17th International Conference on Machine Vision Applications (MVA)
Fully Online, July 25–27, 2021.

Analysis of Evaluation Metrics with the Distance between Positive Pairs and Negative Pairs in Deep Metric Learning

Hajime Oi†, Rei Kawakami‡, Takeshi Naemura†
† The University of Tokyo, ‡ Tokyo Institute of Technology
{hoi,naemura}@hc.ic.i.u-tokyo.ac.jp, reikawa@c.titech.ac.jp

Abstract

Deep metric learning (DML) acquires embeddings via deep learning, where distances among samples of the same class are shorter than those of different classes. The previous DML studies proposed new metrics to overcome the issues of general metrics, but they have the following two problems; one is that they consider only a small portion of the whole distribution of the data, and the other is that their scores cannot be directly compared among methods when the number of classes is different. To analyze these issues, we consider the histograms of the inner products between arbitrary positive pairs and those of negative pairs. We can evaluate the entire distribution by measuring the distance between the two histograms. By normalizing the histograms by their areas, we can also cancel the effect of the number of classes. In experiments, visualizations of the histograms revealed that the embeddings of the existing DML methods do not generalize well to the validation set. We also confirmed that the evaluation of the distance between the positive and negative histograms is less affected by the variation in the number of classes compared with Recall@1 and MAP@R.

1 Introduction

Metric learning aims to learn an embedding space in which the similarity of data corresponds to the distance between vectors; i.e., the more similar the data are, the shorter the distance between them in the space becomes. In the case of images, the embedding space is acquired such that the distances are shorter for images of the same class and longer for images of different classes. Many applications can benefit from a better evaluation of the semantic distance, such as content-based image retrieval and face recognition. Metric learning that relies on deep learning is called deep metric learning (DML), and its performance has been higher than those based on non-linear kernels and classifiers.

The literature on DML has few discussions on evaluation metrics. One such metric, normalized mutual information (NMI) [9], is used to evaluate clustering; it examines the information gain of the estimated labels for the true labels. Clustering is necessary to estimate the labels in the learned space; thus, NMI depends on the clustering accuracy. Another evaluation metric, Recall@K [5], is used in information retrieval; it represents the percentage of total data in the dataset containing at least one positive among their K nearest neighbors; thus, Recall@K does not consider the number of positives in the neighbors or how close each positive is to the query. R-Precision and Mean Average Precision@R (MAP@R) were used in [10] as alternatives, but these also have two issues. First, they focus only on the accuracy of R nearest neighbors, which is a tiny part of the whole data distribution. Second, when the number of classes increases, their scores probably decrease because the proportion of negatives in the R nearest neighbors increases; therefore, their scores cannot be directly compared if the number of classes is different.

To analyze the issues of evaluation metrics in DML, we consider the distribution of inner products between positive pairs as well as that of negative pairs. We also measure the distance between these two distributions by using the Jensen-Shannon divergence (JSD) to evaluate the performance of DML on the entire distribution. Additionally, we normalize the distribution by its area to cancel out the variation in the number of classes.

In the experiments, we observed changes in the mean value of the inner products between positive pairs and those of negative pairs for the training and validation sets during training. We also determined whether these changes are consistent with those of JSD, Recall@1, and MAP@R. As the training progressed, the average value of the inner products between positives in the training set increased, but those in the validation set remained constant or slightly decreased, suggesting that the learned space overfits the training set and is not well generalized. The changes in JSD were similar to those of the existing metrics, indicating that it does not show any clear advantages in using the entire distribution compared with R nearest neighbors with the examined dataset. However, JSD showed signs of overfitting to the training set and synchronized with it. It starts with the same value for the training and validation sets, and later, only the value for the training set increases. Since the numbers of classes in the training and validation sets are different, other metrics cannot compare their scores directly or capture the overfitting.

The experiments with different data splitting methods also revealed problems with those used in the previous studies.

