Master's Thesis in Graduate School of Library, Information and Media Studies

Rank-sensitive Deep Metric Learning

March 2021

201921648

Muramoto Naoki

Rank-sensitive Deep Metric Learning
ランク考慮型深層距離学習

Student No.: 201921648
氏名：村本　尚生
Name: Muramoto Naoki

Deep metric learning has shown significantly increasing values in a wide range of domains, such as image retrieval, face recognition, zero-shot learning, to name a few. When evaluating the methods for deep metric learning, *top-k precision* is commonly used as a key metric, since few users bother to scroll down to lower-ranked items. Despite being widely studied, how to directly optimize top-k precision is still an open problem. In this thesis, we proposed novel methods on how to optimize top-k precision in a rank-sensitive manner for deep metric learning. Given the cutoff value k, our key idea is to impose different weights to further differentiate misplaced images sampled according to the top-k precision.

To validate the effectiveness of the proposed methods, we conducted a series of experiments on three widely used benchmark datasets with discrete labels and one benchmark dataset with continuous labels. The experimental results on datasets with discrete labels demonstrate that: our proposed method (RS-TopK-Pre) outperforms all baseline methods on two datasets, namely CUB200-2011, Cars196, which shows the potential value of rank-sensitive optimization of top-k precision for deep metric learning. The experimental results on the dataset with continuous labels demonstrate that: our proposed method (RS-TopK-Pre) outperforms the baseline method (TopK-Pre). In view of the unique property of continuous labels, we further modified RS-TopK-Pre by imposing different weights to further differentiate misplaced images based on continuous rank information. We refer to this variant as CRS-TopK-Pre. Unfortunately, CRS-TopK-Pre failed to achieve better performance than the baseline method (TopK-Pre).

Overall, both the results on datasets with discrete labels and the results on the dataset with continuous labels show that: the factors, such as batch size and cutoff value k, significantly affect the performance of approaches that rely on optimizing top-k precision for deep metric learning. Careful examinations of these factors are thus recommended.

Principal Academic Advisor: Joho Hideo
Secondary Academic Advisor: Yu Hai-Tao

Rank-sensitive Deep Metric Learning

Muramoto Naoki

Graduate School of Library,
Information and Media Studies

University of Tsukuba

March 2021

Contents

List of Figures

Chapter 1

Introduction

The concept of machine learning, which allows for computers to learn without being directly programmed, emerged after computers gained the ability to recognize objects [1]. Machine learning succeeded in many fields, such as object recognition, medical diagnosis, face recognition and text mining.

K-nearest neighbour (KNN), Support Vector Machines (SVM), and Naive Bayes classifiers are typical machine learning algorithms. Although these algorithms have a certain classification performance, it may not always be possible to expect them to produce the desired results in any problems, since each dataset has its own properties [2]. These algorithms do not transform an original dataset to a new space. The effect of each feature is not equal in terms of classification. Therefore, feature weighting might be used before classification. The dataset can be transformed from original space to a new space. To realize this, data transformation algorithms, like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), have been proposed [2].

Metric learning is a technique that can transform an original dataset to a new space. It is based on a distance metric that aims to establish similarity or dissimilarity between objects. Although original ideas can be traced back to some earlier works (e.g., [3, 4, 5, 6, 7]), metric learning emerged in 2002 with the pioneering work of Xing et al. [8] that formulated it as a convex optimization problem [9]. Metric learning has been intensively studied and has shown significantly increasing values in a wide range of domains, such as image retrieval, face recognition, zero-shot learning [10]. In order to capture the important relationships among instances, the objective of metric learning is to learn an embedding space, where the embedded vectors corresponding to similar instances are expected to be closer, while the vectors corresponding to dissimilar instances are pushed apart from each other [2].

There are two approaches to metric learning, depending on the type of transformation. One is a linear method, the other is a nonlinear method. Linear metric learning approaches have some advantages. They are more convenient to optimize (in particular, it is easier to derive convex formulations with the guarantee of finding the global optimum) and less prone to overfitting [9]. However, they also have some disadvantages. Since they have poor performance to capture nonlinear feature structure, linear transformations have a limited ability to achieve optimal performance over the new representation of data [2]. In order to solve this problem, kernel methods that are nonlinear metric learning methods are proposed [11]. Although these nonlinear approaches are practical to solve nonlinear problems, they may be prone to overfitting [2].

In recent years, thanks to the development of GPU technology, the power of computing has been increasing day by day. Therefore, it is possible to develop fast and successful solutions in machine learning using big data. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [12]. Deep learning, which provides a new representation of the raw data, obtains the automatic extraction of features where the goal is to achieve higher abstraction levels when transforming data [12, 13].

Recently, deep learning and metric learning have been brought together to introduce the concept of deep metric learning [14]. Based on the study by Lu et al. [14], we show the concept of deep metric learning in Fig. 1.1. The purpose of deep metric learning is to bring the same samples from the same classes closer to each other, and pushing the samples from different classes apart from each other (Fig. 1.1(a)). Fig. 1.1(b) shows the example of deep metric learning.

Deep metric learning has received much attention, since deep metric learning methods can solve problems that exist in both linear and nonlinear methods. Deep metric learning is a metric learning approach using deep neural networks. Since the input data is transformed by a neural network to a vector representation, deep metric learning is essentially a vector embedding problem [15].

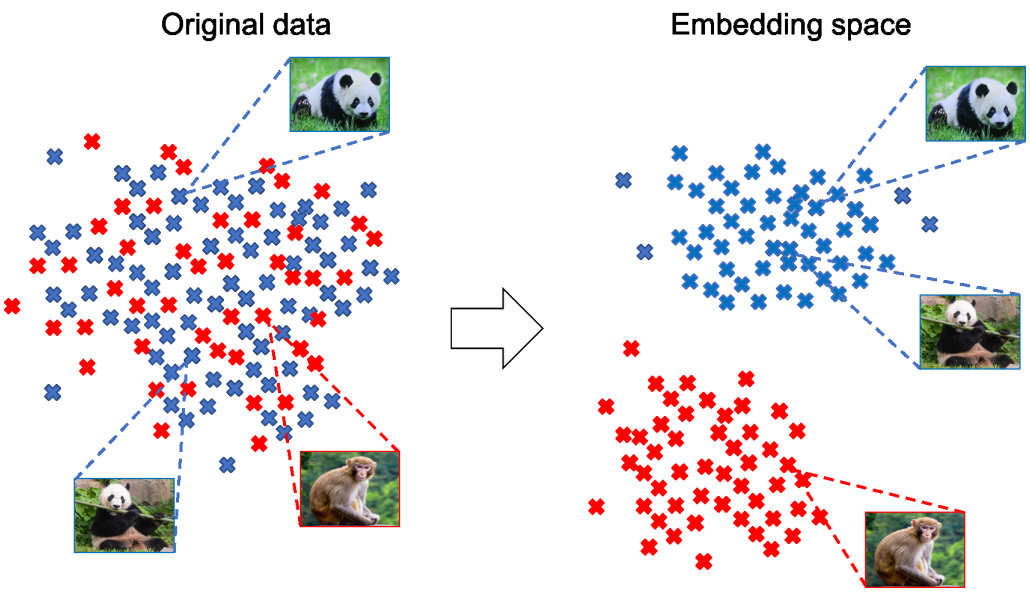
Despite the success achieved by the deep metric learning approaches, there are still many open issues. When evaluating the methods for deep metric learning, top-k precision is commonly used as a key metric, since few users bother to scroll down to lower-ranked items. Despite being studied actively, most of the existing training losses or sampling strategies are based on heuristic observations instead of theoretical analysis. Thus, the gradient descent on the training loss is mostly inconsistent with the direction of optimizing the concerned evaluation metric [16].

1.1 Contributions

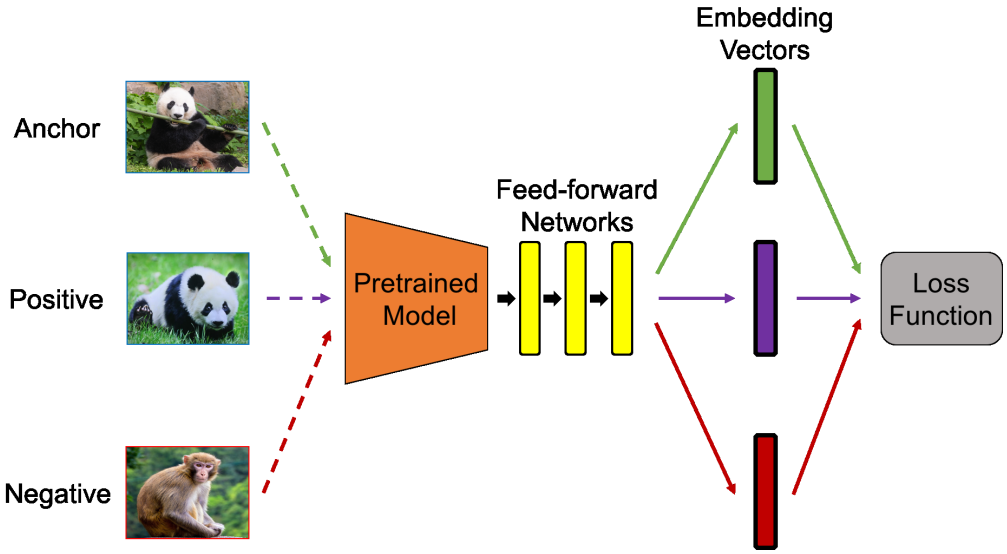
In this thesis, we provide novel solutions for deep metric learning.

To start with, we investigate what extent incorporating rank information helps to improve the optimization of top-k precision. To validate the effectiveness of the proposed method, we conduct a series of experiments on three widely used benchmark datasets with discrete labels. The experimental results demonstrate that: (1) Our proposed method (RS-TopK-Pre) outperforms all baseline methods on two datasets, which shows the potential value of rank-sensitive optimization of top-k precision for deep metric learning. (2) The factors, such as batch size and cutoff value k, significantly affect the performance of approaches that rely on optimizing top-k precision for deep metric learning. Careful examinations of these factors are highly recommended. This work has been published in [17].

Next, we investigate whether our proposed method is effective on datasets with continuous labels. We adjust the proposed method (RS-TopK-Pre) to cope with continuous labels instead of being limited to datasets with discrete labels. We conduct a series of experiments on one widely used benchmark dataset with continuous labels. The experimental results demonstrate that: our proposed method (RS-TopK-Pre) outperforms the baseline method (TopK-Pre). It shows the potential value of rank-sensitive optimization of top-k precision for deep metric learning with continuous labels. In view of the unique property of con-



(a) Purpose of deep metric learning.



