\mathcal{S}}$; both z_i, z being $\in \mathcal{R}_i$ implies $|\hat{\mathcal{L}}(z_i;\theta) - \hat{\mathcal{L}}(z;\theta)| \leqslant \zeta\,\delta_{\mathcal{S}}$. Hence, from Theorem 3 of [67], with probability at least $1-\gamma$, we have:

$$\begin{aligned}&\left|\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}\left[\ell(z,z';\theta)\right] - \tfrac{1}{m}\sum_{i\in[m]}\mathbb{E}_{z\sim p_{\mathcal{Z}}}\left[\ell(z_i,z;\theta)\right]\right| =\\ &\left|\mathbb{E}_{z\sim p_{\mathcal{Z}}}\left[\hat{\mathcal{L}}(z;\theta)\right] - \tfrac{1}{m}\sum_{i\in[m]}\hat{\mathcal{L}}(z_i;\theta)\right| \leqslant \zeta\,\delta_{\mathcal{S}} + L\sqrt{\frac{2\,K\,\log 2 + 2\,\log {}^1\!/\!_\gamma}{m}}\end{aligned}$$

Note that K is dependent on δ_s and satisfies $\lim\limits_{m\to\infty}\frac{K}{m} \to 0$ ensuring that the right hand side goes to zero as more samples are exploited and the covering radius is improved. Thus, asymptotically the following holds:

$$\mathbb{E}_{z,z'\sim p_{\mathcal{Z}}}\left[\ell(z,z';\theta^*)\right] \leqslant \mathcal{O}(\delta_s) + \mathcal{O}(\sqrt{\log \tfrac{1}{\gamma}/m}) \text{ with probability at least } 1-\gamma\,.$$

■

A.3 Proof for Proposition 3.2.2

Proposition 3.2.2 *Given* $\{z_i\}_{i\in[n]} \overset{i.i.d.}{\sim} p_{\mathcal{Z}}$ *and a set* $s \subset [n]$. *If* $s = \cup_k s'_k$ *with* s'_k *is the* δ_s*-cover of* $\{i \in [n] \mid y_i = k\}$ *(*i.e.,*the samples in class* k *),* $\ell(z,z';\theta)$ *is*

ζ-Lipschitz in x, x' for all y, y' and θ, and bounded by L, $e(\mathcal{A}_{s\times[n]})$ training error; then with probability at least $1-\gamma$ we have:

$$\left|\frac{1}{n^2}\sum_{i,j\in[n]\times[n]}\ell(z_i,z_j;\mathcal{A}_{s\times[n]})-\frac{1}{|s|\,n}\sum_{i,j\in s\times[n]}\ell(z_i,z_j;\mathcal{A}_{s\times[n]})\right|$$
$$\leqslant \mathcal{O}(\zeta\,\delta_s)+\mathcal{O}(e(\mathcal{A}_{s\times[n]}))+\mathcal{O}(L\sqrt{\log \tfrac{1}{\gamma}/n})$$

Proof:

We are given a condition on s that we can partition $\mathcal{Z}$ into $m=|s|$ disjoint sets such that any sample from the dataset $(x_i,c), i\in[n]$, has a corresponding sample from s, $(x'_j,c), j\in s$ within δ_s ball. Thus, we start with partitioning $\mathcal{Z}$ into s disjoint sets as $\mathcal{Z}=\cup_i\mathcal{S}_i$ with $\mathcal{S}_i\cap\mathcal{S}_j=\emptyset$, $\forall i\neq j$.

We define $\ell_{[n]}(z)=\frac{1}{n}\sum\limits_{i\in[n]}\ell(z,z_i,\mathcal{A}_{s\times[n]})$ and $\ell_s(z)=\frac{1}{m}\sum\limits_{i\in s}\ell(z,z_i,\mathcal{A}_{s\times[n]})$ for the sake of clarity. Hence, we are interested in bounding $|\frac{1}{n}\sum_{[n]}\ell_{[n]}(z_i)-\frac{1}{m}\sum_s\ell_{[n]}(z_i)|$. We proceed with using triangle inequality to write:

$$\left|\frac{1}{n}\sum_{i\in[n]}\ell_{[n]}(z_i)-\frac{1}{m}\sum_{i\in s}\ell_{[n]}(z_i)\right|$$
$$\leqslant\left|\frac{1}{n}\sum_{i\in[n]}\ell_{[n]}(z_i)-\sum_{i\in s}\frac{n_i}{n}\ell_{[n]}(z_i)\right|^{(T1)}+\left|\sum_{i\in s}\frac{n_i}{n}\ell_{[n]}(z_i)-\frac{1}{m}\sum_{i\in s}\ell_{[n]}(z_i)\right|^{(T2)}$$

