GradML: A Gradient-based Loss for Deep Metric Learning

Bhavya Vasudeva[1]* **Puneesh Deora**[1]* **Saumik Bhattacharya**[2], **Umapada Pal**[1], **Sukalpa Chanda**[3]
[1] ISI Kolkata [2] IIT Kharagpur [3] Østfold University College

Abstract

Deep metric learning (ML) uses a carefully designed loss function to learn distance metrics for improving the discriminatory ability for tasks like clustering and retrieval. Most loss functions are designed by considering the distance between the embeddings to induce certain properties without exploring how such losses would impact the movement of the said embeddings via their gradients during optimization. In this work, we analyze the gradients of various ML loss functions and propose a gradient-based loss for ML (GradML). Instead of directly formulating the loss, we first formulate the gradients of the loss and use them to derive the loss to be optimized. It has a simple formulation and lowers the computational cost as compared to other methods. We evaluate our approach on three datasets and find that the performance is data-dependent on properties like inter-class variance.

1 Introduction

Deep metric learning (ML) involves moving samples from similar classes close and dissimilar classes farther apart in the embedding space. This is done by training a deep network to learn the mapping from the sample space to the embedding space, using some criterion in the form of a loss function to be optimized. This has proven to be useful in a variety of applications like face recognition (1; 2), re-identification (3; 4; 5; 6), image retrieval (7; 8), etc.

Contrastive (1; 9) and triplet loss (2; 10) are two common criteria that consider a pair or a triplet of samples to learn the embeddings. Other loss functions such as lifted structure (LS) (11) and N-pair (12) serve as generalizations of triplet loss as they use higher-order relations among the data points. Angular loss (13) aims to leverage the angular constraints in the loss criterion and multi-similarity (MS) (14) loss assigns different weights to each pair indicative of their usefulness. Proxy-based loss functions (15; 16; 17; 18) rely on proxies to overcome some of the issues associated with sampling-based pair-wise losses, like the tendency to overfit, by introducing proxies.

All the aforementioned works try to improve the learned embeddings by designing loss functions that use similarity metrics to force the data points to follow a particular criterion. However, the direction in which the embeddings move during optimization is a critical aspect, determined by the gradients of the loss function. Simply forcing the negative points away and attracting the positive points close to each other (under some criterion) may not be sufficient to obtain discriminative regions. To study this, we formulate a loss function that takes into account the direction of movement of samples. We start with the expression for the gradient of the loss function with respect to each sample which is subsequently integrated to obtain the desired loss. The proposed loss function turns out as the k^{th} power of the ℓ_2-norm of the differences between the sample pairs.

The proposed approach is simple as it relies on pair-wise distances, avoids computational complexity of some of other approaches, and uses fewer samples to achieve competitive performance, when evaluated on CUB-200-2011 (19), a benchmark dataset. We analyze the gradients of other losses

*Equal contribution;

and compare them with our method to show how it benefits the learning process. Surprisingly, when evaluated on two other benchmark datasets, it does not perform so well. This might be because the inter-class variations are not uniform across the entire dataset.

1.1 Related Work

To accelerate the training process, mining-based strategies are used to search for the most informative samples. Methods like semi-hard mining (2), distance-weighted (Dis-wtd) sampling (20), smart mining (21), hard-aware cascaded embeddings (HDC) (22), structured clustering (StructClust) (23), discriminative sampling policy (DE-DSP) (24) have been proposed. Their drawback of is that they rely on a subset of samples and may induce overfitting and bias in the learned embeddings.

In addition, some methods try to generate new samples which are harder to classify. This can be done either in the sample space, as in adversarial ML (DAML) (25), hardness-aware ML (HDML) (26), which requires an additional generative network or directly in the embedding space, as in embedding expansion (EE) (27), symmetrical synthesis (SymmSyn) (28), looking for optimal hard negatives (LoOp) (29), which may not always ensure the class belongingness of the generated samples.

Recently, a direction regularizer (DR)-based method (30) has been proposed, to induce orthogonality between an anchor-positive pair and a negative sample, used in conjuction with other losses.