(b) Deep metric learning example.

Figure 1.1: Concept of metric learning.

tinuous labels, we further modified RS-TopK-Pre by imposing different weights to further differentiate misplaced images based on continuous rank information. We refer to this variant as CRS-TopK-Pre. Unfortunately, CRS-TopK-Pre failed to achieve better performance than the baseline method (TopK-Pre).

1.2 Roadmap of Thesis

The rest of the thesis is organized as follows. In Chapter 2, we review related works of metric learning, pretrained models and learning to rank. In Chapter 3, we describe our proposed methods for the problem of deep metric learning. We firstly discuss the motivation, then we elaborate the details of our proposed methods. In Chapter 4, we investigate to what extent incorporating rank information helps to improve the optimization of top-k precision. We conduct a series of experiments on datasets with both discrete and continuous labels. In Chapter 5, we summarize our contributions and discuss the possible directions to improve the proposed models as a future work.

Chapter 2

Related Work

In this chapter, we review related work. First, we introduce metric learning. Then, we describe pretrained models. Finally we explain learning to rank.

2.1 Metric Learning

In this section, we provide a brief overview of metric learning by classifying the related methods into two groups, namely, traditional metric learning and deep metric learning.

2.1.1 Traditional Metric Learning

Most of the work in metric learning has focused on the Mahalanobis distance. Let $Z = \{z_1, ..., z_m\} \in \mathbb{R}^{d \times m}$ be the training samples, where $z_i \in \mathbb{R}^d$ is i-th training example and m is the total number of training samples. The Mahalanobis distance between z_i and z_j is calculated as:

$$d_M(z_i, z_j) = \sqrt{(z_i - z_j)^{\mathrm{T}} M (z_i - z_j)} \tag{2.1}$$

Most methods learn the metric (here, the positive semi-definite matrix M in d_M) in a weakly-supervised way from pair or triplet-based constraints [9]. The pair-based constraint is: in positive pairs (z_a, z_p), z_a and z_p should be similar. And in negative pairs (z_a, z_n), z_a and z_n should be dissimilar. The triplet-based constraint is: in triplets (z_a, z_p, z_n), z_p should be more similar to z_a than to z_n. The aim of typical metric learning is to find the best parameters for these constraints in order to approximate the underlying semantic metric.

There are two types of metric learning approach, depending on the type of transformation. One is a linear method, the other is a nonlinear method. Most of the work in linear metric learning has focused on Mahalanobis distance [9]. Despite that, there are some works that focus on other similarity functions. [18, 19] focused on cosine similarity, widely used in text and image retrieval (see for instance [20, 21]). And, [22, 23] learn a biliner similarity, which has been used for instance in image retrieval [24].

Since linear metrics are more convenient to optimize, metric learning has focused on linear metrics. In some cases, however, there are nonlinear structures in the data that linear metrics are unable to capture [9]. To capture nonlinear structure, the kernelization of linear methods have been proposed. The idea of kernelization is to learn a linear metric in the nonlinear feature space induced by a kernel function and thereby combine the best of both worlds, like in SVM [9]. Several general karnelization methods based on Kernel

PCA [25] and a nonlinear extension of PCA [26] have been proposed [27, 28]. There are a few approaches that learn nonlinear forms of metrics directly [29, 30, 31, 32, 33]. Although these approaches may significantly outperform linear approaches on nonlinear problems, they may be prone to overfitting, and thereby lead to poor generalization power.

2.1.2 Deep Metric Learning

The performance of traditional metric learning is limited, since linear metric learning methods have poor performance to capture nonlinear feature structures, and nonlinear metric learning methods have the negative impact on overfitting. Unlike traditional metric learning, deep metric learning methods solve the problems that exist in both methods. Below we introduce deep metric learning.

Contrastive loss [29] is a representative pair-based metric learning approach that uses Siamese Network [34]. It receives pair images, including positive and negative samples, and learns an embedding space, a positive pair to be closer, and a negative pair apart from each other. This approach is a study that provides inspiration for the field of deep metric learning [2]. In [35], they combine the Siamese Network and Convolutional Neural Network, which enables learning a similarity metric from image pixels directly. Triplet network inspired by Siamese Network contains three objects, which form positive, negative and anchor samples [36]. Triplet networks utilize Euclidean space to compare the objects in the pattern recognition process, and this approach is directly related to metric learning [2].

Schroff et al. [37] introduced a FaceNet deep model that learns a mapping from the original sample space to a compact Euclidean space. Once this space is produced, face recognition and clustering can be easily implemented under the network with FaceNet embeddings as feature vectors. FaceNet uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches.

Song et al. [38] introduced a deep metric learning method via lifted structured feature embedding. Their algorithm took a full advantage of the training batches in the neural network training stage by lifting the vector of pairwise distances within the batch to the matrix of pairwise distances. This step enables the algorithm to learn the state of the art feature embedding by optimizing a new structured prediction objective on the lifted problem [14].

Recently, due to the breakthrough successes of neural networks, deep metric learning has attracted increased attention, and many approaches have been proposed. The representative examples include n-pair loss [39], clustering loss [40], angular loss [41], histogram loss [42], to name a few.

Despite the success achieved by the deep metric learning approaches, there are still many open issues. For example, most of the existing training loss functions or sampling strategies are based on heuristic observations rather than theoretical analysis. Consequently, gradient descent direction on the training loss is mostly inconsistent with the direction of optimizing the concerned evaluation metric [16].

Two recent representative works address deep metric learning by optimizing concerned evaluation metric [16, 43]. Lu et al. [16] proposed to perform metric learning by optimizing a training loss which is equivalent to maximising the top-k precision. In this thesis, this

method is referred to as TopK-Pre. Cakir et al. [43] proposed to perform metric learning by optimizing average precision (AP).

Our work is closely related to TopK-Pre. Though TopK-Pre leads to promising performance, the similarity scores within the training loss are treated equally. To overcome this drawback, we propose to optimize top-k precision in a rank-sensitive manner.

2.2 Pretrained Models

In deep metric learning, metric learning loss functions follow the deep neural network models that extract feature embeddings. Recent deep metric learning models are able to use pretrained deep neural network models as the backbone networks, and finetune the network architecture for each specific task.

In the experiments of Section 4.1, we use ResNet50 [44] initialized by ImageNet [45] pretraining as the network backbone. We use ResNet50 due to its success in recent state-of-the-art approaches [46, 47] with discrete labels. ResNet50 means ResNets of 50 layers. In the experiments of Section 4.2, we use ResNet34 initialized by ImageNet pretraining as the network backbone. We use ResNet34 due to its success in recent state-of-the-art approaches [48] with continuous lables. ResNet34 means ResNets of 34 layers.

ResNets present a residual learning framework to ease the training of deep networks. They explicitly reformulate the layers as learning residual functions with reference to inputs of the layer, instead of learning unreferenced functions. These residual networks are easier to optimize, and can gain accuracy from considerably increased depth. ResNets won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation at ILSVRC[1] & COCO[2] 2015 competitions [44].

2.3 Learning to Rank

Learning to rank refers to a broad range of approaches that aim to tackle ranking problems using machine learning techniques [49]. It is used in multiple fields, such as document retrieval, question answering and recommender systems, where ranking lies in the core. In the representative work [50], major learning to rank approaches can be classified into three categories: pointwise, pairwise, and listwise approaches.

2.3.1 Pointwise Methods

Pointwise is the simplest approach to learning to rank. The input is a document. And the model outputs predicted relevance score to the query. The typical pointwise approaches include regression-based [51], classification-based [52], and ordinal regression-based algorithms [53, 54]. The loss functions of these algorithms are defined on the basis of each individual document [49].

[1] http://image-net.org/challenges/LSVRC/2015/
[2] http://mscoco.org/dataset/#detections-challenge2015

2.3.2 Pairwise Methods

The input of *pairwise* approaches is a pair of documents. The model cares about which document is more relevant to the query. The goal of learning is to maximize the number of correctly ordered document pairs [49]. With the assumption, if all the document pairs are correctly ordered the optimal ranking of documents can be achieved. Towards this end, many representative methods have been proposed [55, 56, 57, 58, 59].

2.3.3 Listwise Methods

The input of *listwise* is a list of all the documents associated with the same query in the training data. The model needs to optimize the total ordering of such list of documents according to relevance [15]. In particular, there are two types of loss functions in listwise learning to rank. For the first type, the loss function is related to a specific listwise evaluation metric (e.g., nDCG [60] and ERR [61]). Since the commonly used listwise evaluation metrics are non-differentiable and non-decomposable, and the methods of this type either try to optimize the upper bounds as surrogate objective functions [62, 63, 64] or approximate the target metric using some smooth functions [65, 66, 67]. However, the first type methods still have some open issues. On the one hand, some of the surrogate functions or approximation metrics are not convex, which makes optimization more hard. On the other hand, the relationship between the surrogate function and the adopted metric has not been sufficiently investigated, which makes it unclear whether optimizing the surrogate functions can indeed optimize the target metric [49]. For the second type, the loss function is not explicitly related to a specific evaluation metric. The loss function is related to the discrepancy between the list of predicted ranking and the list of ground-truth ranking. Example algorithms include [68, 69, 70]. Although no particular evaluation metrics are directly involved and optimized here, it is possible that the learned ranking function perform well in terms of evaluation metrics [49].

2.3.4 Connection to Metric Learning

When we look at deep metric learning from the perspective of nearest neighbour retrieval, we can treat it as ranking problem. Therefore, learning to rank can be used as a deep metric learning problem. The scoring function in learning to rank can also be induced from a distance metric: a low distance between the query and document indicates high relevance, and vice versa [15].

Learning to rank formulations for metric learning have been discussed in [71, 72, 73]. Contrastive loss that uses Siamese network can be viewed as a pointwise approach, in the sense that an instance pair can be viewed as consisting of a query and a document, and the loss function encourages the distances between the query and relevant documents to be low, and high otherwise [15]. Similarly, triplet-based loss functions used in recent deep metric learning approaches [38, 74, 75, 76] can be viewed as pairwise ranking losses, in that the query is constrained to have lower distance to a relevant document than the other irrelevant document [15]. TopK-Pre which we referred to can be viewed as a listwise approach, in that the query is constrained to have a lower distance to a relevant image than the other irrelevant images according to the top-k precision. Inspired by TopK-Pre, we proposed novel

listwise deep metric learning approaches which optimize top-k precision in a rank-sensitive manner.