For term $(T1)$ we write:

$$(T1)\leqslant\frac{1}{n}\sum_{i\in[m]}\sum_{z_j\in\mathcal{S}_i}|\ell_{[n]}(z_{s(i)})-\ell_{[n]}(z_j)|\overset{(1)}{\leqslant}\zeta\,\delta_s$$

where in (1), we use ζ-Lipschitz of the loss function and the condition $|z_{s(i)}-z_j|\leqslant\delta_s$, $\forall z_j\in\mathcal{S}_i$.

Using triangle inequality, we bound term $(T2)$ as:

$$\left|\sum_{i\in s}\frac{n_i}{n}\ell_{[n]}(z_i)-\frac{1}{m}\sum_{i\in s}\ell_{[n]}(z_i)\right|\leqslant\left|\mathbb{E}_{z\sim p_{\mathcal{Z}}}\left[\ell_s(z)\right]-\mathbb{E}_{z\sim p_{\mathcal{Z}}}\left[\ell_{[n]}(z)\right]\right|^{(T2.1)}$$
$$+\left|\mathbb{E}_{z\sim p_{\mathcal{Z}}}\left[\ell_{[n]}(z)\right]-\sum_{i\in s}\frac{n_i}{n}\ell_{[n]}(z_i)\right|^{(T2.2)}+\left|\mathbb{E}_{z\sim p_{\mathcal{Z}}}\left[\ell_s(z)\right]-\frac{1}{n}\sum_{i\in[n]}\ell_s(z_i)\right|^{(T2.3)}$$

where we use $\frac{1}{m}\sum_s\ell_{[n]}(z_i)=\frac{1}{n}\sum_{[n]}\ell_s(z_i)$ in $(T2.3)$.

For $(T2.1)$ we have:

$$(T2.1) \leqslant \left| \mathbb{E}_{z \sim p_{\mathcal{Z}}} \left[\frac{1}{m} \sum_{i \in s} \ell(z_i, z) - \frac{1}{n} \sum_{i \in [n]} \ell(z_i, z) \right] \right|$$

where we abuse the notation for the sake of clarity and drop parameter, $\mathcal{A}_{s\mathsf{x}[n]}$, dependency from the loss. Rearranging the terms, we have:

$$(T2.1) \leqslant$$
$$\left| \mathbb{E}_{z \sim p_{\mathcal{Z}}} \left[\frac{1}{m} \sum_{i \in [m]} \frac{n - m\, n_i}{n} \ell(z_{s(i)}, z) \right] \right| + \left| \mathbb{E}_{z \sim p_{\mathcal{Z}}} \left[\frac{1}{n} \sum_{i \in [m]} \sum_{j \in \mathcal{S}_i} \ell(z_{s(i)}, z) - \ell(z_j, z) \right] \right|$$

where similar to $(T1)$, the second summand is upper bounded by $\zeta\, \delta_s$. Using triangle inequality for the first summand, we write:

$$\left| \mathbb{E}_{z \sim p_{\mathcal{Z}}} \left[\frac{1}{m} \sum_{i \in [m]} \frac{n - m\, n_i}{n} \ell(z_{s(i)}, z) \right] \right| \leqslant (T2.3) + e(\mathcal{A}_{s\mathsf{x}[n]})$$

Hence, we have:

$$(T2.1) \leqslant \zeta\, \delta_s + (T2.3) + e(\mathcal{A}_{s\mathsf{x}[n]})$$

where from Hoeffding's Bound, $(T2.3) \leqslant L\sqrt{\log \frac{1}{\gamma} / 2n}$ with probability at least $1 - \gamma$:

Finally, we express $(T2.2)$ as:

$$\begin{aligned}
(T2.2) &= \left| \sum_{i \in [m]} \mathbb{E}_{z \sim p_{\mathcal{Z}}} \left[\ell_{[n]}(z) \mid z \in \mathcal{S}_i \right] p(z \in \mathcal{S}_i) - \sum_{i \in s} \frac{n_i}{n} \ell_{[n]}(z_i) \right| \\
&\leqslant \left| \sum_{i \in [m]} \mathbb{E}_{z \sim p_{\mathcal{Z}}} \left[\ell_{[n]}(z) \mid z \in \mathcal{S}_i \right] \frac{n_i}{n} - \sum_{i \in s} \frac{n_i}{n} \ell_{[n]}(z_i) \right| \\
&+ \left| \sum_{i \in [m]} \mathbb{E}_{z \sim p_{\mathcal{Z}}} \left[\ell_{[n]}(z) \mid z \in \mathcal{S}_i \right] p(z \in \mathcal{S}_i) - \sum_{i \in [m]} \mathbb{E}_{z \sim p_{\mathcal{Z}}} \left[\ell_{[n]}(z) \mid z \in \mathcal{S}_i \right] \frac{n_i}{n} \right|
\end{aligned}$$

Rearranging the terms we have:

$$(T2.2) \leqslant \sum_{i \in [m]} \frac{n_i}{n} \max_{z \in \mathcal{S}_i} \left| \ell_{[n]}(z) - \ell_{[n]}(z_{s(i)}) \right| + \max_{z \in \mathcal{Z}} \left| \ell_{[n]}(z) \right| \sum_{i \in [m]} \left| \frac{n_i}{n} - p(z \in \mathcal{S}_i) \right|$$

where the first summand is bounded above by $\zeta\,(\delta_s + \varepsilon(n))$ owing to loss being ζ-Lipschitz. Here, we denote $\varepsilon(n)$ as the covering radius of $\mathcal{Z}$, *i.e.*,the dataset, $\{x_i, y_i\}_{[n]}$ is $\varepsilon(n)$-cover of $\mathcal{X}\mathsf{x}\mathcal{Y}$. We note that $(n_i)_{i \in [m]}$ is an *i.i.d.* multinomial random variable with parameters n and $(p_{\mathcal{Z}}(z \in \mathcal{S}_i))_{i \in [m]}$. Thus, by the Breteganolle-Huber-Carol inequality (Proposition A6.6 of [122]), we have :

$$(T2.2) \leqslant \zeta\,(\delta_s + \varepsilon(n)) + L\sqrt{\frac{2m \log 2 + 2 \log 1/\gamma}{n}}$$

Finally, with probability at least $1-\gamma$, we end up with:

$$\begin{aligned}&\left|\frac{1}{n}\sum_{i\in[n]}\ell_{[n]}(z_i)-\frac{1}{m}\sum_{i\in s}\ell_{[n]}(z_i)\right|\\&\qquad\leqslant \zeta\,(3\,\delta_s+\varepsilon(n))+e(\mathcal{A}_{s\times[n]})+L\,\Big(\sqrt{\log\tfrac{1}{\gamma}/2n}+\sqrt{\tfrac{2m\log 2+2\log 1/\gamma}{n}}\Big)\end{aligned}$$

■

Corollary 3.2.2.1 *Generalization performance of the proxy-based methods can be limited by the maximum of distances between the proxies and the corresponding class samples in the dataset.*

Proof: The covering radius for each class subset is the maximum distance between the corresponding class samples and the class proxy. We at least know that the generalization error is bounded above with a term proportional to that distance. ■

Appendix B

PROOFS FOR CHAPTER 5

B.1 Proof for Claim 5.2.1

Proof: ρ^* is obtained as the solution of the following optimal transport problem:

$$\rho^*, \pi^* = \underset{\substack{\rho,\pi\geqslant 0\\ \rho_j+\Sigma_i\pi_{ij}={}^1\!/n\\ \Sigma_{ij}\pi_{ij}=\mu}}{\arg\min} \textstyle\sum_{ij} c_{ij}\pi_{ij}.$$

Given solutions (ρ^*, π^*), for μ=1, from the 3^{rd} constraint, we have $\Sigma_{ij}\pi^*_{ij}$=1. Then, using the 2^{nd} constraint, we write:

$$\textstyle\sum_j \rho^*_j + \sum_j \sum_i \pi^*_{ij} = \sum_j \frac{1}{n}$$

where $j\in[n]$ for n-many convolutional features. Hence, we have $\sum_j \rho^* = 0$ which implies ρ^*=0 owing to non-negativity constraint. Finally, pooling weights becomes $p_i = \frac{{}^1\!/n-\cancel{\rho^*_i}}{\underbrace{\mu}_{=1}} = {}^1\!/n$.