2 Methodology

Let $\mathbf{x_1}, \mathbf{x_2}$ and $\mathbf{y_1}, \mathbf{y_2}$ denote embedding vectors belonging to class A and class B, respectively, and L denote the loss function. $\mathbf{x_1} - \mathbf{y_1}, \mathbf{x_1} - \mathbf{y_2}, \mathbf{x_2} - \mathbf{x_1}$ denote the direction vectors that would be used in the gradient updates of $\mathbf{x_1}$. $\frac{\partial L}{\partial \mathbf{x_1}}$ can be formulated as a weighted sum of these vectors:

$$\frac{\partial L}{\partial \mathbf{x_1}} = 2(f_1(\mathbf{x_1}, \mathbf{x_2}, \mathbf{y_1}, \mathbf{y_2})(\mathbf{x_1} - \mathbf{x_2}) - f_2(\mathbf{x_1}, \mathbf{x_2}, \mathbf{y_1}, \mathbf{y_2})(\mathbf{x_1} - \mathbf{y_1}) - f_3(\mathbf{x_1}, \mathbf{x_2}, \mathbf{y_1}, \mathbf{y_2})(\mathbf{x_1} - \mathbf{y_2})),$$

where $f_i's$ are scalar functions of $\mathbf{x_{1,2}}, \mathbf{y_{1,2}}$. Writing $f_i(\mathbf{x_1}, \mathbf{x_2}, \mathbf{y_1}, \mathbf{y_2})$ as f_i for conciseness gives:

$$\frac{\partial L}{\partial \mathbf{x_1}} = 2(f_1(\mathbf{x_1} - \mathbf{x_2}) - f_2(\mathbf{x_1} - \mathbf{y_1}) - f_3(\mathbf{x_1} - \mathbf{y_2})). \tag{1}$$

Integrating (1), we get:

$$L = f_1||\mathbf{x_1} - \mathbf{x_2}||_2^2 - f_2||\mathbf{x_1} - \mathbf{y_1}||_2^2 - f_3||\mathbf{x_1} - \mathbf{y_2}||_2^2 + \phi_1(\mathbf{x_2}, \mathbf{y_1}, \mathbf{y_2}), \tag{2}$$

where $|| \cdot ||_2$ denotes the l_2-norm. Proceeding in the same manner, we write the expression for $\frac{\partial L}{\partial \mathbf{x_2}}$ and integrate it to get:

$$L = g_1||\mathbf{x_2} - \mathbf{x_1}||_2^2 - g_2||\mathbf{x_2} - \mathbf{y_1}||_2^2 - g_3||\mathbf{x_2} - \mathbf{y_2}||_2^2 + \phi_2(\mathbf{x_1}, \mathbf{y_1}, \mathbf{y_2}). \tag{3}$$

One of the ways to establish the equivalence of (2) and (3) is discussed hereon. Since ϕ_1 is independent of $\mathbf{x_1}$ and ϕ_2 is independent of $\mathbf{x_2}$, we get $f_1 = g_1$, f_2, f_3 are independent of $\mathbf{x_2}$ and g_2, g_3 are independent of $\mathbf{x_1}$. Using these, we get:

$$\begin{aligned} L = {} & f_1||\mathbf{x_1} - \mathbf{x_2}||_2^2 - f_2(\mathbf{x_1}, \mathbf{y_1}, \mathbf{y_2})||\mathbf{x_1} - \mathbf{y_1}||_2^2 - f_3(\mathbf{x_1}, \mathbf{y_1}, \mathbf{y_2})||\mathbf{x_1} - \mathbf{y_2}||_2^2 \\ & - f_4(\mathbf{x_2}, \mathbf{y_1}, \mathbf{y_2})||\mathbf{x_2} - \mathbf{y_1}||_2^2 - f_5(\mathbf{x_2}, \mathbf{y_1}, \mathbf{y_2})||\mathbf{x_2} - \mathbf{y_2}||_2^2 + \phi_y(\mathbf{y_1}, \mathbf{y_2}). \end{aligned} \tag{4}$$

Proceeding in the same manner for $\mathbf{y_1}, \mathbf{y_2}$, we get:

$$\begin{aligned} L = {} & h_1||\mathbf{y_1} - \mathbf{y_2}||_2^2 - h_2(\mathbf{y_1}, \mathbf{x_1}, \mathbf{x_2})||\mathbf{y_1} - \mathbf{x_1}||_2^2 - h_3(\mathbf{y_1}, \mathbf{x_1}, \mathbf{x_2})||\mathbf{y_1} - \mathbf{x_2}||_2^2 \\ & - h_4(\mathbf{y_2}, \mathbf{x_1}, \mathbf{x_2})||\mathbf{y_2} - \mathbf{x_1}||_2^2 - h_5(\mathbf{y_2}, \mathbf{x_1}, \mathbf{x_2})||\mathbf{y_2} - \mathbf{x_2}||_2^2 + \phi_x(\mathbf{x_1}, \mathbf{x_2}). \end{aligned} \tag{5}$$