To compare our proposed method to other listwise approaches, we directly applied to two listwise learning to rank approaches, Lambdarank and ListNet. In the deep metric learning settings, the distance metrics are used as the scoring functions. A low distance between the two samples indicates a high relevance score, and vice versa.

This is the first attempt that directly adopts learning to rank approaches to deep metric learning problems.

Chapter 3

Rank-Sensitive Optimization of Top-K Precision

In this chapter, we detail the proposed methods for deep metric learning. Specifically, we describe the motivation in Section 3.1. In Section 3.2, we elaborate the details of RS-TopK-Pre.

3.1 Motivation

Metric learning has been intensively studied and has shown significantly increasing values in a wide range of domains. In order to capture the important relationships among instances, the objective of metric learning is to learn an embedding space, where the embedded vectors corresponding to similar instances are expected to be closer, while the vectors corresponding to dissimilar instances are pushed apart from each other [2]. Recently, due to the breakthrough successes of neural networks, deep metric learning has attracted increased attention, and many approaches have been proposed. The representative examples include contrastive loss [29], triplet loss [77], lifted loss [38], n-pair loss [39], clustering loss [40], angular loss [41], histogram loss [42]. These approaches have explored different aspects in order to maximize the within-class similarities as well as minimize the between-class similarities, such as loss designing, model ensemble, and improved sampling.

Despite the success achieved by the aforementioned approaches for metric learning, there are still many open issues. When evaluating the methods for deep metric learning, top-k precision is commonly used as a key metric, since few users bother to scroll down to lower-ranked items. Inspired by this, Lu et al. [16] proposed to perform metric learning by optimizing a training loss which is equivalent to maximising the top-k precision (TopK-Pre). The key idea underlying TopK-Pre is as follows: given a mini-batch of training images, they sample misplaced images according to the top-k precision by equally treating each image as the query image and the remaining images as candidate images. Though TopK-Pre leads to promising performance, the similarity scores within the training loss are treated equally. Intuitively, we wish to use a higher weight to penalize the misplaced negative images ranked at top positions and the misplaced positive images ranked at lower positions. As a result, TopK-Pre fails to further differentiate the misplaced images during the training process. To overcome this drawback, we propose to optimize top-k precision in a rank-sensitive manner. Our key idea is to *impose different weights to the similarity scores between the query image*

and the misplaced images sampled according to top-k precision.

3.2 Proposed Methods

In this section, we first review the TopK-Pre approach, then we show how to perform rank-sensitive optimization of top-k precision. Finally, we show how to adjust the proposed method to cope with continuous labels, and propose to optimize top-k precision in a continuous rank sensitive manner.

3.2.1 Review of TopK-Pre

Let $Z = \{z_1, ..., z_m\}$ be the training set of images. The goal of deep metric learning is to learn a nonlinear convolutional neural network (CNN) based function $f(\cdot)$ that maps an image z into a compact feature vector $f(z) \in \mathbb{R}^d$ while preserving the semantic similarities. The outputs of $f(\cdot)$ are L2-normalized. Namely, semantically similar images should have a higher similarity score than semantically dissimilar ones. In this thesis, we adopt the cosine similarity, $s_{i,j} = \frac{f(z_i)^\top f(z_j)}{\|f(z_i)\|\|f(z_j)\|}$ as the similarity score between image z_i and image z_j. Most models for deep metric learning are trained with the stochastic gradient descent algorithm and only a mini-batch of images are available at each iteration. The motivations under TopK-Pre are as follows: (1) Very few users bother to scroll down to lower-ranked items. The example contexts include visual product search and face verification. (2) top-k precision is not only the most widely used evaluation metric for embedding quality, but also closely related to the user experience. Given the query image (or anchor image) z_a and the set of candidate images $C = \{z_1, ..., z_m\}$ in a mini-batch, the embedded vectors are obtained as $f(z_a)$, $f(z_1)$, ..., $f(z_m)$, respectively. We denote the similarity scores between z_a and the candidate images as $\mathbf{s} \in \mathbb{R}^m$, where s_i represents the similarity score between z_a and the i-th candidate z_i. Let $\mathbf{y} \in \mathcal{Y} = \{0, 1\}^m$ be the ground-truth label vector that $y_i = 1$ iff z_a and z_i have the same class label, the top-k precision is defined as:

$$Pre@k(\mathbf{s}, \mathbf{y}) = \frac{\sum_1^m y_i \mathbb{I}_{[s_i \geq s_{[k]}]}}{k} \tag{3.1}$$

where $s_{[k]}$ denotes the k-th largest element within vector $\mathbf{s}$, and $\mathbb{I}_{[A]}$ is an indicator with the value of 1 if A is true, and 0 otherwise.

Because directly optimizing top-k precision is quite challenging, they propose an alternative loss, which is equivalent to maximizing top-k precision (please refer to [16] for the theoretical demonstration). This alternative loss is computed as follows: (1) For negative candidate images, they add a margin to the similarity score as:

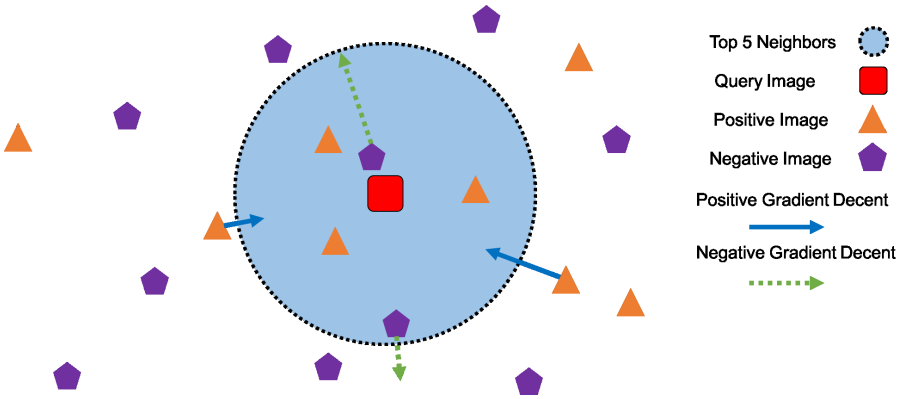
$$\hat{\mathbf{s}} = \mathbf{s} + \gamma(1 - \boldsymbol{y}) \tag{3.2}$$

where γ is the margin; (2) They sort the similarity scores with the margin added in a decreasing order; (3) From the ordered images, they first select all misplaced negative images within the top-k positions as $\mathcal{N}$. Next, they select the same number of positive images ranked below k from top to bottom, which is denoted as $\mathcal{P}$; (4) Finally, the training loss is defined as:

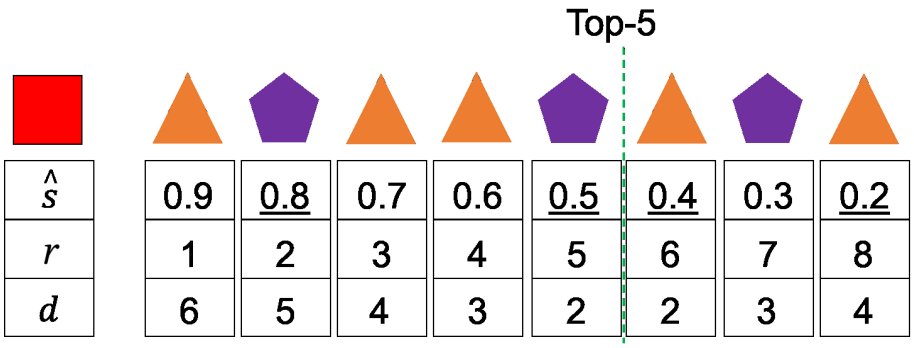
$$\ell_{TopK-Pre}(\mathbf{s}, \mathbf{y}) = \sum_{z_i \in \mathcal{N}} \hat{s}_i - \sum_{z_i \in \mathcal{P}} \hat{s}_i \tag{3.3}$$

According to [16], TopK-Pre achieves the state-of-the-art result for deep metric learning when compared with all the popular methods mentioned in Section 3.1.

3.2.2 RS-TopK-Pre



(a) Misplaced images according to top-5 precision, where the arrow length indicates the gradient power.



$\hat{s}$	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
r	1	2	3	4	5	6	7	8
d	6	5	4	3	2	2	3	4

(b) Illustration of RS-TopK-Pre with $k = 5$ and $|\mathcal{N}| = 2$. Rank-sensitive weights (i.e., $1 - \frac{1}{d_i}$) for misplaced negative images and positive images are 0.8, 0.5, 0.5 and 0.75, respectively.

Figure 3.1: Rank-sensitive optimization of top-k precision.

A closer look at Eq-3.3 reveals that the similarity values are treated equally. As a result, TopK-Pre fails to further differentiate the misplaced images sampled according to top-k precision during the training process. As an illustration, Fig. 3.1(a) depicts the misplaced images according to top-5 precision. Taking the two misplaced negative images for example, instead of treating them equally, it is intuitive to impose a higher push-force on the negative image nearby the query image compared with the other one. Inspired by this observation, we propose to optimize the top-k precision in a rank-sensitive manner, which is referred to as RS-TopK-Pre. The key idea is to impose different weights to the similarity values in Eq-3.3 based on the rank information. Specifically, given the cutoff value k, we associate each misplaced image z_i (either $z_i \in \mathcal{N}$ or $z_i \in \mathcal{P}$) with a distance d_i, which is defined as:

$$d_i = \begin{cases} k + 2 - r_i & if\ z_i \in \mathcal{N} \\ r_i + 1 - k & if\ z_i \in \mathcal{P} \end{cases} \tag{3.4}$$

where r_i is the rank position of the similarity score s_i within the sorted similarity scores. In other words, the larger the distance d_i is, the worse the sampled image z_i is misplaced.

Furthermore, we use the coefficient $1-\frac{1}{d_i}$ to weight the similarity scores, which is given as:

$$1-\frac{1}{d_i} \in \begin{cases} \left[\frac{1}{2}, 1-\frac{1}{k+1}\right] & if\ z_i \in \mathcal{N} \\ \left[\frac{1}{2}, 1-\frac{1}{b+1-k}\right] & if\ z_i \in \mathcal{P} \end{cases} \tag{3.5}$$

where b is the batch size. Thanks to this, we can further differentiate the misplaced images when optimizing top-k precision. Fig. 3.1(b) shows the example weights with top-5 precision. Finally, the rank-sensitive loss function for optimizing top-k precision is given as

$$\ell_{RS-TopK-Pre}(\mathbf{s},\mathbf{y}) = \frac{3}{|\mathcal{N}|}\left\{\sum_{z_i\in\mathcal{N}}(1-\frac{1}{d_i})\cdot\hat{s}_i - \sum_{z_i\in\mathcal{P}}(1-\frac{1}{d_i})\cdot\hat{s}_i)\right\} \tag{3.6}$$

The proposed training loss is also equivalent to maximising top-k precision. We omit the theoretical demonstration, which is straightforward by following the work [16]. The algorithm is summarized in Algorithm 1.