$\blacksquare$

B.2 Proof for Proposition 5.2.1

Before starting our proof, we first derive an iterative approach for the solution of (P2). We then prove that the iterations in Proposition 4.1 can be used to obtain the solution.

We can write (P2) equivalently as:

$$\rho^{(\varepsilon)}, \pi^{(\varepsilon)} = \operatorname*{arg\,min}_{\substack{\rho,\pi \geqslant 0\\ \rho_j + \Sigma_i \pi_{ij} = {}^1\!/_n\\ \Sigma_{ij}\pi_{ij} = \mu}} \textstyle\sum_{ij} c_{ij}\pi_{ij} + \frac{1}{\varepsilon}(\sum_{ij} \pi_{ij} \log \pi_{ij} + \sum_j \rho_j \log \rho_j)$$

$$+ \textstyle\sum_j 0\rho_j - \sum_{ij} \pi_{ij} - \sum_j \rho_j + \sum_{ij} \mathrm{e}^{\text{-}\varepsilon c_{ij}} + \sum_j \mathrm{e}^{\text{-}\varepsilon 0}$$

Rearranging the terms we get:

$$\rho^{(\varepsilon)}, \pi^{(\varepsilon)} = \operatorname*{arg\,min}_{\substack{\rho,\pi \geqslant 0\\ \rho_j + \Sigma_i \pi_{ij} = {}^1\!/_n\\ \Sigma_{ij}\pi_{ij} = \mu}} \textstyle\sum_{ij} \pi_{ij} \log \frac{\pi_{ij}}{\mathrm{e}^{\text{-}\varepsilon c_{ij}}} + \sum_j \rho_j \log \frac{\rho_j}{\mathrm{e}^{\text{-}\varepsilon 0}}$$

$$- \textstyle\sum_{ij} \pi_{ij} - \sum_j \rho_j + \sum_{ij} \mathrm{e}^{\text{-}\varepsilon c_{ij}} + \sum_j \mathrm{e}^{\text{-}\varepsilon 0}$$

which is generalized *Kullback–Leibler divergence* (KLD) between (ρ, π) and $(\exp(\text{-}\varepsilon 0), \exp(\text{-}\varepsilon c))$. Hence, we reformulate the problem as a KLD prjoection onto a convex set, which can be solved by *Bregman Projections* (*i.e.*,alternating projections onto constraint sets) [99, 100]. Defining $\mathcal{C}_1 \coloneqq \{(\rho, \pi) \mid \rho_j + \sum_{ij} \pi_{ij} = {}^1\!/_n \ \forall j\}$ and $\mathcal{C}_2 \coloneqq \{(\rho, \pi) \mid \sum_{ij} \pi_{ij} = \mu\}$, and omitting constants, we can write the problem as:

$$\rho^{(\varepsilon)}, \pi^{(\varepsilon)} = \operatorname*{arg\,min}_{\substack{\rho,\pi \geqslant 0\\ (\rho,\pi) \in \mathcal{C}_1 \cap \mathcal{C}_2}} \textstyle\sum_{ij} \pi_{ij}(\log \frac{\pi_{ij}}{\mathrm{e}^{\text{-}\varepsilon c_{ij}}} \text{ - } 1) + \sum_j \rho_j(\log \frac{\rho_j}{\mathrm{e}^{\text{-}\varepsilon 0}} \text{ - } 1) \tag{P2$'$}$$

