Learning Intra-Batch Connections for Deep Metric Learning

Jenny Seidenschwarz [1] **Ismail Elezi** [1] **Laura Leal-Taixé** [1]

Abstract

The goal of metric learning is to learn a function that maps samples to a lower-dimensional space where similar samples lie closer than dissimilar ones. Particularly, deep metric learning utilizes neural networks to learn such a mapping. Most approaches rely on losses that only take the relations between pairs or triplets of samples into account, which either belong to the same class or two different classes. However, these methods do not explore the embedding space in its entirety. To this end, we propose an approach based on message passing networks that takes all the relations in a mini-batch into account. We refine embedding vectors by exchanging messages among all samples in a given batch allowing the training process to be aware of its overall structure. Since not all samples are equally important to predict a decision boundary, we use an attention mechanism during message passing to allow samples to weigh the importance of each neighbor accordingly. We achieve state-of-the-art results on clustering and image retrieval on the CUB-200-2011, Cars196, Stanford Online Products, and In-Shop Clothes datasets. To facilitate further research, we make available the code and the models at https://github.com/dvl-tum/intra_batch.

1. Introduction

Metric learning is a widely popular technique that constructs task-specific distance metrics by learning the similarity or dissimilarity between samples. It is often used for object retrieval and clustering by training a deep neural network to learn a mapping function from the original samples into a new, more compact, embedding space. In that embedding space, samples coming from the same class should be closer than samples coming from different classes.

To learn the mapping function, current approaches utilize siamese networks (Bromley et al., 1994), typically trained using loss functions that measure distances between pairs of samples of the same class (positive) or different classes (negative). Contrastive loss (Bromley et al., 1994) minimizes the distance of the feature embeddings for a positive pair, and maximizes their distance otherwise. Triplet loss (Schultz & Joachims, 2003; Weinberger & Saul, 2009) takes a triplet of images and pushes the embedding distance between an anchor and a positive sample to be smaller than the distance between the same anchor and a negative sample by a given margin. While the number of possible image pairs and triplets in a dataset of size n is $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$, respectively, the vast majority of these pairs (or triplets) are not informative and do not contribute to the loss. This leads to slow convergence and possible overfitting when the pairs (triplets) are not appropriately sampled. Perhaps more worryingly, because these losses are focused on pairs (triplets), they are unable to consider the global structure of the dataset resulting in lower clustering and retrieval performance. To compensate for these drawbacks, several works resort to training tricks like intelligent sampling (Ge et al., 2018; Manmatha et al., 2017), multi-task learning (Zhang et al., 2016), or hard-negative mining (Schroff et al., 2015; Xuan et al., 2020a). Recently, researchers started exploring the global structure of the embedding space by utilizing rank-based (Çakir et al., 2019; He et al., 2018a; Revaud et al., 2019) or contextual classification loss functions (Çakir et al., 2019; Elezi et al., 2020; He et al., 2018a; Revaud et al., 2019; Sohn, 2016; Song et al., 2016; Zheng et al., 2019). The Group Loss (Elezi et al., 2020) explicitly considers the global structure of a mini-batch and refines class membership scores based on feature similarity. However, the global structure is captured using a handcrafted rule instead of learning, hence its refinement procedure cannot be adapted depending on the samples in the mini-batch.

1.1. Contributions

In this work, we propose a fully learnable module that takes the global structure into account by refining the embedding feature vector of each sample based on *all* intra-batch relations. To do so, we utilize message passing networks (MPNs) (Gilmer et al., 2017). MPNs allow the samples in a mini-batch to communicate with each other, and to

[1]Department of Computer Science, Technical University of Munich, Munich, Germany. Correspondence to: Jenny Seidenschwarz <j.seidenschwarz@tum.de>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