Comparing (4) and (5), we get $h_2 = f_2 = \gamma_3(\mathbf{x_1}, \mathbf{y_1})$, $h_3 = f_4 = \gamma_5(\mathbf{x_2}, \mathbf{y_1})$, $h_4 = f_3 = \gamma_4(\mathbf{x_1}, \mathbf{y_2})$, $h_5 = f_5 = \gamma_6(\mathbf{x_2}, \mathbf{y_2})$. Also, $f_1 = \gamma_1(\mathbf{x_1}, \mathbf{x_2})$, $h_1 = \gamma_2(\mathbf{y_1}, \mathbf{y_2})$. Using these, we get:

$$\begin{aligned} L = {} & \gamma_1(\mathbf{x_1}, \mathbf{x_2})||\mathbf{x_1} - \mathbf{x_2}||_2^2 + \gamma_2(\mathbf{y_1}, \mathbf{y_2})||\mathbf{y_1} - \mathbf{y_2}||_2^2 - \gamma_3(\mathbf{x_1}, \mathbf{y_1})||\mathbf{x_1} - \mathbf{y_1}||_2^2 \\ & - \gamma_4(\mathbf{x_1}, \mathbf{y_2})||\mathbf{x_1} - \mathbf{y_2}||_2^2 - \gamma_5(\mathbf{x_2}, \mathbf{y_1})||\mathbf{x_2} - \mathbf{y_1}||_2^2 - \gamma_6(\mathbf{x_2}, \mathbf{y_2})||\mathbf{x_2} - \mathbf{y_2}||_2^2. \end{aligned} \tag{6}$$

Differentiating (6) w.r.t. $\mathbf{x_1}$:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{x_1}} = {} & 2(\gamma_1(\mathbf{x_1}, \mathbf{x_2})(\mathbf{x_1} - \mathbf{x_2}) - \gamma_3(\mathbf{x_1}, \mathbf{y_1})(\mathbf{x_1} - \mathbf{y_1}) - \gamma_4(\mathbf{x_1}, \mathbf{y_2})(\mathbf{x_1} - \mathbf{y_2})) \\ & + \frac{\partial \gamma_1(\mathbf{x_1}, \mathbf{x_2})}{\partial \mathbf{x_1}}||\mathbf{x_1} - \mathbf{x_2}||_2^2 - \frac{\partial \gamma_3(\mathbf{x_1}, \mathbf{y_1})}{\partial \mathbf{x_1}}||\mathbf{x_1} - \mathbf{y_1}||_2^2 - \frac{\partial \gamma_4(\mathbf{x_1}, \mathbf{y_2})}{\partial \mathbf{x_1}}||\mathbf{x_1} - \mathbf{y_2}||_2^2. \end{aligned}$$

The expressions for γ's can be obtained by solving:

$$2\gamma_1(\mathbf{x_1},\mathbf{x_2})(\mathbf{x_1}-\mathbf{x_2}) + \frac{\partial \gamma_1(\mathbf{x_1},\mathbf{x_2})}{\partial \mathbf{x_1}}||\mathbf{x_1}-\mathbf{x_2}||_2^2 = k\gamma_1(\mathbf{x_1},\mathbf{x_2})(\mathbf{x_1}-\mathbf{x_2}), \quad (7)$$

where $k > 0$ is a hyperparameter. Simplifying (7), we get:

$$\frac{\partial \gamma_1(\mathbf{x_1},\mathbf{x_2})}{\partial \mathbf{x_1}} = (k-2)\gamma_1(\mathbf{x_1},\mathbf{x_2})||\mathbf{x_1}-\mathbf{x_2}||_2^{-2}(\mathbf{x_1}-\mathbf{x_2}). \quad (8)$$

Let $\gamma_1(\mathbf{x_1},\mathbf{x_2}) = a_1 e^{f(\mathbf{x_1},\mathbf{x_2})}$. Substituting in (8), we get:

$$\frac{\partial f(\mathbf{x_1},\mathbf{x_2})}{\partial \mathbf{x_1}} = (k-2)||\mathbf{x_1}-\mathbf{x_2}||_2^{-2}(\mathbf{x_1}-\mathbf{x_2}). \quad (9)$$