Algorithm 1 RS-TopK-Pre

Input: mapping function $f(\cdot)$; training dataset $\mathcal{Z}$;

for each $t \in \{1, ..., N\}$ **do** ▷ where N is the number of epochs
  for each $Z \in \mathcal{Z}$ **do** ▷ where $Z = \{z_1, ..., z_{m+1}\}$
    Calculate embeddings $f(z_1)$,..., $f(z_{m+1})$.
    for each $j \in \{1, ..., m+1\}$ **do**
      Assign z_j as the query image z_a, other m images as candidate set C.
      Sample $(\mathcal{P}, \mathcal{N})$.
      Calculate $[\ell_{RS-TopK-Pre}]_j$ as Eq-3.6.
    end for
    Sum up $\ell_{RS-TopK-Pre} = \sum_j^{m+1}[\ell_{RS-TopK-Pre}]_j$.
    Backpropagate the gradient, update $f(\cdot)$.
  end for
end for

3.2.3 TopK-Pre and RS-TopK-Pre for Continuous Labels

TopK-Pre and RS-TopK-Pre are designed for the datasets with discrete labels. To use them for the dataset with continuous labels, we show how to adjust the proposed method to cope with continuous labels.

Let $Z = \{z_1, \ldots, z_m\}$ be the training set of images. Given the query image (or anchor image) z_a and the set of candidate images $C = \{z_1, \ldots, z_m\}$ in a mini-batch, the embedding vectors are obtained as $f(z_a), f(z_1), \ldots, f(z_m)$, respectively. We denote the similarity scores (squared Euclidean distances) between z_a and candidate images as $\mathbf{s} \in \mathbb{R}^n$, where s_i represents the similarity score between z_a and i-th candidate z_i. We compute the score $s_i = \|f(z_a) - f(z_i)\|_2^2$. Because of continuous labels, to calculate top-k precision we use alternative binary labels, which are labeled as: top-N nearest neighbours are positive, others are negative. Let $\mathbf{y} \in \mathcal{Y} = \{0,1\}^m$ be the ground-truth alternative label vector, that $y_i = 1$ iff z_a and z_i are the top-k nearest neighbours, the top-k precision is defined as:

$$Pre@k(\mathbf{s},\mathbf{y}) = \frac{\sum_1^m y_i \mathbb{I}_{[s_i \geq s_{[k]}]}}{k} \tag{3.7}$$

where $s_{[k]}$ denotes the k-th largest element within vector s, and $\mathbb{I}_{|A|}$ is an indicator with the value of 1 if A is true, and 0 otherwise.

The loss functions of TopK-Pre and RS-TopK-Pre are the same as Section 3.2.

3.2.4 CRS-TopK-Pre

TopK-Pre and RS-TopK-Pre were designed for the datasets with discrete labels, where the images cannot be ordered within positive (negative) images. For continuous labels, it can be ordered within positive images. Inspired by this observation, we propose to optimize the top-k precision in a continuous rank-sensitive manner, which is referred to as CRS-TopK-Pre.

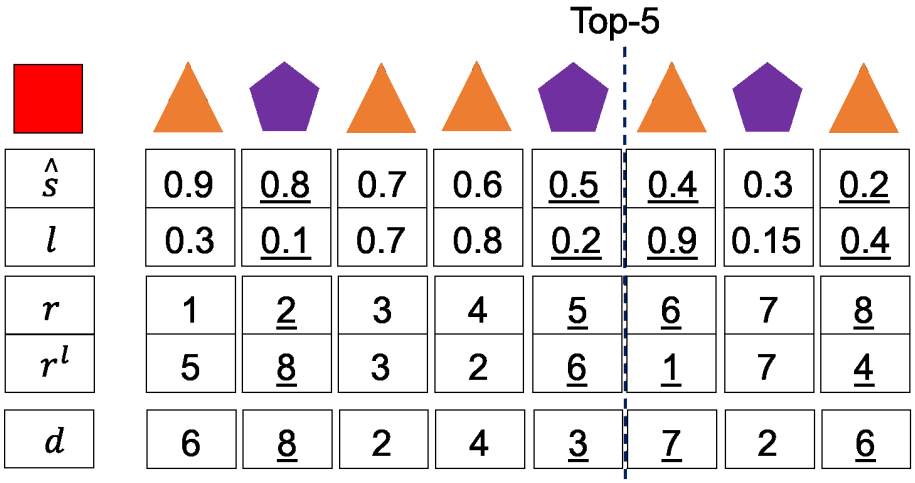


Figure 3.2: Illustration of CRS-TopK-Pre with $k = 5$ and $|\mathcal{N}| = 2$. Continuous rank-sensitive weights (i.e., $1 - \frac{1}{d_i}$) for misplaced negative images and positive images are $\frac{7}{8}$, $\frac{2}{3}$, $\frac{6}{7}$ and $\frac{5}{6}$, respectively.

The key idea is to impose different weights in the similarity value in Eq-3.3 based on the continuous rank information. Specifically, given the cutoff value k, we associate each misplaced image z_i (either $z_i \in \mathcal{N}$ or $z_i \in \mathcal{P}$) with a distance d_i, which is defined as:

$$d_i = \left| r_i^l - r_i \right| + 2 \tag{3.8}$$

where r_i is the rank position of the similarity score s_i within the sorted similarity scores. r_i^l is the rank of continuous labels. Furthermore, we use the coefficient $1 - \frac{1}{d_i}$ to weight the similarity scores, which is given as:

$$1 - \frac{1}{d_i} \in \left[\frac{1}{2}, 1 - \frac{1}{b+1} \right] \tag{3.9}$$

where b is the batch size. Thanks to this, we can further differentiate the misplaced images when optimizing top-k precision. Fig. 3.2 shows the example weights with top-5 precision. Finally, the continuous rank-sensitive loss function for optimizing top-k precision is given as

$$\ell_{CRS-TopK-Pre}(\mathbf{s}, \mathbf{y}) = \frac{3}{|\mathcal{N}|} \left\{ \sum_{z_i \in \mathcal{N}} (1 - \frac{1}{d_i}) \cdot \hat{s}_i - \sum_{z_i \in \mathcal{P}} (1 - \frac{1}{d_i}) \cdot \hat{s}_i) \right\} \tag{3.10}$$

Chapter 4

Experiments

In this chapter, we conduct a series of experiments on datasets with discrete and continuous labels, in order to investigate to what extent incorporating rank information helps to improve the optimization of top-k precision. Following the previous studies on deep metric learning [16, 38, 39, 40, 41, 42, 43, 48, 78], we evaluate the involved methods from the viewpoint of nearest neighbour retrieval.

4.1 Evaluation Using Discrete Labels

In this section, we evaluate whether our proposed methods are effective on the datasets with discrete labels. We first describe the test collections and evaluation metrics with discrete labels. We then describe the configuration of each method, including the parameter and the way of training.

4.1.1 Datasets

In our experiments, we use three widely used benchmark datasets with discrete labels, CUB200-2011 [79], Cars196 [80] and Stanford Online Products [38]. In this section, we introduce them in detail.

CUB200-2011

The dataset contains 11,788 images of 200 bird species. Each species is associated with a Wikipedia article and organized by scientific classification (order, family, genus, species). The list of species names was obtained using an online field guide[1]. Images were harvested using Flickr image search and then filtered by showing each image to multiple users of Mechanical Turk [79]. For the CUB200-2011 dataset, we use the first 100 classes with 5,864 images for training, and the remaining 100 classes with 5,924 images for testing.

Cars196

The Cars dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split. Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or

[1] http://www.birdfieldguide.com/

2012 BMW M3 coupe [80]. The first 98 model categories are used for training, with the rest for testing.

Stanford Online Products

The Stanford Online Products dataset created using the web crawling API from eBay.com[2] to download images and filtered duplicate and irrelevant images (i.e. photos of contact phone numbers, logos, etc). The preprocessed dataset has 120,053 images of 22,634 online products (classes) from eBay.com. Each product has approximately 5.3 images. For the experiments, they split 59,551 images of 11,318 classes for training and 60,502 images of 11,316 classes for testing [38].

4.1.2 Baseline Methods

Although the main focus of our experiment is to compare TopK-Pre and RS-TopK-Pre, we adopted several baseline methods. The triplet loss with semi-hard negative mining [37] is a widely used baseline representing the traditional way of metric learning. And since RS-TopK-Pre can be viewed as a listwise approach, we applied two listwise learning to rank methods, namely Lambdarank, ListNet.

Triplet Loss

Let $Z = \{z_1, ..., z_m\}$ be the training set of images. The goal of deep metric learning is to learn a nonlinear convolutional neural network (CNN) based function $f(\cdot)$ that maps an image z into a compact feature vector $f(z) \in \mathbb{R}^d$ while preserving the semantic similarities. The outputs of $f(\cdot)$ are L2-normalized.

We use the triplet loss with semi-hard negative mining [37] is a widely used baseline representing the traditional way of metric learning. The triplet loss has three inputs: anchor image z_a, positive image z_p, and negative image z_n. The method is based on learning a Euclidean embedding per image using a deep convolutional network. The network is trained such that the squared L2 distances in the embedding space directly correspond to image similarity: images of the same label have small distances and images of distinct labels have large distances [37]. The loss function is defined as:

$$\ell_{Triplet}(z_a, z_p, z_n) = \left[\|f(z_a) - f(z_p)\|_2^2 - \|f(z_a) - f(z_n)\|_2^2 + \alpha\right]_+ \tag{4.1}$$

where α is the margin.

We applied semi-hard negative mining for triplet loss. Semi-hard negative mining used for the first time in [37] aims to find negative samples within the margin. Negative samples are farther from the anchor sample when compared with false-positive samples. There is also a softer transition between positive and negative samples in this approach [2].