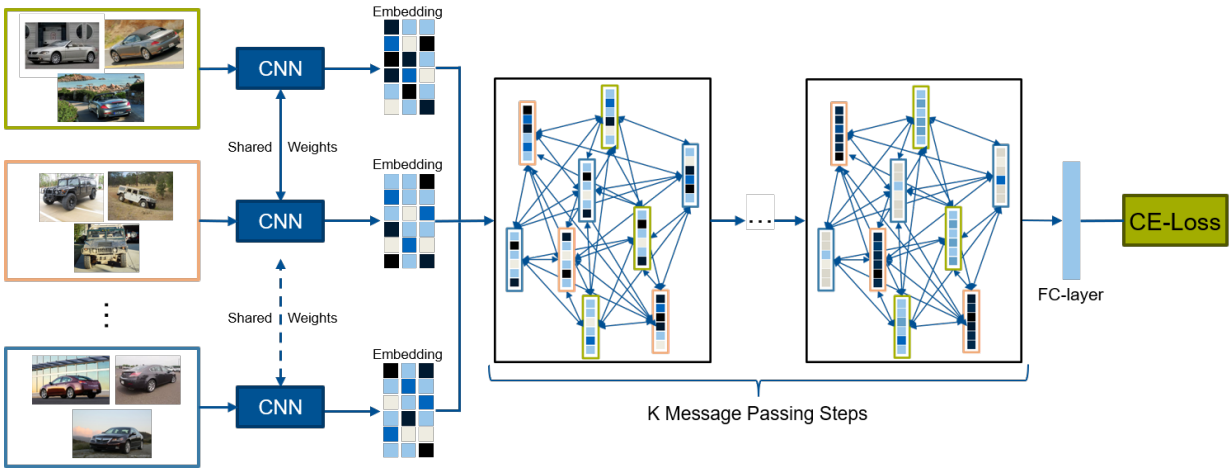

Figure 1. Overview of our proposed approach. Given a mini-batch consisting of N classes, each of them having P images, we initialize the embedding vectors using a backbone CNN. We then construct a fully connected graph that refines their initial embeddings by performing K message-passing steps. After each step, the embeddings of the images coming from the same class become more similar to each other and more dissimilar to the embeddings coming from images that belong to different classes. Finally, we apply Cross-Entropy loss and we backpropagate the gradients to update the network.

refine their feature representation based on the information taken from their neighbors. More precisely, we use a convolutional neural network (CNN) to generate feature embeddings. We then construct a fully connected graph where each node is represented by the embedding of its corresponding sample. In this graph, a series of message passing steps are performed to update the node embeddings. Not all samples are equally important to predict decision boundaries, hence, we allow each sample to weigh the importance of neighboring samples by using a dot-product self-attention mechanism to compute aggregation weights for the message passing steps.

To draw a parallelism with the triplet loss, our MPN formulation would allow samples to choose their own triplets which are best to make a prediction on the decision boundary. Unlike the triplet loss though, we are not limited to triplets, as each sample can choose to attend over all other samples in the mini-batch. By training the CNN and MPN in an end-to-end manner, we can directly use our CNN backbone embeddings during inference to perform image retrieval and clustering. While this reaches state-of-the-art results without adding any computational overhead, we also show how to further boost the performance by using the trained MPN at test time, constructing the batches based on k-reciprocal nearest neighbor sampling (Zhong et al., 2017).

Our **contribution** in this work is three-fold:

- We propose an approach for deep metric learning that computes sample embeddings by taking into account all intra-batch relations. By leveraging message passing networks, our method can be trained end-to-end.
- We perform a comprehensive robustness analysis showing the stability of our module with respect to the choice of hyperparameters.
- We present state-of-the-art results on CUB-200-2011 (Wah et al., 2011), Cars196 (Krause et al., 2013), Stanford online Products (Song et al., 2016) and In-Shop Clothes (Liu et al., 2016) datasets.

2. Related Work

Metric Learning Losses. Siamese neural networks were first proposed for representation learning in (Bromley et al., 1994). The main idea is to use a CNN to extract a feature representation from an image and using that representation, or embedding, to compare it to other images. In (Chopra et al., 2005), the contrastive loss was introduced to train such a network for face verification. The loss minimizes the distance between the embeddings of image pairs coming from the same class and maximizes the distance between image pairs coming from different classes. In parallel, researchers working on convex optimization developed the triplet loss (Schultz & Joachims, 2003; Weinberger & Saul, 2009) which was later combined with the expressive power of CNNs, further improving the solutions on face verification (Schroff et al., 2015). Triplet loss extends contrastive loss by using a triplet of samples consisting of an anchor, a positive, and a negative sample, where the loss is defined to make the distance between the anchor and the positive smaller than the distance between the anchor and the negative, up to a margin. The concept was later generalized to N-Pair loss (Sohn, 2016), where an anchor and a positive sample are compared to $N-1$ negative samples at the same

time. In recent years, different approaches based on optimizing other qualities than the distance, such as clustering (Law et al., 2017; McDaid et al., 2011) or angular distance (Wang et al., 2017), have shown to reach good results.