Integrating (9), we get $f(\mathbf{x_1},\mathbf{x_2}) = (k-2)\log(||\mathbf{x_1}-\mathbf{x_2}||_2)$. Using this, we get:

$$\gamma_1(\mathbf{x_1},\mathbf{x_2}) = a_1 e^{\log(||\mathbf{x_1}-\mathbf{x_2}||_2^{k-2})} = a_1||\mathbf{x_1}-\mathbf{x_2}||_2^{k-2}.$$

Similarly, we can obtain the expressions for the other γ_i's. Substituting in (6), we get:

$$\begin{aligned} L = a_1||\mathbf{x_1}-\mathbf{x_2}||_2^k + a_2||\mathbf{y_1}-\mathbf{y_2}||_2^k - a_3||\mathbf{x_1}-\mathbf{y_1}||_2^k - a_4||\mathbf{x_1}-\mathbf{y_2}||_2^k \\ - a_5||\mathbf{x_2}-\mathbf{y_1}||_2^k - a_6||\mathbf{x_2}-\mathbf{y_2}||_2^k, \quad (10)\end{aligned}$$

which is the proposed gradient-based loss. The 'weights' (a_i's) are hyperparameters. In this work, we set $a_1 = a_2 = 1$ and $a_3 = a_4 = a_5 = a_6 = w$, to get:

$$L = ||\mathbf{x_1}-\mathbf{x_2}||_2^k + ||\mathbf{y_1}-\mathbf{y_2}||_2^k - w(||\mathbf{x_1}-\mathbf{y_1}||_2^k + ||\mathbf{x_1}-\mathbf{y_2}||_2^k + ||\mathbf{x_2}-\mathbf{y_1}||_2^k + ||\mathbf{x_2}-\mathbf{y_2}||_2^k). \quad (11)$$

3 Experiments and Results

We perform various experiments to evaluate GradML. We report the F_1 score, normalized mutual information (NMI), and Recall@K values for K = 1, 2, 4 for the standardized tasks of image clustering (31) and retrieval, respectively. We use the Caltech-UCSD Birds-200-2011 (CUB-200-2011) (19), Cars196 (32), Stanford Online Products (SOP) (11) datasets to evaluate GradML. More details about the datasets, implementation settings and results for different values of hyperparameters w and k are in the appendix.

3.1 Analysis of Gradients of Other Losses

Here, we analyze the gradients of existing ML-based losses, namely triplet loss, DR-triplet loss, LS loss. Considering only 2 samples from 2 classes, the expressions for each loss are given as follows:
Triplet: $[||\mathbf{x_1}-\mathbf{x_2}||_2^2 - ||\mathbf{x_1}-\mathbf{y_1}||_2^2 + m]_+$, where $[\cdot]_+$ denotes hinge loss, and m is the margin.
DR-Triplet: $||\mathbf{x_1}-\mathbf{x_2}||_2^2 - ||\mathbf{x_1}-\mathbf{y_1}||_2^2 + m - g\frac{1-\mathbf{x_1}\cdot\mathbf{x_2}}{||\mathbf{x_1}-\mathbf{y_1}||_2||\mathbf{x_1}-\mathbf{x_2}||_2}$, where g is the DR parameter.
LS: $[||\mathbf{x_1}-\mathbf{x_2}||_2^2 - D + m]_+ + [||\mathbf{y_1}-\mathbf{y_2}||_2^2 - D + m]_+$, where $D = \min(D_{11}, D_{12}, D_{21}, D_{22})$, $D_{ij} = ||\mathbf{x_i}-\mathbf{y_j}||_2^2$.
GradML: (11), with $k = 2$.

Table 1: Expressions for gradients of the loss functions w.r.t each sample.