Lambdarank

Lambdarank is a listwise learning to rank method. It is an iterative method and uses ranking metrics to dynamically re-weight training examples after each iteration [81]. In this experiment, we use this model to learn nDCG. nDCG is a ranking metric which is

[2]https://developer.ebay.com/products/ebay-apis

commonly used in IR problems [82]. The nDCG metric for a single query image over the image list ranked by decreasing scores $\mathbf{s}$ is defined as:

$$\text{nDCG} = \frac{1}{\text{maxDCG}} \sum_{i=1}^{m} \frac{2^{y_i} - 1}{\log_2(1+i)} = \sum_{i=1}^{m} \frac{G_i}{D_i} \tag{4.2}$$

where

$$G_i = \frac{2^{y_i} - 1}{\text{maxDCG}}, D_i = \log_2(1+i)$$

are gain and discount functions respectively and maxDCG is a normalization factor per query and computed as the DCG for the list ranked by decreasing relevance labels y of the query [81]. Lambdarank multiply pairwise loss function (RankNet) by the size of change in nDCG ($\Delta\text{nDCG}(z_i, z_j)$) given by swapping the rank positions of z_i and z_j. RankNet is traditional pairwise learning to rank model. The cost function is defined as:

$$C = \frac{1}{2}\left(1 - S_{ij}\right)\sigma\left(s_i - s_j\right)) + \log\left(1 + e^{-\sigma(s_i - s_j)}\right) \tag{4.3}$$

For a given query image, let $S_{ij} \in \{0, \pm 1\}$ be defined to be 1 if image z_i has been labeled to be more relevant than image z_j, -1 if image z_i has been labeled to be less relevant than image z_j, and 0 if they have the same label [83]. In the deep metric learning settings, the distance metric is used as the scoring function. We compute the scores $s_i = \|f\left(z_a\right) - f\left(z_i\right)\|_2$ and $s_j = \|f\left(z_a\right) - f\left(z_j\right)\|_2$. Where $f(\cdot)$ denotes the CNN based mapping function. Since if the outputs of $f(\cdot)$ are L2-normalized, the performance of Lambdarank was quite limited, the outputs are without L2-normalization. In this experiment, Lambdarank is the only loss function that the outputs are not L2-normalized. $\Delta\text{nDCG}(z_i, z_j)$ is defined as the absolute difference between the nDCG values when two documents i and j are swapped [81]

$$\Delta\text{nDCG}(z_i, z_j) = |G_i - G_j| \left| \frac{1}{D_i} - \frac{1}{D_j} \right| \tag{4.4}$$

Lambdarank uses RankNet loss in Eq-4.3 and adapts it by re-withting each image pair by $\Delta\text{nDCG}(z_i, z_j)$ in each iteration

$$\ell_{Lambdarank} = \sum \Delta\text{nDCG}(z_i, z_j) \left(\frac{1}{2}\left(1 - S_{ij}\right)\sigma\left(s_i - s_j\right) + \log\left(1 + e^{-\sigma(s_i - s_j)}\right) \right) \tag{4.5}$$

ListNet

ListNet is also a listwise learning to rank method. ListNet optimizes the listwise loss function based on top one probability [69]. Given the query image (or anchor image) z_a and the set of candidate images $C = \{z_1, ..., z_m\}$ in a mini-batch, the embedded vectors are obtained as $f(z_a)$, $f(z_1)$, ..., $f(z_m)$, respectively. Each of the vectors are L2-normalized. We denote the similarity scores between z_a and the candidate images as $\mathbf{s} \in \mathbb{R}^n$, where s_i represents the similarity score between z_a and the i-th candidate z_i. We compute the score $s_i = \|f\left(z_a\right) - f\left(z_i\right)\|_2$. Then the top one probability of the score of image z_i is calculated as

$$P_{\mathbf{s}}\left(z_i\right) = \frac{\exp\left(s_i\right)}{\sum_{k=1}^{m} \exp\left(s_k\right)} \tag{4.6}$$

Let $\mathbf{y}(z_a) \in \{0,1\}^m$ be the ground-truth label vector that $y_i = 1$ iff z_a and z_i have the same class label, the top one probability of the label of image z_i is calculated as:

$$P_{\mathbf{y}}(z_i) = \frac{\exp(y_i)}{\sum_{k=1}^{m} \exp(y_k)} \tag{4.7}$$

With Cross Entropy as metric, the loss for query image (or anchor image) z_a becomes

$$L(\mathbf{s}, \mathbf{y}) = -\sum_{j=1}^{n} P_{\mathbf{y}}(z_j) \log(P_{\mathbf{s}}(z_j)) \tag{4.8}$$

4.1.3 Metrics

First we introduce the Precision, Recall and F1. Precision, recall, and the F measure are set-based measures. They are computed using unordered sets of documents [84]. Later we will see how to extend these notions to ranked retrieval situations. Precision (P) is the fraction of retrieved documents that are relevant [84]

$$\text{Precision} = \frac{\#(\text{ relevant items retrieved })}{\#(\text{ retrieved items })} = P(\text{ relevant } \mid \text{ retrieved }) \tag{4.9}$$

Recall (R) is the fraction of relevant documents that are retrieved [84]

$$\text{Recall} = \frac{\#(\text{ relevant items retrieved })}{\#(\text{ relevant items })} = P(\text{ retrieved } \mid \text{ relevant }) \tag{4.10}$$

A single measure that trades off precision versus recall is the F measure, which is the weighted harmonic mean of precision and recall [84]

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \quad \text{where} \quad \beta^2 = \frac{1-\alpha}{\alpha} \tag{4.11}$$

In our experience The balanced F measure equally weights precision and recall, which means making $\alpha = 1/2$ or $\beta = 1$. It is written as F1.

$$F1 = \frac{2PR}{P+R} \tag{4.12}$$

For many prominent applications, particularly web search, this may not be germane to users [84]. Since very few users bother to scroll down to lower-ranked items. This leads to measuring precision, recall and F1 at fixed low levels of retrieved results. These are referred to as Recall@k, Precision@k and F1@k. We use these metrics.

Specifically, for each test image, we sort all the remaining test images in a decreasing order based on the similarity score. Then we compute the average metric values with different cutoff values, which are scaled by multiplying 100.

The metrics for measuring clustering performance, such as NMI [84], are not used because they do not take into account the rank information.

4.1.4 Model Configuration

In our experiments, we adopted four baseline methods. TopK-Pre represents the state-of-the-art method for optimizing top-k precision, two listwise learning to rank methods, Lambdarank and ListNet, and the triplet loss with semi-hard negative mining [37] is a

widely used baseline representing the traditional way of metric learning. We implemented and trained all the above models using PyTorch v1.4. Specifically, ResNet50 is used as the network backbone with the embedding dimension of 128. We use the ADAM optimizer [85]. We use the training data to learn the model, and use the testing data for evaluation. For TopK-Pre and RS-TopK-Pre, we set γ as 0.1. For the triplet loss, we set the *margin* as 0.2.

4.1.5 Results and Analysis

Table 4.1: The performance of each method on CUB200-2011, Cars196 and Stanford Online Products, respectively. The best result corresponding to each metric is indicated in bold.

CUB200-2011										
	R/P/F1@1	R@2	R@4	R@8	P@2	P@4	P@8	F1@2	F1@4	F1@8
Triplet	58.09	69.65	79.93	87.32	55.50	52.76	49.11	61.78	63.56	62.86
ListNet	55.44	67.88	77.73	84.67	53.56	51.17	47.42	59.88	61.71	60.79
Lambdarank	57.93	70.00	80.40	87.49	55.79	52.90	49.24	62.09	63.81	63.01
TopK-Pre	60.65	72.55	81.74	88.83	59.06	56.18	52.40	65.11	66.59	65.92
RS-TopK-Pre	**61.92**	**73.41**	**81.95**	**89.16**	**59.52**	**56.41**	**52.70**	**65.74**	**66.82**	**66.24**
Cars196										
	R/P/F1@1	R@2	R@4	R@8	P@2	P@4	P@8	F1@2	F1@4	F1@8
Triplet	61.55	73.21	82.14	89.05	58.14	53.56	48.42	64.81	64.84	62.73
ListNet	71.30	80.19	86.78	91.28	67.77	63.57	58.41	73.46	73.38	71.24
Lambdarank	71.54	81.16	88.13	92.88	68.29	64.53	59.52	74.17	74.51	72.55
TopK-Pre	77.05	85.52	90.89	94.40	**74.50**	**70.64**	**65.46**	**79.63**	**79.50**	**77.31**
RS-TopK-Pre	**77.31**	**85.57**	**91.15**	**94.67**	73.98	69.51	63.59	79.35	78.87	76.08
Stanford Online Products										
	R/P/F1@1	R@10	R@100	R@1000	P@10	P@100	P@1000	F1@10	F1@100	F1@1000
Triplet	76.20	88.85	95.53	98.68	**32.27**	4.57	**0.55**	47.34	8.72	**1.09**
ListNet	59.04	74.93	86.39	94.85	21.43	3.33	0.46	33.33	6.41	0.92
Lambdarank	73.50	87.78	95.21	98.73	30.53	4.45	**0.55**	45.30	8.50	**1.09**
TopK-Pre	**76.44**	**89.24**	**95.86**	**98.80**	32.23	4.57	**0.55**	**47.36**	8.72	**1.09**
RS-TopK-Pre	75.54	88.91	95.68	98.79	32.15	**4.58**	**0.55**	47.22	**8.74**	**1.09**

Table 4.2: The performance of RS-TopK-Pre and TopK-Pre in terms of R/P/F1@1 on the adopted datasets. The superscripts + and $*$ indicate the scores of RS-TopK-Pre and TopK-Pre, respectively. The best result per dataset is indicated in bold.

(a) CUB200-2011

b \ k	5	10
30	59.06^{+} 59.40^{*}	59.49^{+} 58.52^{*}
50	59.32^{+} 60.50^{*}	$\mathbf{61.92}^{+}$ 59.66^{*}
100	59.12^{+} 60.65^{*}	61.24^{+} 59.96^{*}

(b) Cars196

b \ k	5	10
30	71.85^{+} 73.93^{*}	75.93^{+} 71.34^{*}
50	72.75^{+} 75.82^{*}	$\mathbf{77.31}^{+}$ 73.96^{*}
100	0.77^{+} 77.05^{*}	0.71^{+} 75.77^{*}

(c) Stanford Online Products

b \ k	5	10
30	74.67^{+} 75.49^{*}	74.68^{+} 75.01^{*}
50	75.06^{+} 75.83^{*}	74.93^{+} 74.99^{*}
100	75.54^{+} $\mathbf{76.44^{*}}$	75.27^{+} 75.42^{*}

In order to conduct a comprehensive comparison, we examined different batch sizes, such as 30, 50 and 100, for all the five methods. For TopK-Pre and RS-TopK-Pre, we also examined different cutoff values (i.e., k), such as 5 and 10, since they aim to optimize top-k precision. Due to space limitation, we show the best performance of each method after 100 epochs in Table 4.1. In particular, the best batch settings for Triplet on CUB200-2011, Cars196 and Stanford Online Products are 50, 50 and 100, respectively. The best batch settings for ListNet on CUB200-2011, Cars196 and Stanford Online Products are 100, 100 and 100, respectively. The best batch settings for Lambdarank on CUB200-2011, Cars196

and Stanford Online Products are 50, 100 and 100, respectively. The best hyper-parameter settings for TopK-Pre on CUB200-2011, Cars196 and Stanford Online Products are $b = 100$, $k = 5$, $b = 100$, $k = 5$ and $b = 100$, $k = 5$, respectively. The best hyper-parameter settings for RS-TopK-Pre on CUB200-2011, Cars196 and Stanford Online Products are $b = 50$, $k = 10$, $b = 50$, $k = 10$ and $b = 100$, $k = 5$, respectively. In Fig. 4.1, we show the example of retrieval results on CUB200-2011 using RS-TopK-Pre in the best setting.