2 Preliminaries

R-Precision and MAP@R Musgrave *et al.* [10] propose to use R-Precision and MAP@R to evaluate DML, where R is the total number of positives for each query. These two measures are related to Precision@K, which refers to the percentage of positives in the K nearest neighbors. R-Precision is the average of Precision@R over all of the queries. MAP@R is the

average of Average Precision@R (AP@R) over all of the queries, where AP@R is given by [1]

$$AP@R = \frac{1}{R}\sum_{k=0}^{R} Precision@k \times p(k) \tag{1}$$

$$p(k) = \begin{cases} 1 & \text{if } k\text{-th nearest neighbor is positive,} \\ 0 & \text{if } k\text{-th nearest neighbor is negative.} \end{cases}$$

MAP@R and AP@R have two problems. The first is that they do not evaluate the distribution of the entire data; namely, they ignore the improvement when a correct positive sample not included in its R neighbors approaches the query. The second is that the scores are sensitive to changes in the number of classes. In other words, when the number of classes increases, the possibility of having negatives in the R nearest neighbors increases, and the score decreases because all the added classes are negative.

Loss functions in DML Early losses in DML attempted to obtain the global metric space by optimizing distances within small local groups [3, 17, 4], which was inefficient because of the enormous number of possible group combinations. One way to meet this challenge is to prepare proxies for each class and to put constraints only on the distances between the data and the proxies. Most of the losses in this category are similar to the softmax loss.

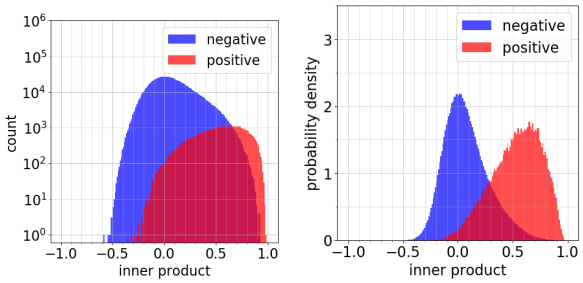
Here, we will review the softmax loss and the normalized softmax loss (NSL) to clarify the distance and inner products between any pair. Let us define x_i to be the input to the last fully connected (FC) layer of the i-th sample; it is a d-dimensional vector belonging to the y_i-th class. The softmax function, which also serves as a loss function in metric learning, outputs a vector whose j-th element represents the predicted probability that x_i belongs to class j. The FC layer consists of a weight matrix $\mathbf{W} \in \mathbb{R}^{d\times C}$ and a bias vector $\mathbf{b} \in \mathbb{R}^C$. W_j denotes the j-th column of $\mathbf{W}$, and C is the number of classes. The softmax loss maximizes the y_i-th element of the final vector as it is the true class of x_i.

$$Softmax = -\log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{C} e^{W_j^T x_i + b_j}} \tag{2}$$

In DML, W_j is often treated as a proxy of the j-th class and b is ignored; therefore, the inner product of x_i and W_j, both of which are ℓ^2-normalized, is considered to be a distance [2, 12, 6]. The simplest loss [16, 19] is called the normalized softmax loss (NSL) in this paper, following [10]:

$$NSL = -\log \frac{e^{s\bar{W}_{y_i}^T \bar{x}_i}}{\sum_{j=1}^{C} e^{s\bar{W}_j^T \bar{x}_i}}, \tag{3}$$

where $\bar{\cdot}$ is ℓ^2-normalized and s is a scaling parameter.



(a) Before normalization (b) After normalization

Figure 1: Distribution of inner products before and after normalization. The histogram in red shows the inner products between every positive pair after training, and the histogram in blue shows those of the negative pairs. Best viewed in color.

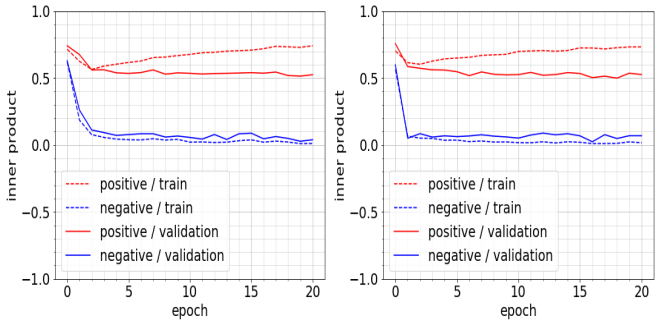
3 Distribution of inner products between positives and negatives

In many DML losses, the inner product between an embedded vector of a given sample and a proxy vector for its class, which are ℓ^2-normalized, are evaluated and maximized during training. Fig. 1 (a) depicts two histogram distributions of inner products; the one in red shows the inner products between every positive pair and the one in blue is for those between every negative pair in the dataset. It is clear that the number of negative pairs is much larger than the number of positive pairs. If the number of classes is c and the average number of samples per class is n, the number of positive pairs is $O(cn^2)$ and the number of negative pairs is $O((cn)^2)$; thus, the latter is about c times larger. This imbalance may make it impossible for metrics based on rankings arranged in order of increasing inner product, such as MAP@R, to evaluate the true performance of models.