Loss	$\frac{\partial L}{\partial \mathbf{x_1}}$	$\frac{\partial L}{\partial \mathbf{x_2}}$	$\frac{\partial L}{\partial \mathbf{y_1}}$	$\frac{\partial L}{\partial \mathbf{y_2}}$
Triplet	$2(\mathbf{y_1}-\mathbf{x_2})$	$2(\mathbf{x_2}-\mathbf{x_1})$	$2(\mathbf{y_1}-\mathbf{x_2})$	$\mathbf{0}$
DR-Triplet	$2(\mathbf{y_1}-\mathbf{x_2}) - g(c(\mathbf{x_1}-\mathbf{x_2})$ $+dk(\mathbf{y_1}-\mathbf{x_1}))$	$2(\mathbf{x_2}-\mathbf{x_1})$ $-gc(\mathbf{x_2}-\mathbf{x_1})$	$2(\mathbf{x_1}-\mathbf{y_1})$ $-gc(\mathbf{x_1}-\mathbf{y_1})$	$\mathbf{0}$
LS	$2(\mathbf{x_1}-\mathbf{x_2}) - \mathbb{1}_{D=D_{11}}(\mathbf{x_1}-\mathbf{y_1})$ $-\mathbb{1}_{D=D_{12}}(\mathbf{x_1}-\mathbf{y_2})$	$2(\mathbf{x_2}-\mathbf{x_1}) - \mathbb{1}_{D=D_{21}}(\mathbf{x_2}-\mathbf{y_1})$ $-\mathbb{1}_{D=D_{22}}(\mathbf{x_2}-\mathbf{y_2})$	$2(\mathbf{y_1}-\mathbf{y_2}) - \mathbb{1}_{D=D_{11}}(\mathbf{y_1}-\mathbf{x_1})$ $-\mathbb{1}_{D=D_{21}}(\mathbf{y_1}-\mathbf{x_2})$	$2(\mathbf{y_2}-\mathbf{y_1}) - \mathbb{1}_{D=D_{12}}(\mathbf{y_2}-\mathbf{x_1})$ $-\mathbb{1}_{D=D_{22}}(\mathbf{y_2}-\mathbf{x_2})$
GradML	$2w(\mathbf{y_1}+\mathbf{y_2}-\mathbf{x_2})$	$2w(\mathbf{y_1}+\mathbf{y_2}-\mathbf{x_1})$	$2w(\mathbf{x_1}+\mathbf{x_2}-\mathbf{y_2})$	$2w(\mathbf{x_1}+\mathbf{x_2}-\mathbf{y_1})$

The derivatives of these expressions w.r.t each sample, when the output of the hinge function is positive, are listed in Table 1. $\mathbb{1}$ denotes the indicator function, $c = (||\mathbf{y_1}-\mathbf{x_1}||_2||\mathbf{x_1}-\mathbf{x_2}||_2)^{-1}$, $d = ||\mathbf{y_1}-\mathbf{x_1}||_2^{-2}$, and $k = ||\mathbf{x_1}-\mathbf{x_2}||_2||\mathbf{y_1}-\mathbf{x_1}||_2^{-1}$. Fig. 1 illustrates an example with four samples and their updated positions obtained by using the gradients obtained by these loss functions. We see that for the first three loss functions, a larger learning rate may cause $\mathbf{x_1}$ and $\mathbf{x_2}$ to move away from each other. Further, in the first two cases, involving triplets, $\mathbf{y_2}$ remains unused and unchanged.

3.2 Complexity Analysis

Table 2 lists the orders of computations per iteration for computing various loss functions. B_S denotes the batch size, N is the number of samples per class, and n controls the number of generated points. It can be seen that GradML is one of the losses with the lowest complexity.

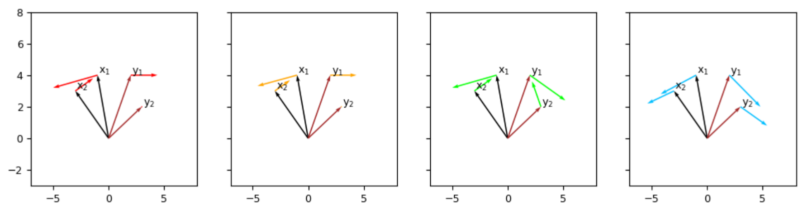

Figure 1: An example showing four samples and the gradients obtained by (a) triplet, (b) DR-triplet, (c) LS, and (d) GradML losses, used to obtain the updated positions of the samples.

Table 2: Comparison of computations per iteration for various ML-based loss functions.

Methods	Computations per Iteration
GradML, Proxy NCA, Triplet	$O(B_S)$
HPHNTriplet, LS	$O((B_S - N)B_S)$
Binomial	$O(B_S^2)$
EE	$O(n^2 B_S^2)$
MS	$O(N(B_S - N)B_S)$

3.3 Comparison with Other Methods

We compare GradML with several other approaches - the ones discussed in Section 1, and losses like histogram (33), binomial deviance (BinomDev) (33). For methods presented in Section 1.1, triplet loss is used. The results (with $k = 2$ and $w = 1$) are shown in Table 3. We see that GradML outperforms all other methods on the CUB dataset but shows weak performance on Cars196 and SOP.