Sampling and Ensembles. Since computing the loss of all possible triplets is computationally infeasible even for moderately-sized datasets and, furthermore, based on the knowledge that the majority of them are not informative (Schroff et al., 2015), more researchers have given attention to intelligent *sampling*. The work of (Manmatha et al., 2017) showed conclusive evidence that the design of smart sampling strategies is as important as the design of efficient loss functions. In (Ge et al., 2018), the authors propose a hierarchical version of triplet loss that embeds the sampling during the training process. More recent techniques continue this line of research by developing new sampling strategies (Duan et al., 2019; Xuan et al., 2020a;b) while others introduce new loss functions (Wang et al., 2019a; Xu et al., 2019). In parallel, other researchers investigated the usage of ensembles for deep metric learning, unsurprisingly finding out that ensembles outperform single networks trained on the same loss (Kim et al., 2018; Opitz et al., 2017; Sanakoyeu et al., 2019; Xuan et al., 2018; Yuan et al., 2017).

Global Metric Learning Losses. Most of the mentioned losses do not consider the global structure of the mini-batch. The work of (Movshovitz-Attias et al., 2017) proposes to optimize the triplet loss on a space of triplets different from the one of the original samples, consisting of an anchor data point and similar and dissimilar learned proxy data points. These proxies approximate the original data points so that a triplet loss over the proxies is a tight upper bound of the loss over the original samples. The introduction of proxies adds additional contextual knowledge that shows to significantly improve triplet loss. The results of this approach were significantly improved by using training tricks (Teh et al., 2020) or generalizing the concept of proxy triplets to multiple proxy anchors (Kim et al., 2020; Zhu et al., 2020). In (Duan et al., 2018) the authors generate negative samples in an adversarial manner, while in (Lin et al., 2018) a deep variational metric learning framework was proposed to explicitly model the intra-class variance and disentangle the intra-class invariance. In the work of (Wang et al., 2019b), a non-proxy contextual loss function was developed. The authors propose a loss function based on a ranking distance that considers all the samples in the mini-batch

Classification Losses for Metric Learning. A recent line of work (Zhai & Wu, 2019; Zheng et al., 2019) is showing that a carefully designed classification loss function can rival, if not outperform, triplet-based functions in metric learning. This has already been shown for multiple tasks such as hashing (binary-embedding) (He et al., 2018a), landmark detection (He et al., 2018b; Revaud et al., 2019), few-shot learning (Çakir et al., 2019), and person re-identification (Alemu et al., 2019; Zhao et al., 2019). In metric learning, SoftTriple loss (Qian et al., 2019) develops a classification loss where each class is represented by K centers. In the same classification spirit, the Group Loss (Elezi et al., 2020) replaces the softmax function with a contextual module that considers all the samples in the mini-batch at the same time.

Message Passing Networks. Recent works on message passing networks (Gilmer et al., 2017) and graph neural networks (Battaglia et al., 2018; Kipf & Welling, 2017) have been successfully applied to problems such as human action recognition (Guo et al., 2018), visual question answering (Narasimhan et al., 2018) or tracking (Brasó & Leal-Taixé, 2020). Given a graph with some initial features for nodes and edges, the main idea behind these models is to embed nodes and edges into representations that take into account not only the node's own features but also those of its neighbors in the graph, as well as the graphs overall topology. The attention-based Transformers (Vaswani et al., 2017; Xu et al., 2015), which can be seen as message passing networks, have revolutionized the field of natural language processing, and within the computer vision, have shown impressive results in object detection (Carion et al., 2020).

Closely related to message passing networks, (Elezi et al., 2020) considered contextual information for metric learning based on the similarity (dissimilarity) between samples coming from the same class (respectively from different classes). However, they use a handcrafted rule as part of their loss function that only considers the label preferences (Elezi et al., 2018). In contrast, based on message passing networks, we develop a novel learnable model, where each sample uses learned attention scores to choose the importance of its neighbors, and based on this information, refines its own feature representation.

3. Methodology

The goal of the message passing steps is to exchange information between all samples in the mini-batch and to refine the feature embeddings accordingly. Note that this approach is very different from label-propagation methods as used in (Elezi et al., 2020), where samples exchange information only on their label preferences, information which only implicitly affects the choice of their final feature vectors.