To resolve this issue, we consider a distribution that is normalized so that the areas of the histograms of the positive and negative pairs are each 1.0, as shown in Fig. 1 (b). In this way, we cancel out the imbalance in the number by normalization and obtain distributions independent of the number of classes.

The two histograms in Fig. 1 (b) can be regarded as discrete probability distributions since they sum to 1. We can evaluate the distance between them by using tools to evaluate the distance between probability distributions, such as the Kullback-Leibler divergence (KLD). KLD, as shown in Eq. (4), represents the difference between the true probability distribution P and its predicted probability distribution Q, and it is equal to the cross-entropy of P and Q minus the self-information content of P. In general, KLD is not symmetric between P and Q, i.e., $D_{KL}(P||Q) \neq D_{KL}(Q||P)$. Thus, we employ the JSD as shown in Eq. (5), which a modification to KLD that is symmetric. JSD is obtained by adding the probability distribution M averaged over P and Q and averaging the KLD of P and M and the KLD of Q and M. The minimum

[1] Note that Eq. (3) in [10] should be a typo.



(a) Cars196 [7] (b) CUB-200-2011 [15]

Figure 2: Mean value of inner products between every positive pair and those between every negative pair in the training and validation sets. Overfitting occurred for positive pairs during training.

value of JSD is 0 when P and Q coincide perfectly, and its maximum is 1 when P and Q do not overlap at all.

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}, \tag{4}$$

$$D_{JS}(P||Q) = \frac{D_{KL}(P||M) + D_{KL}(Q||M)}{2}, \tag{5}$$

$$\left(M(i) = \frac{P(i) + Q(i)}{2}. \right)$$

4 Experiments

We implemented our model in the PyTorch framework [11].

Dataset Our evaluation used the Cars196 [7], CUB-200-2011 [15] datasets. The validation set was prepared for model selections aligned with [10]; therefore, the datasets were split into training, validation, and test sets in a ratio of 4:1:5 in the number of classes.

We also conducted experiments in which we split the classes in the default order or in random order. The elements of the two datasets are originally arranged in the order of their labels names; cars from the same company are placed near each other in the Cars196 dataset [7] and birds belonging to the same family and genus are near each other in the CUB-200-2011 dataset [15]. Since there seems to be many similarities among the items belonging to the same domain in the higher level classifications, there is possibly a domain bias among the training, validation, and test sets when they are divided up according to the default order of the classes. We set the random seed to 1.

Augmentation The input images in the training set were first randomly cropped in the scale-ratio range of [1/8, 1] and aspect ratio [3/4, 4/3]; then, they were resized to 299 pixels square. Though the previous studies set the size 244 pixels, we used a pre-trained model provided PyTorch framework [1] and could not choose the size. After that, they were flipped horizontally with a probability of 50 percent. Finally, the values of their RGB channels were normalized according to the

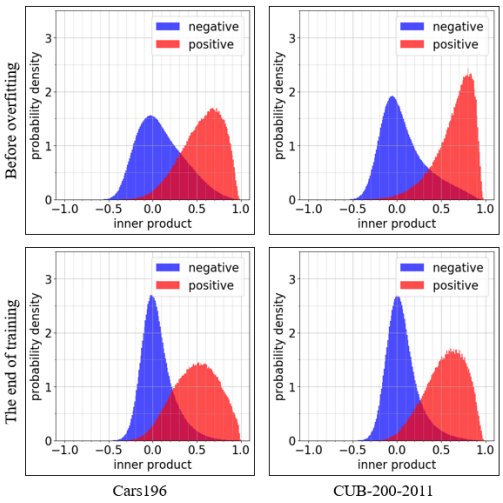


Figure 3: Comparison of the histograms in the test set before and after training with two datasets where overfitting occurs. The variances of the histograms for negative pairs decreased, and the peak is closer to 0 after training; thus, although the overfitting occurs, the network are successful to separate the negatives from each other. It is more difficult to let positives be closer.

mean = (0.485, 0.456, 0.406) and standard deviation = (0.229, 0.224, 0.225). The images in the validation and test sets were resized to 299 pixels square and then normalized in the same way as above.