At first glance, Table 4.1 suggests that Triplet shows poor performance, especially on CUB200-2011 and Cars196. On Stanford Online Products, the performance results are not significantly different. The possible reasons are as follows: First, the dataset difference is a possible reason. For example, the average number of images per class for CUB200-2011, Cars196 and Stanford Online Products are 58.94, 82.58 and 5.30, respectively. Thus it is difficult to achieve high performance on Stanford Online Products due to the limited number of training instances per class. Second, as shown by the prior studies [16, 75], how to sample training images plays an important role in deep metric learning. The most probable reason for Triplet's lower performance is that: the training loss is inconsistent with the direction of optimizing the adopted evaluation metric.

The performances of two listwise learning to rank methods are limited. Learning to rank methods are not designed for deep metric learning. They were proposed in document retrieval task. Specifically, Lambdarank is designed to learn nDCG, and ListNet optimizes the listwise loss function based on top one probability. The most probable reason for learning to rank's lower performances are that: the training losses are inconsistent with the direction of optimizing the adopted evaluation metric.

A closer look at Table 4.1 reveals that: On CUB200-2011, RS-TopK-Pre outperforms TopK-Pre in terms of all evaluation metrics. On Cars196, when we focus on the nearest neighbour, namely measuring the performance in terms of R/P/F1@1, RS-TopK-Pre shows improved performance. With the increase of cutoff value, TopK-Pre shows better performance. On Stanford Online Products, TopK-Pre shows better performance in most cases. RS-TopK-Pre only shows improved performance in terms of F1@100. Overall, it is reasonable to say that RS-TopK-Pre can achieve better performance than TopK-Pre in terms of R/P/F1@1. Because the nearest neighbour mostly reflects the semantic similarity. In other words, optimizing the top-k precision in a rank-sensitive manner is a better choice in the context of deep metric learning.

To clearly show the effects of batch size b and cutoff value k on RS-TopK-Pre and TopK-Pre, we investigate how their performance scores vary on three datasets in Table 4.2. In terms of R/P/F1@1, Table 4.2 shows that: (1) RS-TopK-Pre achieves better performance with $b = 50$ and $k = 10$ on both CUB200-2011 and Cars196. With a smaller cutoff value, the performance of RS-TopK-Pre is significantly impacted. However, a relatively small batch size is recommended for RS-TopK-Pre. The reason is that: if the batch size is too large, namely $b \gg k$, the rank of a false negative image within $\mathcal{P}$ can become quite large. In contrast, the rank of a false positive image within $\mathcal{N}$ is always smaller or equal to k. As a result, the weights for differentiating similarity scores within the loss become highly skewed, leading to unstable training results. This is also why we observed that RS-TopK-Pre failed to converge with $b = 100$ on Cars196. (2) TopK-Pre achieves better performance on three datasets with $b = 100$ and $k = 5$. With a smaller batch size, the performance of TopK-Pre would be impacted, especially on Cars196. To summarize, the factors, such as batch

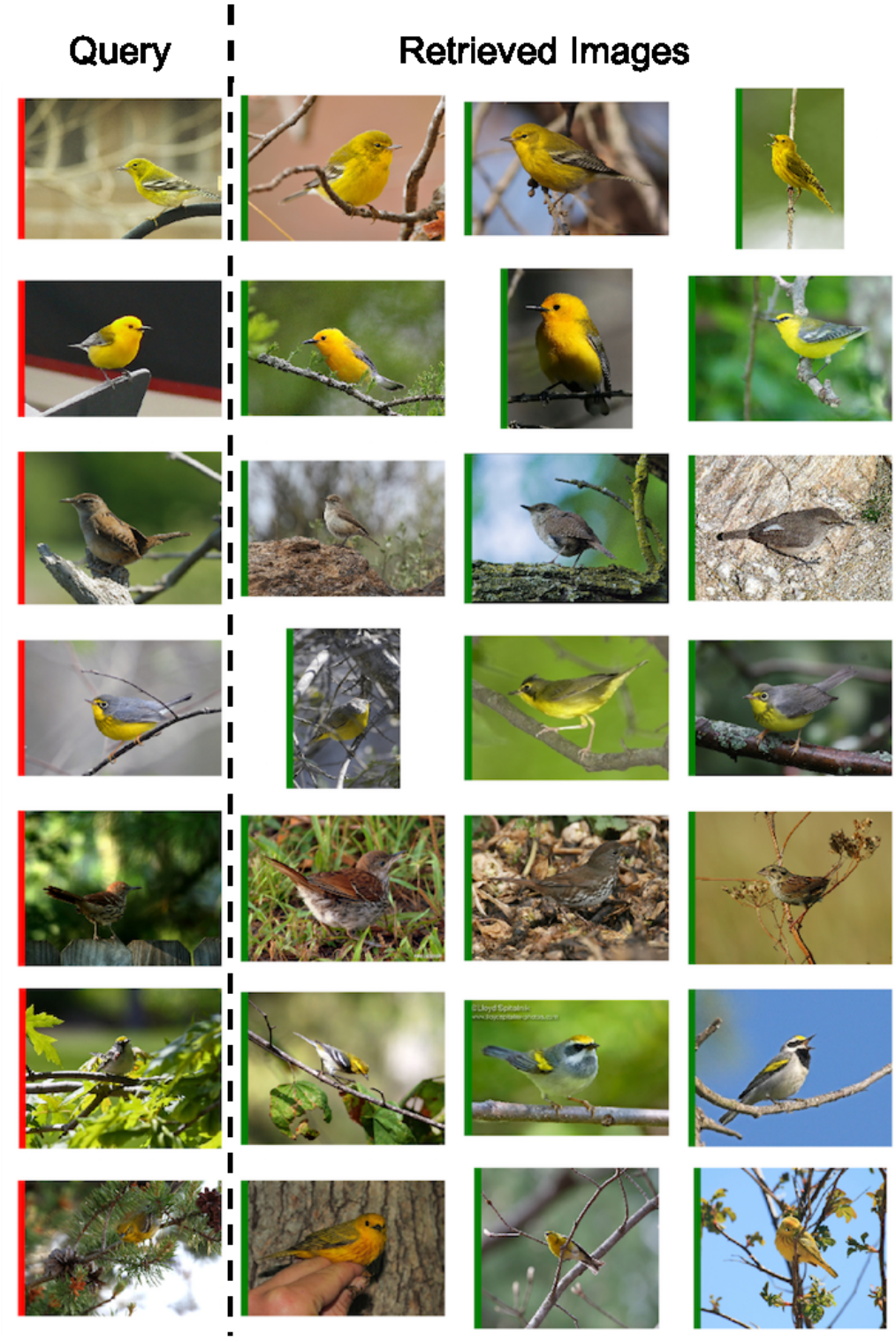


Figure 4.1: Example retrieval results on CUB200-2011 using RS-TopK-Pre. The images in the first column are the query images, images in the other column are the top-3 retrieved images.

size and the cutoff value, significantly affect the performance of approaches that rely on optimizing top-k precision for deep metric learning. Careful examinations of these factors are highly recommended.

In a nutshell, we can conclude that though the idea is intuitive, the experimental results show that the proposed method can achieve better results on two datasets, namely CUB200-2011, Cars196. Optimizing top-k precision in a rank-sensitive manner has the potential value for deep metric learning.

4.2 Evaluation Using Continuous Labels

In Chapter 3, we proposed to perform deep metric learning by optimizing the top-k precision in a rank-sensitive manner, and the experimental results show that the proposed method can achieve better results on two datasets, namely CUB200-2011 and Cars196. Although the datasets are widely used ones for deep metric learning, they merely provide discrete labels over image pairs indicating whether the image pairs are similar or not. In discrete settings, the images cannot be ordered within positive (negative) images. On the contrary the images can be ordered within positive (negative) images in continuous settings. According to that, the rank information is more important in continuous settings. Since our proposed method optimizes in a rank sensitive manner, it is interesting to investigate whether our proposed methods are effective on the dataset with continuous labels. Our main focuses are to compare TopK-Pre, RS-TopK-Pre for continuous labels (Section 3.2.3) and whether CRS-TopK-Pre (Section 3.2.4) is effective.

In this section, we first describe the test collections and evaluation metrics with continuous labels. We then describe the configuration of each method, including the parameter and the way of training.

4.2.1 Dataset

In this experiment, the effectiveness of the proposed framework is validated on an image retrieval task based on continuous similarities: human pose retrieval on the MPII human pose dataset [86].

MPII human pose

For the human pose retrieval, we directly adopt the dataset and settings of [48, 78]. Among in total 22,285 fullbody pose images, 12,366 images are used for training and 9,919 for testing, while 1,919 images among the test set are used as queries for retrieval.

These images are collected from YouTube using queries based on the activity descriptions. Using YouTube allows them to access a rich collection of videos originating from various sources, including amateur and professional recordings and capturing a variety of public events and performances. Annotated are the body joints, 3D viewpoint of the head and torso, and position of the eyes and nose. Additionally for all body joints and parts visibility is annotated [86].

The goal of human pose retrieval is to search for images similar to the query in terms of human poses they exhibit [48]. Following [48, 78], the distance between two poses is defined as the sum of Euclidean distances between body-joint locations. Unlike datasets with

discrete labels, human pose dataset don't have positive or negative labels. To use typical deep metric learning approaches top-N nearest neighbours are used as positive images others are negative.

4.2.2 Baseline Method

There are several metric learning tasks with continuous labels. In this thesis, we address the task of human pose retrieval. The task is to learn a pose-aware, compact embedding that projects images with similar human poses to be placed close-by in the embedding space [78].