In our proposed method, each sample exchanges messages with all the other samples in the mini-batch, regardless of whether the samples belong to the same class or not. In this way, *our method considers both the intra-class and inter-class relations between all samples in the mini-batch*, allowing our network to receive information about the overall structure of the mini-batch. We can use cross-entropy loss to train our network since the information of the mini-

batch is already contained in the refined individual feature embeddings.

3.1. Overview

In Figure 1, we show an overview of our proposed approach. We compute feature vectors for each sample as follows:

1. Generate initial embedding feature vectors using a CNN and construct a fully connected graph, where each node represents a sample in the mini-batch.
2. Perform message-passing between nodes to refine the initial embedding feature vectors by utilizing dot-product self-attention.
3. Perform classification and optimize both the MPN and the backbone CNN in an end-to-end fashion using cross-entropy loss on the refined node feature vectors.

3.2. Feature Initialization and Graph Construction

The global structure of the embedding space is modeled by a graph $\mathcal{G} = (V, E)$, where V represents the nodes, i.e., all images in the training dataset, and E the edges connecting them. An edge represents the importance of one image to the other, expressed, for example, by their similarity. During training, we would ideally take the graph of the whole dataset into account, but this is computationally infeasible. Therefore, we construct mini-batches consisting of n randomly sampled classes with p randomly chosen samples per class. Each sample in the mini-batch is regarded as a node in a mini-batch graph $\mathcal{G}_{\mathcal{B}} = (V_B, E_B)$. Unlike CNNs that perform well on data with an underlying grid-like or Euclidean structure (Bronstein et al., 2017), graphs have a non-euclidean structure. Thus, to fully explore the graph-like structure, we model the mini-batch relations using MPNs.

More precisely, we use a backbone CNN to compute the initial embeddings $\boldsymbol{f} \in \mathcal{R}^d$ for all samples in a mini-batch, where d is their embedding dimension. To leverage all relations in the batch, we utilize a fully connected graph, where every node with initial node features $\boldsymbol{h}_i^0 = \boldsymbol{f}$ is connected to all the other nodes in the graph (see Figure 2 in the upper left corner).

3.3. Message Passing Network

In order to refine the initial feature vectors based on the contextual information of the mini-batch, we use message passing to exchange information between single nodes, i.e., between samples of the mini-batch. To this end, we utilize MPNs with graph attention (Velickovic et al., 2018) for deep metric learning. It should be noted that the following formulation is equivalent to the Transformers architecture (Vaswani et al., 2017), which can be seen as a fully connected graph attention network (Velickovic et al., 2018).

Passing Messages. We apply L message passing steps successively. In each step, we pass messages between all samples in a batch and obtain updated features $\boldsymbol{h}_i^{l+1}$ of node i at message passing step $l+1$ by aggregating the features $\boldsymbol{h}_j^l$ of all neighbouring nodes $j \in N_i$ at message passing step l:

$$\boldsymbol{h}_i^{l+1} = \sum_{j \in N_i} \boldsymbol{W}^l \boldsymbol{h}_j^l \tag{1}$$

where $\boldsymbol{W}^l$ is the corresponding weight matrix of message passing step l. As we construct a fully connected graph, the neighboring nodes N_i consist of all nodes in the given batch, thus each feature representation of an image is affected by all the other images in the mini-batch.

Attention Weights on the Messages. Not all samples of a mini-batch are equally informative to predict the decision boundaries between classes. Hence, we add an attention score α to every message passing step (see Figure 2 on Message Passing) to allow each sample to weigh the importance of the other samples in the mini-batch:

$$\boldsymbol{h}_i^{l+1} = \sum_{j \in N_i} \alpha_{ij}^l \boldsymbol{W}^l \boldsymbol{h}_j^l \tag{2}$$

where α_{ij} is the attention score between node i and node j. We utilize dot-product self-attention to compute the attention scores, leading to α_{ij} at step l defined as:

$$\alpha_{ij}^l = \frac{\boldsymbol{W}_q^l \boldsymbol{h}_i^l (\boldsymbol{W}_k^l \boldsymbol{h}_j^l)^T}{\sqrt{d}} \tag{3}$$