Training and Evaluation We used the Inception-v3 network [14] pre-trained on ImageNet [13] as the embedding network, whose last fully-connected (FC) layer was replaced by a new one with 2064 input and 64 output dimensions. For the loss, we used the NSL expressed by Eq. 3 with $s = 10$. Training was performed for 20 epochs in all cases. The initial learning rates of the network and NSL were set to 10^{-4} and 10^{-2}, respectively, and they were tuned with the AdamW optimizer [8]. In the evaluation, we used Recall@1 and MAP@R as representatives of the metrics used in the previous studies and JSD as proposed.

Results Fig. 2 shows the mean values of the inner products in the training and validation sets. The averages of the two sets for the negative pairs were almost the same during training and converged to around 0. The negative pairs were sufficiently separated because the Euclidean distance between negative pairs is likely to be $\sqrt{2}$ for negative pairs uniformly distributed on the unit hypersphere [18]. On the other hand, although the positive mean was initially small and subsequently increased in the training set, it remained constant or slightly decreased in the validation set, indicating that overfitting occurred.

To clarify the difficulty of letting positives be closer in metric learning, we also visualize the histograms in the test set before and after the training in Fig. 3. The histograms for the positive pairs show a slight decrease of the inner products though we hope them to be larger, and this tendency is the same for the mean

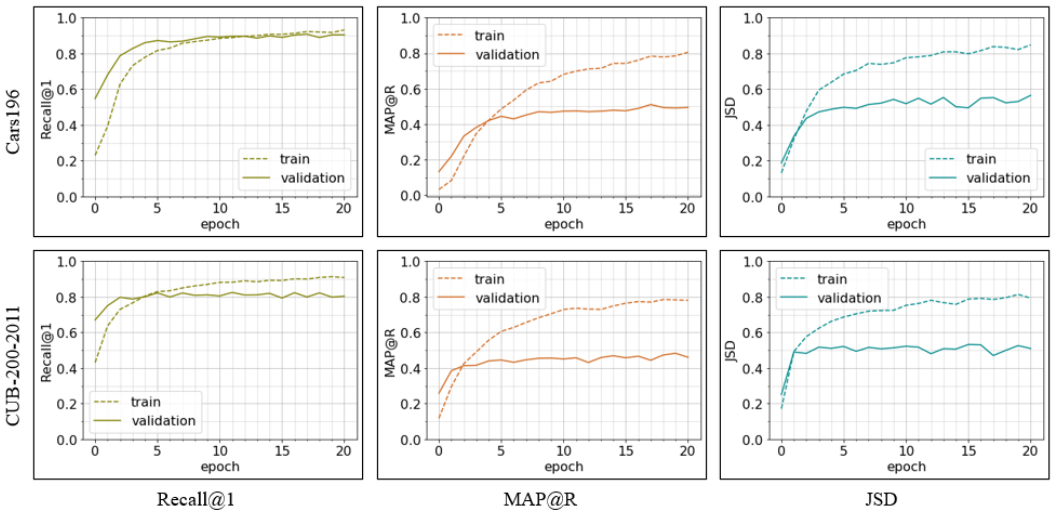


Figure 4: Scores of existing metrics and JSD for training and validation set. Though the changes in the individual metrics were similar on validation set, the scores for the training set were different from those of the validation set. On the one hand, in JSD, these score were initially almost the same; then, the training score started to exceed the validation score when the overfitting began in Fig. 2. On the other hand, regarding the metrics used in the previous studies, the validation scores were initially larger than the training scores; moreover, the large-small relationship reversed after the overfitting started.

value of them, although the inner products for training set indeed become larger as in Fig. 2. On the other hand, the variance of the distribution of negative pairs becomes smaller, and the inner products between the negatives are closer to 0 after training.