Given, $(\rho^{(k)}, \pi^{(k)}))$, at iteration k, KLD projection onto $\mathcal{C}_1$, *i.e.*,$(\rho^{(k+1)}, \pi^{(k+1)}) \coloneqq \mathcal{P}^{KL}_{\mathcal{C}_1}(\rho^{(k)}, \pi^{(k)})$, reads:

$$\rho_j^{(k+1)} = {}^1\!/_n(\rho_j^{(k)} + \textstyle\sum_i \pi_{ij}^{(k)})^{\text{-}1} \rho_j^{(k)} \, ,$$

$$\pi^{(k+1)} = {}^1\!/_n(\rho_j^{(k)} + \textstyle\sum_i \pi_{ij}^{(k)})^{\text{-}1} \pi_{ij}^{(k)}$$

where the results follow from *method of Lagrange multipliers*. Similarly, for $\mathcal{P}^{KL}_{\mathcal{C}_2}(\rho^{(k)}, \pi^{(k)})$, we have:

$$\rho^{(k+1)} = \rho^{(k)} \, , \quad \pi^{(k+1)} = \frac{\mu}{\sum_{ij} \pi_{ij}^{(k)}} \pi^{(k)} \, .$$

Given initialization, $(\rho^{(0)}, \pi^{(0)}) = (\mathbf{1}_n, \exp(\text{-}\varepsilon c))$, we obtain the pairs $(\rho^{(k)}, \pi^{(k)})$ for $k = 0, 1, 2, \ldots$ as:

$$\begin{gathered} \rho^{(k+1)} = {}^1\!/_n(\rho^{(k)} + \pi^{(k)\intercal}\mathbf{1}_m)^{\text{-}1} \odot \rho^{(k)} \, , \quad \pi^{(k+1)} = \mu(\mathbf{1}_m^\intercal \hat{\pi} \mathbf{1}_n)^{\text{-}1} \hat{\pi} \\ \text{where} \quad \hat{\pi} = \pi^{(k)} Diag\Big({}^1\!/_n(\rho^{(k)} + \pi^{(k)\intercal}\mathbf{1}_m)^{\text{-}1}\Big) \end{gathered} \tag{B.2.1}$$

Proof: We will prove by induction. From Proposition 4.1, we have

$$\rho^{(k+1)} = {}^{1}\!/_{n}\,(1 + t^{(k)}\exp(\text{-}\varepsilon c)^{\intercal}\mathbf{1}_m)^{\text{-}1},\;\; t^{(k+1)} = \mu\,(\mathbf{1}_m^{\intercal}\exp(\text{-}\varepsilon c)\rho^{(k+1)})^{\text{-}1}$$

and $\pi^{(k)} = t^{(k)}\exp(\text{-}\varepsilon c)Diag(\rho^{(k)})$. It is easy to show that the expressions hold for the pair $(\rho^{(1)}, \pi^{(1)})$. Now, assuming that the expressions also holds for arbitrary $(\rho^{(k')}, \pi^{(k')})$. We have

$$\rho^{(k'+1)} = {}^{1}\!/_{n}(\rho^{(k')} + \pi^{(k')\intercal}\mathbf{1}_m)^{\text{-}1} \odot \rho^{(k')}$$

Replacing $\pi^{(k')} = t^{(k')}\exp(\text{-}\varepsilon c)Diag(\rho^{(k')})$ we get:

$$\rho^{(k'+1)} = {}^{1}\!/_{n}(\rho^{(k')} + Diag(\rho^{(k')})t^{(k')}\exp(\text{-}\varepsilon c)^{\intercal}\mathbf{1}_m)^{\text{-}1} \odot \rho^{(k')}$$

where $\rho^{(k')}$ terms cancel out, resulting in:

$$\rho^{(k'+1)} = {}^{1}\!/_{n}(1 + t^{(k')}\exp(\text{-}\varepsilon c)^{\intercal}\mathbf{1}_m)^{\text{-}1}.$$

Similarly, we express $\hat{\pi}$ as:

$$\hat{\pi} = t^{(k')}\exp(\text{-}\varepsilon c)Diag(\rho^{k'})Diag({}^{1}\!/_{n}(\rho^{(k')} + Diag(\rho^{(k')})t^{(k')}\exp(\text{-}\varepsilon c)^{\intercal}\mathbf{1}_m)^{\text{-}1})$$

again $\rho^{(k')}$ terms cancel out, resulting in:

$$\hat{\pi} = t^{(k')}\exp(\text{-}\varepsilon c)Diag({}^{1}\!/_{n}(1 + t^{(k')}\exp(\text{-}\varepsilon c)^{\intercal}\mathbf{1}_m)^{\text{-}1}) = t^{(k')}\exp(\text{-}\varepsilon c)Diag(\rho^{(k'+1)}).$$