Table 3: Clustering and retrieval performance of various methods for the three benchmark datasets. **Bold** numbers indicate the best values.

	CUB-200-2011					Cars196					SOP				
Method	NMI	F_1	R@1	R@2	R@4	NMI	F_1	R@1	R@2	R@4	NMI	F_1	R@1	R@10	R@100
Triplet	49.8	15.0	35.9	47.7	59.1	52.9	17.9	45.1	57.4	69.7	86.3	20.2	53.9	72.1	85.7
DAML	51.3	17.6	37.6	49.3	61.3	56.5	22.9	60.6	72.5	82.5	87.1	22.3	58.1	75.0	88.0
Semi-hard	53.4	17.9	40.6	52.3	64.2	55.7	22.4	53.2	65.4	74.3	86.7	22.1	57.8	75.3	88.1
DE-DSP	53.7	19.8	41.0	53.2	64.8	55.0	22.3	59.3	71.3	81.3	87.4	22.7	58.2	75.8	88.4
HDML	55.1	21.9	43.6	55.8	67.7	59.4	27.2	61.0	72.6	80.7	87.2	22.5	58.5	75.5	88.3
Dis-wtd	56.3	25.4	44.1	57.5	70.1	58.3	25.4	59.4	72.3	81.6	87.9	23.4	58.9	77.2	89.6
EE	55.7	22.4	44.3	57.0	68.1	60.3	25.1	57.2	70.5	81.3	87.4	24.8	62.4	79.0	91.0
SmartMin	58.1	-	45.9	57.7	69.6	58.2	-	56.1	68.3	78.0	-	-	-	-	-
StructClust	59.2	-	48.2	61.4	71.8	59.0	-	58.1	70.6	80.3	89.5	-	67.0	83.7	93.2
SymmSyn	59.6	26.2	51.4	63.0	74.4	62.4	31.8	69.7	78.7	86.1	88.9	30.6	65.7	81.4	91.7
HDC	-	-	53.6	65.7	77.0	-	-	**73.7**	**83.2**	**89.5**	-	-	69.5	84.4	92.8
DR	-	-	54.2	66.1	72.5	-	-	-	-	-	-	-	-	-	-
LS	56.4	22.6	46.9	59.8	71.2	57.8	25.1	59.9	70.4	79.6	87.2	25.3	62.6	80.9	91.2
HPHNtri	58.1	24.2	48.3	61.9	73.0	57.4	22.6	60.3	73.4	83.5	**91.4**	**43.3**	**75.5**	**88.8**	95.4
DAML (LS)	59.5	26.6	49.0	62.2	73.7	63.1	**31.9**	72.5	82.1	88.5	89.1	31.7	66.3	82.8	92.5
Proxy NCA	59.5	-	49.2	61.9	67.9	**64.9**	-	73.2	82.4	86.4	-	-	-	-	-
BinomDev	-	-	50.3	61.9	72.6	-	-	-	-	-	-	-	65.5	82.3	92.3
MS	59.3	26.0	50.9	63.0	74.1	63.3	31.7	71.0	80.8	87.5	89.3	33.7	75.0	88.7	**95.7**
Histogram	-	-	52.8	64.4	74.7	-	-	-	-	-	-	-	63.9	81.7	92.2
Ours	**60.4**	**26.8**	**54.7**	**67.0**	**77.5**	40.9	10.8	45.3	56.9	68.4	84.8	14.9	52.4	68.5	82.1

This can be attributed to data dependent properties. It appears that a global hyperparameter w (11) is not sufficient to capture all the inter-class variances in SOP, Cars196 datasets, i.e. in these datasets the inter-class variance between all classes might not be the same. A small inter-class variance between a pair of classes would require a large w in the loss and vice-versa. We believe that in order to calculate w in such instances, we need to use information of the pair of classes involved.

4 Conclusion

We propose a novel loss for deep metric learning (ML) that focuses on the direction of movement of the data points in the embedding space. This is done by formulating the loss function using gradients of the loss. We observe that the performance of our loss is influenced by data-dependent properties. For datasets where inter-class variances are similar across all pairs of classes our we see multiple benefits of our approach: considerable boost in performance and a lower computational complexity.