Fig. 4 depicts the dissimilarity between the metrics used in the previous studies and JSD. There was no significant difference between them on the validation set; therefore, the necessity of evaluating the whole data distribution could not be verified. By contrast, comparing the training with validation results, we can see that the scores of JSD were equal before the overfitting began in Fig. 2; then, the training scores exceeded the validation scores. For Recall@1 and MAP@R, the scores for the validation set were initially larger; moreover, the reversal of the large-small relationship was later than the start of overfitting. These results are likely due to the fact that the number of classes in the training set was about four times larger than that in the validation set, and the number of negatives was large, suggesting that the existing evaluations are easily affected by variations in the number of classes.

Finally, Table 1 reveals the differences in the scores for the test set between classes split in default order and in random order. These scores were calculated with embeddings that were extracted using the model of the epoch having the highest score for the validation set. The scores for all metrics were lower when the classes were partitioned in default order; thus, there seemed to be a domain bias between the training, validation, and test sets in this partition.

Table 1: Difference in scores of evaluation metrics on the test set among class partitioning methods. For all of the datasets, the scores were lower when the classes were split in the default order than in a random order, suggesting that there is a domain bias between the training, validation, and test sets in the default order.

Dataset	Splitting	Recall@1	MAP@R	JSD
Cars196	default	0.673	0.168	0.439
	random	0.721	0.226	0.472
CUB-200 -2011	default	0.481	0.164	0.401
	random	0.659	0.260	0.579

5 Conclusion

We analyzed DML, focusing on the histogram of inner products between positive pairs and that of negatives. Measuring the distance between the two histograms with JSD enabled us to consider all the data other than R nearest neighbors, unlike the evaluation metrics used in the previous studies. Moreover, normalization of the histograms removed the effect of the variation in the number of classes. As a result, we suggest that the existing methods are likely to overfit the distance between positives. Although we cannot verify the necessity of considering data other than R nearest neighbors, we can confirm that JSD reduces the effect of the variation in the number of classes compared with the existing metrics. In addition, there is a possibly local domain bias with the default order of labels in the datasets, suggesting that we should assign the classes randomly in the training-validation-test splitting.

One of our future tasks will be to remove the overfitting. Disentangled representations perhaps resolve this issue, each variables of which are lower dimensional features and independent of each other. We expect that the use of disentangled representations will improve class discrimination by including only features whose inter-class variances are large and intra-class variances are small.

References

[1] Inception_v3 |PyTorch. `https://pytorch.org/hub/pytorch_vision_inception_v3/`. Accessed: 2021-06-11.

[2] J. Deng, S. Zafeririou. Arcface for disguised face recognition. In *IEEE International Conference on Computer Vision Workshop*, pages 485–493, 2019.

[3] R. Hadsell *et al.* Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1735–1742, 2006.

[4] E. Hoffer, N. Ailon. Deep metric learning using triplet network. In *Similarity-Based Pattern Recognition*, pages 84–92, 2015.

[5] H. Jégou *et al.* Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[6] S. Kim *et al.* Proxy anchor loss for deep metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3235–3244, 2020.

[7] J. Krause *et al.* 3d object representations for fine-grained categorization. In *International IEEE Workshop on 3D Representation and Recognition*, 2013.

[8] I. Loshchilov, F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[9] C. D. Manning *et al. Introduction to Information Retrieval.* Cambridge University Press, 2008.

[10] K. Musgrave *et al.* A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699, 2020.

[11] A. Paszke *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. 2019.

[12] Q. Qian *et al.* Softtriple loss: Deep metric learning without triplet sampling. In *IEEE International Conference on Computer Vision*, pages 6449–6457, 2019.

[13] O. Russakovsky *et al.* ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[14] C. Szegedy *et al.* Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[15] C. Wah *et al.* The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

[16] F. Wang *et al.* Normface: L2 hypersphere embedding for face verification. In *ACM International Conference on Multimedia*, page 1041–1049, 2017.

[17] J. Wang *et al.* Learning fine-grained image similarity with deep ranking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.

[18] C. Wu *et al.* Sampling matters in deep embedding learning. In *IEEE International Conference on Computer Vision*, pages 2859–2867, 2017.

[19] A. Zhai, H. Wu. Classification is a strong baseline for deep metric learning. In *British Machine Vision Conference*, page 91, 2019.