Hence, $\pi^{(k'+1)}$ becomes:

$$\begin{aligned}\pi^{(k'+1)} &= \mu(\mathbf{1}_m^{\intercal}t^{(k')}\exp(\text{-}\varepsilon c)Diag(\rho^{(k'+1)})\mathbf{1}_n)^{\text{-}1}t^{(k')}\exp(\text{-}\varepsilon c)Diag(\rho^{(k'+1)})\\ &= \tfrac{1}{t^{(k')}}\underbrace{\mu(\mathbf{1}_m^{\intercal}\exp(\text{-}\varepsilon c)\rho^{(k'+1)})^{\text{-}1}}_{=t^{(k'+1)}}t^{(k')}\exp(\text{-}\varepsilon c)Diag(\rho^{(k'+1)})\\ &= t^{(k'+1)}\exp(\text{-}\varepsilon c)Diag(\rho^{(k'+1)}),\end{aligned}$$

meaning that the expressions also hold for the pair $(\rho^{(k'+1)}, \pi^{(k'+1)})$. ■

B.3 Proof for Proposition 5.2.2

Proof: We start our proof by expressing (P2′) in a compact form as:

$$x^{(\varepsilon)} = \underset{\substack{x \geqslant 0\\ Ax=b}}{\arg\min}\, \bar{c}^{\intercal}x + \tfrac{1}{\varepsilon}x^{\intercal}(\log x - \mathbf{1}_{(m+1)n})$$

where x denotes the vector formed by stacking ρ and the row vectors of π, $\bar{c}$ denotes the vector formed by stacking n-dimensional zero vector and the row vectors of c, and A and b are such that $Ax = b$ imposes transport constraints as:

$$A = \begin{bmatrix} I_{n \times n} & \overbrace{I_{n \times n} \quad \cdots \quad I_{n \times n}}^{m} \\ \mathbf{0}_n^\intercal & \mathbf{1}_{mn}^\intercal \end{bmatrix}, \quad b = [{}^1\!/_n \mathbf{1}_n^\intercal \quad \mu]$$

From *Lagrangian dual*, we have:

$$x^{(\varepsilon)} = \exp(\text{-}\varepsilon(\bar{c} + A^\intercal \lambda^*))$$

where λ^* is the optimal dual Lagrangian satisfying:

$$A \exp(\text{-}\varepsilon(\bar{c} + A^\intercal \lambda^*)) = b$$

Defining $[\frac{\partial x}{\partial c}]_{ij} := \frac{\partial x_j}{\partial c_i}$, we have;

$$\tfrac{\partial x^{(\varepsilon)}}{\partial c} = -\varepsilon \bar{I}(I + \tfrac{\partial \lambda^*}{\partial \bar{c}} A) Diag(x^{(\varepsilon)})$$

where $\bar{I} := [\mathbf{0}_{(mn) \times n} \quad I_{(mn) \times ((m+1)n)}]$. Similarly, for the dual variable, we have:

$$-\varepsilon(I + \tfrac{\partial \lambda^*}{\partial \bar{c}} A) Diag(x^{(\varepsilon)}) A^\intercal = 0 \Rightarrow \tfrac{\partial \lambda^*}{\partial \bar{c}} = Diag(x^{(\varepsilon)}) A^\intercal (A Diag(x^{(\varepsilon)}) A^\intercal)^{\text{-}1}.$$

Putting back the expression for $\frac{\partial \lambda^*}{\partial \bar{c}}$ in $\frac{\partial x^{(\varepsilon)}}{\partial c}$, we obtain:

$$\tfrac{\partial x^{(\varepsilon)}}{\partial c} = -\varepsilon \bar{I} \Big(Diag(x^{(\varepsilon)}) - Diag(x^{(\varepsilon)}) A^\intercal (A Diag(x^{(\varepsilon)}) A^\intercal)^{\text{-}1} A Diag(x^{(\varepsilon)}) \Big),$$

which includes $(m{+}1)$ by n matrix inversion, $H := A Diag(x^{(\varepsilon)}) A^\intercal$. We now show that $H^{\text{-}1}$ can be obtained without explicit matrix inversion.