References

[1] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 539–546, 2005.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[3] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep metric learning for person re-identification," in *International Conference on Pattern Recognition*, pp. 34–39, 2014.

[4] W. Li, R. Zhao, T. Xiao, and X. Wang, "DeepReID: Deep filter pairing neural network for person re-identification," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 152–159, 2014.

[5] J. Zhou, P. Yu, W. Tang, and Y. Wu, "Efficient online local metric adaptation via negative samples for person re-identification," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 2439–2447, 2017.

[6] F. Wang, W. Zuo, L. Lin, D. Zhang, and L. Zhang, "Joint learning of single-image and cross-image representations for person re-identification," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1288–1296, 2016.

[7] J. Wang *et al.*, "Learning fine-grained image similarity with deep ranking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1386–1393, 2014.

[8] A. Gordo *et al.*, "Deep image retrieval: Learning global representations for image search," in *European Conference on Computer Vision (ECCV)*, (Cham), pp. 241–257, Springer International Publishing, 2016.

[9] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 1735–1742, 2006.

[10] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, p. 207–244, 2009.

[11] H. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4004–4012, 2016.

[12] K. Sohn, "Improved deep metric learning with multi-class N-pair loss objective," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[13] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, "Deep metric learning with angular loss," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[14] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5022–5030, 2019.

[15] 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 (ICCV)*, Oct 2017.

[16] N. Aziere and S. Todorovic, "Ensemble deep manifold similarity learning using hard proxies," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[17] 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 (ICCV)*, Oct 2017.

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

[19] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, “The Caltech-UCSD Birds-200-2011 Dataset,” tech. rep., 2011.

[20] R. Manmatha, C. Wu, A. Smola, and P. Krahenbuhl, “Sampling matters in deep embedding learning,” in *IEEE International Conference on Computer Vision (ICCV)*, pp. 2859–2867, 10 2017.

[21] B. Harwood, V. Kumar B.G., G. Carneiro, I. Reid, and T. Drummond, “Smart mining for deep metric learning,” in *IEEE International Conference on Computer Vision (ICCV)*, pp. 2840–2848, 2017.

[22] Y. Yuan, K. Yang, and C. Zhang, “Hard-aware deeply cascaded embedding,” in *IEEE International Conference on Computer Vision (ICCV)*, pp. 814–823, 11 2017.

[23] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy, “Learnable structured clustering framework for deep metric learning,” *CoRR*, vol. abs/1612.01213, 2016.

[24] Y. Duan, L. Chen, J. Lu, and J. Zhou, “Deep embedding learning with discriminative sampling policy,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[25] Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou, “Deep adversarial metric learning,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2780–2789, 2018.

[26] W. Zheng, Z. Chen, J. Lu, and J. Zhou, “Hardness-aware deep metric learning,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 72–81, 2019.

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

[28] G. Gu and B. Ko, “Symmetrical synthesis for deep metric learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 10853–10860, Apr. 2020.

[29] B. Vasudeva, P. Deora, S. Bhattacharya, U. Pal, and S. Chanda, “Loop: Looking for optimal hard negative embeddings for deep metric learning,” *arXiv preprint arXiv:2108.09335*, 2021.

[30] D. D. Mohan, N. Sankaran, D. Fedorishin, S. Setlur, and V. Govindaraju, “Moving in the right direction: A regularization for deep metric learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[31] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. USA: Cambridge University Press, 2008.

[32] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, “3d object representations for fine-grained categorization,” in *IEEE International Conference on Computer Vision Workshops (ICCVW)*, p. 554–561, 2013.

[33] E. Ustinova and V. Lempitsky, “Learning deep embeddings with histogram loss,” in *Advances in Neural Information Processing Systems (NeurIPS)* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, pp. 4170–4178, Curran Associates, Inc., 2016.

[34] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

[35] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2017.

Appendix

4.1 Implementation Details

We use PyTorch to implement our proposed approach. The experiments were carried out using one NVIDIA GeForce RTX 2080 Ti GPU with 11 GB memory. We use an ImageNet ILSVRC (34) pre-trained ResNet50 [check this one out] with frozen Batch-Normalization layers as the feature extracting network. We use 512-dimensional feature embedding vectors. For pre-processing, we randomly resize and crop images to 224×224 and horizontally flip them left or right for training, and center crop to the same size for testing. We use the Adam optimizer (35) for training with a learning rate of 10^{-7} and a constant weight decay of 4×10^{-4}. We use a batch size of 32.