where $\boldsymbol{W}_q^l$ is the weight matrix corresponding to the receiving node and $\boldsymbol{W}_k^l$ is the weight matrix corresponding to the sending node on message passing step l. Furthermore, we apply the softmax function to all in-going attention scores (edges) of a given node i. To allow the MPN to learn a diverse set of attention scores, we apply M dot product self-attention heads in every message passing step and concatenate their results. To this end, instead of using single weight matrices $\boldsymbol{W}_q^l$, $\boldsymbol{W}_k^l$ and $\boldsymbol{W}^l$, we now use different weight matrices $\boldsymbol{W}_q^{l,m} \in \mathcal{R}^{\frac{d}{M} \times d}$, $\boldsymbol{W}_k^{l,m} \in \mathcal{R}^{\frac{d}{M} \times d}$ and $\boldsymbol{W}^{l,m} \in \mathcal{R}^{\frac{d}{M} \times d}$ for each attention head:

$$\boldsymbol{h}_i^{l+1} = cat\left(\sum_{j \in N_i} \alpha_{ij}^{l,1} \boldsymbol{W}^{l,1} \boldsymbol{h}_j^l, ..., \sum_{j \in N_i} \alpha_{ij}^{l,M} \boldsymbol{W}^{l,M} \boldsymbol{h}_j^l\right) \tag{4}$$

where cat represents the concatenation.

Note, by using the attention-head specific weight matrices, we reduce the dimension of all embeddings $\boldsymbol{h}_j^l$ by $\frac{1}{M}$ so that when we concatenate the embeddings generated by all attention heads the resulting embedding $\boldsymbol{h}_i^{l+1}$ has the same dimension as the input embedding $\boldsymbol{h}_i^l$.

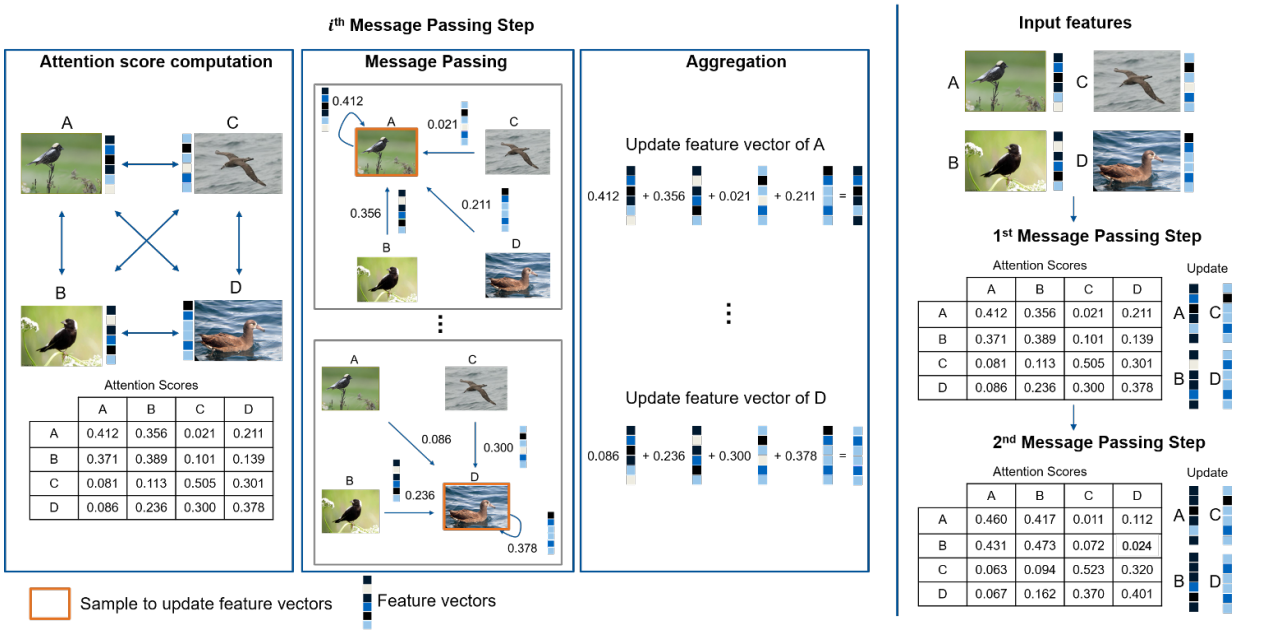

Figure 2. Left: To update the feature vectors in a message-passing step we first construct a fully connected graph and compute attention scores between all samples in a batch. We then pass messages between nodes and weigh them with the corresponding attention scores. During the aggregation step, we sum the weighted messages to get updated node features. Right: Visualization of development of attention scores and feature vectors over two steps of message passing steps showing that feature vectors, as well as attention scores between samples from the same class, get more and more similar.