Recently several deep metric learning approaches [48, 78, 87, 88] using continuous labels were proposed. They almost use conventional metric learning based on binary relations. Log-ratio loss [48] is the first attempt to directly use continuous labels for metric learning. Log-ratio loss aims to approximate the ratio of label distances by the ratio of distances in the learned embedding space. Specifically, they defined the loss function as:

$$\ell_{\mathrm{lr}}(z_a, z_i, z_j) = \left\{ \log \frac{\|f(z_a) - f(z_i)\|_2^2}{\|f(z_a) - f(z_j)\|_2^2} - \log \frac{D(y_a, y_i)}{D(y_a, y_j)} \right\}^2 \tag{4.13}$$

where $f(\cdot)$ indicates an embedding vector, y is a continuous label, and $D(\cdot)$ denotes the distance between their corresponding pose annotations. (z_a, z_i, z_j) is a triplet of an anchor z_a and its two neighbours z_i and z_j without positive-negative separation. By approximating ratios between label distances instead of the distances themselves, the proposed loss enables to learn a metric space more flexibly regardless of the scale of the labels [48]. We use this loss as a baseline.

4.2.3 Metrics

Since the labels are continuous, standard metrics like precision@k and recall@k are not suitable. Instead, following the protocol in [48, 78], we adopt two evaluation metrics, mean label distance and a modified version of nDCG [48, 78].

The mean label distance is the average of the sum of Euclidean distances between body-joint locations of top-K retrievals between queries and retrieved images, and a smaller means a better retrieval quality.

The modified version of normalized discounted cumulative gain (nDCG) for a single query image is defined as [78]:

$$\mathrm{nDCG}_K(z_a) = \frac{1}{Z_K} \sum_{i=1}^{K} \frac{2^{r_i}}{\log_2(i+1)} \tag{4.14}$$

where $r_i = -\log_2(\|y_a - y_i\|_2 + 1)$ is the relevance between the query z_a and i-th retrieval in terms of their true poses y_a and y_i. The relevance is reduced by the discounting factor $1/log2(i+1)$ to place a greater emphasis on one returned at a higher rank. nDCG_K considers only top-K retrievals, and Z_K is for normalization so that the maximum nDCG_K becomes 1. A higher nDCG means better retrieval effectiveness.

4.2.4 Model Configuration

In this experiment, we adopted two baseline methods. TopK-Pre [16] represents the state-of-the-art method for optimizing topk-precision in the settings of binary labels, and the

Log-ratio loss [48] is the state-of-the-art method in the settings of continuous labels. We implemented and trained all the above models using PyTorch v1.4. Specifically, ResNet34 [44] is used as the network backbone with the embedding dimension of 128. We use the SGD optimizer [89]. We use the training data to learn the model, and use the testing data for evaluation. For TopK-Pre, RS-TopK-Pre and CRS-TopK-Pre, we set γ as 3.0.

4.2.5 Results and Analysis

Table 4.3: The performance of each method on MPII human pose dataset. The best result corresponding to each metric is indicated in bold.

(a) The performance of mean distance.

mean distance	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10
LogRatio	**28.48**	**28.91**	**28.58**	**28.68**	**28.65**	**28.71**	**28.76**	**28.72**	**28.73**	**28.74**
TopK-Pre	29.49	29.84	30.15	30.22	30.33	30.38	30.45	30.48	30.55	30.57
RS-TopK-Pre	29.34	29.56	29.39	29.51	29.54	29.59	29.63	29.66	29.71	29.72
CRS-TopK-Pre	30.09	30.57	30.41	30.43	30.41	30.33	30.43	30.47	30.49	30.46

(b) The performance of nDCG

nDCG	@10	@20	@30	@40	@50	@60	@70	@80	@90	@100
LogRatio	**0.4511**	**0.4723**	**0.4869**	**0.4986**	**0.5077**	**0.5157**	**0.5229**	**0.5294**	**0.5352**	**0.5405**
TopK-Pre	0.4233	0.4422	0.4554	0.4655	0.4739	0.4809	0.4874	0.4930	0.4982	0.5030
RS-TopK-Pre	0.4266	0.4454	0.4585	0.4690	0.4773	0.4847	0.4911	0.4968	0.5019	0.5067
CRS-TopK-Pre	0.4224	0.4419	0.4553	0.4654	0.4739	0.4811	0.4875	0.4932	0.4982	0.5028

In order to conduct a comprehensive comparison, we examined different batch sizes, such as 50, 100 and 150, for all the four methods. For TopK-Pre, RS-TopK-Pre and CRS-TopK-Pre, we also examined different cutoff values (i.e., k), such as 5 and 10, since they aim to optimize top-k precision. We show the best performance of each method after 30 epochs in Table 4.3. In particular, the best settings for Log-ratio is 150. The best hyper-parameter settings for TopK-Pre are $b = 150$, $k = 10$. The best hyper-parameter settings for RS-TopK-Pre are $b = 50$, $k = 10$. The best hyper-parameter settings for CRS-TopK-Pre are $b = 150$, $k = 10$.

A closer look at Table 4.3 reveals that: RS-TopK-Pre outperforms TopK-Pre in terms of all evaluation metrics. It is reasonable to say that RS-TopK-Pre can achieve better performance than TopK-Pre. In other words, optimizing the top-k precision in a rank-sensitive manner is a better choice in the context of deep metric learning using continuous labels.

CRS-TopK-Pre does not outperform RS-TopK-Pre in terms of all evaluation metrics. To impose different weights in the similarity values based on the continuous rank information is worse than based on discrete rank information. CRS-TopK-Pre outperforms TopK-Pre in terms of mean distance@6, 7, 8, 9, 10, and nDCG@60, 70, 80. Because the nearest neighbour mostly reflects the semantic similarity, CRS-TopK-Pre underperforms TopK-Pre.

Overall, CRS-TopK-Pre failed to outperform both TopK-Pre and RS-TopK-Pre. The possible reason is as follows: The sampling method of CRS-TopK-Pre is the same as TopK-Pre, it is based on alternative binary judgement. The weighting function is based on continuous labels. The difference between sampling function and weighting function have a negative impact on performance. Optimizing the top-k precision in a continuous rank-sensitive manner

is not a good choice in the context of deep metric learning using a dataset with continuous labels.

Table 4.3 reveals that Log-ratio shows the best performance in terms of all evaluation metrics. The possible reason is as follows: Despite not being able to directly calculate top-k precision since the labels are continuous, TopK-Pre, RS-TopK-Pre and CRS-TopK-Pre try to optimize the top-k precision.

4.2.6 Summary

In this section, we investigate whether our proposed method (RS-TopK-Pre) is effective on the dataset with continuous labels, and propose to perform deep metric learning by optimizing top-k precision in a continuous rank-sensitive manner (CRS-TopK-Pre). The experimental results show that RS-TopK-Pre can achieve better results than TopK-Pre. It shows the potential value of rank-sensitive optimization of top-k precision for deep metric learning with continuous labels.

CRS-TopK-Pre failed to outperform both TopK-Pre and RS-TopK-Pre. To optimize the top-k precision in a continuous rank-sensitive manner is not a good choice in the context of deep metric learning using a dataset with continuous labels.

Since it is necessary to use alternative binary labels to calculate top-k precision in continuous labels, to optimize top-k precision is a worse choice than to optimize Log-ratio loss.

Chapter 5

Conclusions and Future Work

In this chapter, we summarize the contributions achieved in this work, highlight their strengths and limitations, and discuss interesting directions for future research.

5.1 Conclusions

The central contribution of this thesis is to propose to optimize top-k precision in a rank-sensitive manner. The key idea is to differentiate the misplaced images sampled according to top-k precision by incorporating their rank information. Though the idea is intuitive, the proposed method (RS-TopK-Pre) can achieve better results than TopK-Pre on two datasets, namely CUB200-2011, Cars196. And, in deep metric learning approaches that rely on optimizing top-k precision, the factors, such as batch size and cutoff value k, significantly affect the performance.

Next, we adjust RS-TopK-Pre to cope with continuous labels. Although the datasets we used in the above are widely used ones for deep metric learning, they merely provide discrete labels over image pairs indicating whether the image pairs are similar or not. We investigate whether our proposed methods are effective on a dataset with continuous labels. The experimental results show that: RS-TopK-Pre outperforms TopK-Pre. It shows the potential value of rank-sensitive optimization of top-k precision for deep metric learning with continuous labels. In view of the unique property of continuous labels, we further modified RS-TopK-Pre by imposing different weights to further differentiate misplaced images based on continuous rank information (CRS-TopK-Pre). CRS-TopK-Pre failed to achieve better performance than TopK-Pre.

5.2 Limitations and Future Work

We investigated the effect cut off value and batch size on the performance of proposed models and baseline methods. According to experimental results, we observe that the factors, such as batch size and cutoff value k, significantly affect the performance. Specifically, our proposed loss is rather ad-hoc and seems very sensitive to the hyper parameters. As a future work, further examination of these factors are thus recommended.

In the experiments with discrete labels, our proposed method failed to outperform TopK-Pre on Stanford Online Products. The possible reason is that: the average number of images per class for Stanford Online Products are quite limited than others. As a result, optimizing

top-k precision in a rank-sensitive manner is difficult to achieve high performance where the number of training instances per class are limited.

In the experiments with continuous labels, although the experimental results show that optimizing top-k precision in a rank-sensitive manner is effective, the proposed methods underperformed Log-ratio loss. The possible reason is that: it is not possible to directly calculate top-k precision. It is necessary to use alternative binary labels in order to calculate top-k precision .

For future work, the following practical issues are worthy to be investigated. Besides top-k precision, average precision (AP) and nDCG are also popular metrics for measuring semantic similarity. We would like to explore the potential value of optimizing AP and nDCG for deep metric learning.

Acknowledgement

First, I would like to express my deepest gratitude to my advisor, Dr. Yu Hai-Tao, for his kind mentoring over the years. As his student, I was fortunate to have a great deal of academic freedom and rigorous training. His never-waning enthusiasm and high standards permeated our everyday interactions, and I learned a great deal about academic research, writing, and critical thinking in general from him. I am certain that these will continue to reward me for many years to come. I am proud to be his student.

I am also very grateful to my adviser Dr. Joho Hideo. Thoughout my graduate study, he has provided me a lot of helps and inspiring comments. And I would also like to thank my colleagues in the lab for their helps and advices.

References

[1] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, Vol. 44, No. 1.2, pp. 206–226, 2000.

[2] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, Vol. 11, No. 9, p. 1066, 2019.