4.2 Dataset details

Table 4: Summary of the three benchmark datasets used for evalution.

	Training		Testing		Total	
Dataset	No. of classes	No. of images	No. of classes	No. of images	No. of classes	No. of images
CUB-200-2011	100	5864	100	5924	200	11,788
Cars 196	98	8054	98	8131	196	16,185
SOP	11,318	59,551	11,316	60,502	22,634	120,053

4.3 Effect of Hyperparameters

In this section, we carry out two experiments to observe the effect of k and w on the performance of our approach.

Effect of k: To check the effect of k, we set w as 0.5 (this value is obtained by equating the sum of the weights of the positives to that the of the negatives). We vary k from 1 to 4 and the values are listed in Table 5. It can be seen that higher values are centered around k=2.5. To further illustrate this, we plot the variation in NMI and R@1 in Figs. 2(a) and (b), respectively. It can be seen that a higher range of values occurs when k lies between 2 and 3.

Table 5: Variation in clustering and retrieval performance for different values of k. **Bold** and $\underline{\text{underlined}}$ numbers indicate the best and second-best values, respectively.

k	1	1.5	2	2.25	2.5	2.75	3	4
NMI	58.71	58.63	59.20	$\underline{59.21}$	$\underline{59.21}$	**59.40**	59.29	59.08
F1	24.85	23.80	24.47	24.73	$\underline{24.88}$	**24.98**	24.87	24.76
R@1	53.39	53.63	$\underline{54.12}$	54.10	**54.14**	53.88	53.92	53.34
R@2	**66.17**	65.72	66.10	66.12	$\underline{66.15}$	66.04	66.12	65.68
R@4	76.82	76.76	76.50	76.60	76.54	$\underline{76.86}$	**76.99**	76.81

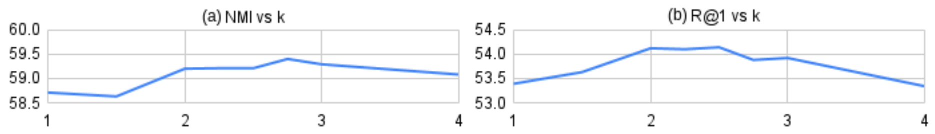

Figure 2: Variation in (a) NMI, (b) R@1 for different values of k.

Table 6: Variation in clustering and retrieval performance for different values of k and w. **Bold** and $\underline{\text{underlined}}$ numbers indicate the best values for each k and the overall best values, respectively.

k	2			2.5			3		
w	0.5	1	2	0.5	1	2	0.5	1	2
NMI	59.20	$\underline{\textbf{60.35}}$	59.48	59.21	**59.89**	59.38	59.29	**59.83**	59.36
F1	24.47	$\underline{\textbf{26.80}}$	25.82	24.88	**26.00**	25.91	24.87	26.01	**26.42**
R@1	54.12	**54.73**	54.15	54.14	$\underline{\textbf{54.90}}$	54.25	53.92	**54.30**	53.65
R@2	66.10	$\underline{\textbf{66.95}}$	66.53	66.15	**66.73**	66.51	**66.12**	66.10	65.87
R@4	76.50	$\underline{\textbf{77.48}}$	77.23	76.54	**77.40**	77.07	76.99	**77.01**	76.79

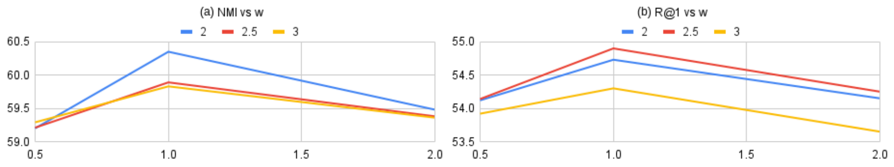

Figure 3: Variation in (a) NMI, (b) R@1 for different values of w for $k = 2, 2.5, 3$.

Effect of w: To check the effect of w, we set it as 0.5, 1, and 2. We also vary k between 2 and 3, in keeping with the observations of the previous experiments. The values are reported in Table 6. It can be seen that for a fixed k, the best values occur for w=1, and k=2 results in overall best performance, other than R@1. This can also be seen from the plots of NMI and R@1, shown in Figs. 3(a) and (b), respectively.