Adding Skip Connections. We add a skip connection around the attention block (He et al., 2016) and apply layer normalization (Ba et al., 2016) given by:

$$f(\boldsymbol{h}_i^{l+1}) = LayerNorm(\boldsymbol{h}_i^{l+1} + \boldsymbol{h}_i^{l}) \tag{5}$$

where $\boldsymbol{h}_i^{l+1}$ is the outcome of Equation 4. We then apply two fully connected layers, followed by another skip connection (He et al., 2016) and a layer normalization (Ba et al., 2016):

$$g(\boldsymbol{h}_i^{l+1}) = LayerNorm(FF(f(\boldsymbol{h}_i^{l+1})) + f(\boldsymbol{h}_i^{l+1})) \tag{6}$$

where FF represents the two linear layers. Finally, we pass the outcome of Equation 6 to the next message passing step. For illustrative purposes, in Figure 2, we show how the attention scores and the feature vectors evolve over the message passing steps. As can be seen, the feature vectors of samples of the same class become more and more similar. Similar to (Velickovic et al., 2018), we indirectly address oversmoothing by applying node-wise attention scores $\alpha_{i,j}$ (Equation 3) during the feature aggregation step (Min et al., 2020).

3.4. Optimization

We apply a fully connected layer on the refined features after the last message passing step and then use cross-entropy loss. Even if cross-entropy loss itself does not take into account the relations between different samples, this information is already present in the refined embeddings, thanks to the message passing steps. As the MPN takes its initial feature vectors from the backbone CNN, we add an auxiliary cross-entropy loss to the backbone CNN, to ensure a sufficiently discriminative initialization. This loss is also needed since at test time we do not use the MPN, as described below. Both loss functions utilize label smoothing and low temperature scaling (Teh et al., 2020; Zhai & Wu, 2019) to ensure generalized, but discriminative, decision boundaries.

3.5. Inference

One disadvantage of using the MPN during inference is that in order to generate an embedding vector for a sample, we need to create a batch of samples to perform message passing as we do during the training. However, using the MPN during inference would be unfair to other methods that directly perform retrieval on the CNN embedding since we would be adding parameters, hence, expressive power, to the model. Therefore, we perform all experiments by directly using the embedding feature vectors of the backbone CNN unless stated differently. The intuition is that when optimizing the CNN and MPN together in an end-to-end fashion, the CNN features will have also improved with the information of sample relations. In the ablation studies, we show how the performance can be further improved with a simple batch construction strategy at test time. For more discussion on using MPN at test time, we refer the reader to the supplementary material.

4. Experiments

In this section, we compare our proposed approach to state-of-the-art deep metric learning approaches on four public benchmarks. To underline the effectiveness of our approach, we further present an extensive ablation study.

4.1. Implementation Details

We implement our method in PyTorch (Paszke et al., 2017) library. Following other works (Brattoli et al., 2019; Çakir et al., 2019; Manmatha et al., 2017; Sanakoyeu et al., 2019; Teh et al., 2020; Xuan et al., 2020b; Zhai & Wu, 2019), we present results using ResNet50 (He et al., 2016) pretrained on ILSVRC 2012-CLS dataset (Russakovsky et al., 2015) as backbone CNN. Like the majority of recent methods (Ge et al., 2018; Kim et al., 2020; Park et al., 2019; Qian et al., 2019; Wang et al., 2019a;b; Zhu et al., 2020), we use embedding dimension of sizes 512 for all our experiments and low temperature scaling for the softmax cross-entropy loss function (Guo et al., 2017). Furthermore, we preprocess the images following (Kim et al., 2020). We resize the cropped image to 227×227, followed by applying a random horizontal flip. During test time, we resize the images to 256×256 and take a center crop of size 227×227. We train all networks for 70 epochs using RAdam optimizer (Liu et al., 2020). To find all hyperparameters we perform random search (Bergstra & Bengio, 2012). For mini-batch construction, we first randomly sample a given number of classes, followed by randomly sampling a given number of images for each class as commonly done in metric learning (Elezi et al., 2020; Schroff et al., 2015; Teh et al., 2020; Zhai & Wu, 2019). We use small mini-batches of size 50-100 and provide an analysis on different numbers of classes and samples on CUB-200-2011 and Cars196 in the supplementary. Our forward pass takes 73% of time for the backbone and the remaining for the MPN. All the training is done in a single TitanX GPU, *i.e.*, the method is memory efficient.