[3] R. Short and K. Fukunaga. The optimal distance measure for nearest neighbor classification. *IEEE Transactions on Information Theory*, Vol. 27, No. 5, pp. 622–627, 1981.

[4] Keinosuke Fukunaga. *Instruction to Statistical Pattern Recognition*. Elsevier, 1972.

[5] Jerome H Friedman. Flexible metric nearest neighbor classification. Technical report, Citeseer, 1994.

[6] Trevor Hastie and Robert Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 18, No. 6, pp. 607–616, 1996.

[7] Jonathan Baxter and Peter Bartlett. The canonical distortion measure in feature space and 1-nn classification. *Proceedings of NeurIPS*, 1997.

[8] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. *Proceedings of NeurIPS*, Vol. 15, pp. 521–528, 2002.

[9] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *CoRR*, 2013.

[10] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, Vol. 10, No. 2, pp. 1–37, 2019.

[11] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, Vol. 10, pp. 207–244, 2009.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, Vol. 521, No. 7553, pp. 436–444, 2015.

[13] Khai Tran and Thi Phan. Deep learning application to ensemble learning—the simple, but effective, approach to sentiment classifying. *Applied Sciences*, Vol. 9, No. 13, 2019.

[14] J. Lu, J. Hu, and J. Zhou. Deep metric learning for visual understanding: An overview of recent advances. *IEEE Signal Processing Magazine*, Vol. 34, No. 6, pp. 76–84, 2017.

[15] Kun He. *Learning deep embeddings by learning to rank*. PhD thesis, Boston University, 2018.

[16] Jing Lu, Chaofan Xu, Wei Zhang, Lingyu Duan, and Tao Mei. Sampling wisely: Deep image by top-k precision optimization. In *Proceedings of ICCV*, pp. 7961–7970, 2019.

[17] Naoki Muramoto and Hai-Tao Yu. Deep metric learning based on rank-sensitive optimization of top-k precision. In *Proceedings of CIKM*, pp. 2161–2164, 2020.

[18] Eui-Hong Sam Han, George Karypis, and Vipin Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *Pacific-asia conference on knowledge discovery and data mining*, pp. 53–65, 2001.

[19] Ali Mustafa Qamar and Eric Gaussier. Online and batch learning of generalized cosine similarities. In *2009 Ninth IEEE International Conference on Data Mining*, pp. 926–931, 2009.

[20] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*. ACM press New York, 1999.

[21] Josef Sivic and Andrew Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 31, No. 4, pp. 591–606, 2008.

[22] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. An online algorithm for large scale image similarity learning. In *Proceedings of NeurIPS*, pp. 306–314, 2009.

[23] Li Cheng. Riemannian similarity learning. In *Proceedings of ICML*, pp. 540–548, 2013.

[24] J. Deng, A. C. Berg, and L. Fei-Fei. Hierarchical semantic indexing for large scale image retrieval. In *Proceedings of CVPR*, pp. 785–792, 2011.

[25] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, Vol. 10, No. 5, pp. 1299–1319, 1998.

[26] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Vol. 2, No. 11, pp. 559–572, 1901.

[27] Ratthachat Chatpatanasiri, Teesid Korsrilabutr, Pasakorn Tangchanachaianan, and Boonserm Kijsirikul. A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomputing*, Vol. 73, No. 10-12, pp. 1570–1579, 2010.

[28] Changshui Zhang, Feiping Nie, and Shiming Xiang. A general kernelization framework for learning algorithms based on kernel pca. *Neurocomputing*, Vol. 73, No. 4–6, pp. 959–967, 2010.

[29] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of CVPR*, pp. 539–546, 2005.

[30] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pp. 412–419, 2007.

[31] Zhixiang Xu, Kilian Q Weinberger, and Olivier Chapelle. Distance metric learning for kernel machines. *arXiv*, 2012.

[32] Dor Kedem, Stephen Tyree, Fei Sha, Gert Lanckriet, and Kilian Q Weinberger. Nonlinear metric learning. *Proceedings of NeurIPS*, Vol. 25, pp. 2573–2581, 2012.

[33] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. Hamming distance metric learning. *Proceedings of NeurIPS*, Vol. 25, pp. 1061–1069, 2012.

[34] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 7, No. 04, pp. 669–688, 1993.

[35] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Deep metric learning for person re-identification. In *Proceedings of ICPR*, pp. 34–39, 2014.

[36] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pp. 84–92, 2015.

[37] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *Proceedings of CVPR*, pp. 815–823, 2015.

[38] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of CVPR*, pp. 554–561, 2016.

[39] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of NeurIPS*, pp. 1857–1865, 2016.

[40] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *Proceedings of CVPR*, pp. 2206–2214, 2017.

[41] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proceedings of ICCV*, pp. 2593–2601, 2017.

[42] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Proceedings of NeurIPS*, pp. 4170–4178, 2016.

[43] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *Proceedings of CVPR*, pp. 1861–1870, 2019.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of CVPR*, pp. 770–778, 2016.

[45] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of CVPR*, pp. 248–255, 2009.

[46] Karsten Roth, Biagio Brattoli, and Bjorn Ommer. Mic: Mining interclass characteristics for improved metric learning. In *Proceedings of ICCV*, pp. 8000–8009, 2019.

[47] Artsiom Sanakoyeu, Vadim Tschernezki, Uta Buchler, and Bjorn Ommer. Divide and conquer the embedding space for metric learning. In *Proceedings of CVPR*, pp. 471–480, 2019.

[48] Sungyeon Kim, Minkyo Seo, Ivan Laptev, Minsu Cho, and Suha Kwak. Deep metric learning beyond binary supervision. In *Proceedings of CVPR*, pp. 2288–2297, 2019.

[49] Hai-Tao Yu, Adam Jatowt, Hideo Joho, Joemon M. Jose, Xiao Yang, and Long Chen. Wassrank: Listwise document ranking using optimal transport theory. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 24–32, 2019.

[50] Tie-Yan Liu. Learning to rank for information retrieval. *Information Retrieval*, Vol. 3, No. 3, pp. 225–331, 2009.

[51] David Cossock and Tong Zhang. Subset ranking using regression. In *Proceedings of the 19th Annual Conference on Learning Theory*, pp. 605–619, 2006.

[52] Ramesh Nallapati. Discriminative models for information retrieval. In *Proceedings of SIGIR*, pp. 64–71, 2004.

[53] Wei Chu and Zoubin Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, Vol. 6, pp. 1019–1041, 2005.

[54] Wei Chu and S. Sathiya Keerthi. New approaches to support vector ordinal regression. In *Proceedings of ICML*, pp. 145–152, 2005.

[55] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of ICML*, pp. 89–96, 2005.

[56] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, Vol. 4, pp. 933–969, 2003.

[57] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of KDD*, pp. 133–142, 2002.

[58] Libin Shen and Aravind K. Joshi. Ranking and reranking with perceptron. *Machine Learning*, Vol. 60, No. 1-3, pp. 73–96, 2005.

[59] Fajie Yuan, Guibing Guo, Joemon Jose, Long Chen, Hai-Tao Yu, and Weinan Zhang. Lambdafm: learning optimal ranking with factorization machines using lambda surrogates. In *Proceedings of CIKM*, pp. 227–236, 2016.

[60] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, Vol. 20, No. 4, pp. 422–446, 2002.

[61] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of CIKM*, pp. 621–630, 2009.

[62] Olivier Chapelle, Quoc Le, and Alex Smola. Large margin optimization of ranking measures. In *NIPS workshop on Machine Learning for Web Search*, 2007.

[63] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of SIGIR*, pp. 391–398, 2007.

[64] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of SIGIR*, pp. 271–278, 2007.

[65] John Guiver and Edward Snelson. Learning to rank with softrank and gaussian processes. In *Proceedings of SIGIR*, pp. 259–266, 2008.

[66] Tao Qin, Tie-Yan Liu, and Hang Li. A general approximation framework for direct optimization of information retrieval measures. *Journal of Information Retrieval*, Vol. 13, No. 4, pp. 375–397, 2010.

[67] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of WSDM*, pp. 77–86, 2008.

[68] Christopher J.C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *Proceedings of NeurIPS*, pp. 193–200, 2006.

[69] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of ICML*, pp. 129–136, 2007.

[70] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of ICML*, pp. 1192–1199, 2008.

[71] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, Vol. 11, No. 36, pp. 1109–1135, 2010.

[72] Daryl Lim and Gert Lanckriet. Efficient learning of mahalanobis metrics for ranking. In *Proceedings of ICML*, pp. 1980–1988, 2014.

[73] Brian McFee and Gert Lanckriet. Metric learning to rank. In *Proceedings of ICML*, pp. 775–782, 2010.

[74] B. G. Vijay Kumar, G. Carneiro, and I. Reid. Learning local image descriptors with deep siamese and triplet convolutional networks by minimizing global loss functions. In *Proceedings of CVPR*, pp. 5385–5394, 2016.

[75] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of ICCV*, pp. 2840–2848, 2017.

[76] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proceedings of ICCV*, pp. 814–823, 2017.

[77] Swami Sankaranarayanan, Azadeh Alavi, and Rama Chellappa. Triplet similarity embedding for face verification. *arXiv*, 2016.

[78] Suha Kwak, Minsu Cho, and Ivan Laptev. Thin-slicing for pose: Learning to understand pose without explicit pose estimation. In *Proceedings of CVPR*, pp. 4938–4947, 2016.

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

[80] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of ICCV*, pp. 554–561, 2013.

[81] Xuanhui Wang, Cheng Li, Nadav Golbandi, Mike Bendersky, and Marc Najork. The lambdaloss framework for ranking metric optimization. In *Proceedings of CIKM*, pp. 1313–1322, 2018.

[82] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, Vol. 20, No. 4, pp. 422–446, 2002.

[83] Chris J.C. Burges. From ranknet to lambdarank to lambdamart: An overview. *Microsoft Research Technical Report*, No. MSR-TR-2010-82, 2010.

[84] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[85] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014.

[86] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of CVPR*, pp. 3686–3693, 2014.

[87] Greg Mori, Caroline Pantofaru, Nisarg Kothari, Thomas Leung, George Toderici, Alexander Toshev, and Weilong Yang. Pose embeddings: A deep architecture for learning to match human poses. *arXiv*, 2015.

[88] Omer Sumer, Tobias Dencker, and Bjorn Ommer. Self-supervised learning of pose embeddings from spatiotemporal relations in videos. In *Proceedings of ICCV*, pp. 4298–4307, 2017.

[89] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of ICML*, pp. 1139–1147, 2013.