H can be expressed as:

$$H = \begin{bmatrix} {}^1\!/_n I & {}^1\!/_n - \rho \\ {}^1\!/_n - \rho^\intercal & \mu \end{bmatrix}$$

H is Hermitian and positive definite. Using block matrix inversion formula for such matrices (Corrolary 4.1 of [123]), we obtain the inverse as;

$$H^{\text{-}1} = \begin{bmatrix} nI + k^{\text{-}1} \hat{\rho} \hat{\rho}^\intercal & -k^{\text{-}1} \hat{\rho} \\ -k^{\text{-}1} \hat{\rho}^\intercal & k^{\text{-}1} \end{bmatrix}$$

where $k = 1 - \mu - n\rho^{(\varepsilon)\intercal} \rho^{(\varepsilon)}$ and $\hat{\rho} = 1 - n\rho^{(\varepsilon)}$.

Given $\frac{\partial \mathcal{L}}{\partial x^{(\varepsilon)}}$, *i.e.*,$(\frac{\partial \mathcal{L}}{\partial \rho^{(\varepsilon)}}, \frac{\partial \mathcal{L}}{\partial \pi^{(\varepsilon)}})$, the rest of the proof to obtain $\frac{\partial \mathcal{L}}{\partial c}$ follows from right multiplying the Jacobian, *i.e.*,$\frac{\partial \mathcal{L}}{\partial c} = \frac{\partial x^{(\varepsilon)}}{\partial c} \frac{\partial \mathcal{L}}{\partial x^{(\varepsilon)}}$ and rearranging the terms. ■

VITA

Surname, Name · Date and Place of Birth · Nationality · Marital Status
Gürbüz, Yeti Z. · 10 December 1988, Sivas · Turkish (TC) · Married
Phone: 00905333526155 · *e-mail:* yeti@metu.edu.tr

Education

MS, Electrical and Electronics Engineering	September 2014
Middle East Technical University (METU)	*Ankara, Turkey*
BS, Electrical and Electronics Engineering	June 2011
Middle East Technical University (METU)	*Ankara, Turkey*

Work Experience

Research Scientist	April 2021 - Present
Center for Image Analysis, METU	*Ankara, Turkey*
Research and Teaching Assistant	September 2011 - April 2021
Dept. of Elect. and Elec. Eng., METU	*Ankara, Turkey*

Publications

1. Görgün, Ada, Yeti Z. Gürbüz, and A. Aydın Alatan. "Feature Embedding by Template Matching as a ResNet Blocks." *BMVC, 2022 (submitted).*

2. Gürbüz, Yeti Z., Ozan Şener, and A. Aydın Alatan. "Generalized Sum Pooling for Metric Learning." *NeurIPS, 2022 (submitted).*

3. Gürbüz, Yeti Z., Oğul Can, and A. Aydın Alatan. "Deep Metric Learning with Chance Constraints." *TNNLS, 2022 (submitted).*

4. Can, Oğul, Yeti Z. Gürbüz, and A. Aydın Alatan. "Deep Metric Learning with Alternating Projections onto Feasible Sets." *ICIP, 2021.*

5. Can, Oğul, Yeti Z. Gürbüz, Berkin Yıldırım, and A. Aydın Alatan. "Blind Deinterleaving of Signals in Time Series with Self-attention Based Soft Min-cost Flow Learning." *ICASSP, 2021.*

6. Gürbüz, Yeti Z., and A. Aydın Alatan. "A Novel BoVW Mimicking End-To-End Trainable CNN Classification Framework Using Optimal Transport Theory." *ICIP, 2019.*

7. Gürbüz, Yeti Z.,Oğul Can, and A. Aydın Alatan. "Roadesic Distance: Flow-aware Tracklet Association Cost for Wide Area Surveillance." *ICIP, 2017.*

8. Can, Oğul, Yeti Z. Gürbüz, and A. Aydın Alatan. "Automatic Road Detection from Gray-level Images in Wide Area Surveillance." *IGARSS, 2016.*

9. Gürbüz, Yeti Z., Cevahir Çığla, and A. Aydın Alatan. "Sparse Recursive Filtering for O (1) Stereo Matching." *ICIP, 2015.*

Hobbies

Hip Hop Culture Dance Styles