4.2. Benchmark Datasets and Evaluation Metrics

Datasets: We conduct experiments on 4 publicly available datasets using the conventional splitting protocols (Song et al., 2016):

- CUB-200-2011 (Wah et al., 2011) consists of 200 classes of birds with each class containing 58 images on average. For training, we use the first 100 classes and for testing the remaining classes.
- Cars196 (Krause et al., 2013) contains 196 classes representing different cars with each class containing on average 82 images. We use the first 98 classes for training and the remaining classes for testing.
- Stanford Online Products (SOP) (Song et al., 2016) consists of 22,634 classes (5 images per class on average) of product images from ebay. We use $11,318$ classes for training and the remaining $11,316$ classes for testing.
- In-Shop Clothes (Liu et al., 2016) contains $7,982$ classes of clothing items, with each class having 4 images on average. We use $3,997$ classes for training, while the test set, containing $3,985$ classes, is split into a query set and a gallery set.

Evaluation Metrics: For evaluation, we use the two commonly used evaluation metrics, Recall@K (R@K) (Jégou et al., 2011) and Normalized Mutual Information (NMI) (McDaid et al., 2011). The first one evaluates the retrieval performance by computing the percentage of images whose K nearest neighbors contain at least one sample of the same class as the query image. To evaluate the clustering quality, we apply K-means clustering (MacQueen, 1967) on the embedding feature vectors of all test samples, and compute NMI based on this clustering. To be more specific, NMI evaluates how much the knowledge about the ground truth classes increases given the clustering obtained by the K-means algorithm.

4.3. Comparison to state-of-the-art

Quantitative Results. In Table 1, we present the results of our method and compare them with the results of other approaches on CUB-200-2011 (Wah et al., 2011), Cars196 (Krause et al., 2013), and Stanford Online Products (Song et al., 2016). On CUB-200-2011 dataset, our method reaches 70.3 Recall@1, an improvement of 0.6 percentage points (pp) over the state-of-the-art *Proxy Anchor* (Kim et al., 2020) using ResNet50 backbone. On the NMI metric, we outperform the highest scoring method, *DiVA* (Milbich et al., 2020) by 2.6*pp*. On Cars196, we reach 88.1 Recall@1, an improvement of $0.4pp$ over *Proxy Anchor* (Kim et al., 2020) with ResNet50 backbone. On the same dataset, we reach 74.8 on the NMI score, $0.8pp$ higher than the previous best-performing method, *Normalized Softmax* (Zhai & Wu, 2019). On Stanford Online Products dataset, our method reaches 81.4 Recall@1 which is $1.3pp$ better than the previous best method, *HORDE* (Jacob et al., 2019). On the NMI metric, our method reaches the highest score, outperforming *SoftTriple Loss* (Qian et al., 2019) by $0.6pp$.

Finally, we present the results of our method on the In-Shop Clothes dataset in Table 2. Our method reaches 92.8 Recall@1, an improvement of $0.7pp$ over the previous best method *Proxy Anchor* (Kim et al., 2020) with ResNet50 backbone. In summary, while in the past, different methods (*Proxy Anchor* (Kim et al., 2020), *ProxyNCA++* (Teh et al., 2020), *Normalized Softmax* (Zhai & Wu, 2019), *HORDE* (Jacob et al., 2019), *SoftTriple Loss* (Qian et al., 2019),

		CUB-200-2011					CARS196					Stanford Online Products			
Method	BB	**R@1**	**R@2**	**R@4**	**R@8**	**NMI**	**R@1**	**R@2**	**R@4**	**R@8**	**NMI**	**R@1**	**R@10**	**R@100**	**NMI**
Triplet^{64} (Schroff et al., 2015) *CVPR15*	G	42.5	55	66.4	77.2	55.3	51.5	63.8	73.5	82.4	53.4	66.7	82.4	91.9	89.5
Npairs^{64} (Sohn, 2016) *NeurIPS16*	G	51.9	64.3	74.9	83.2	60.2	68.9	78.9	85.8	90.9	62.7	66.4	82.9	92.1	87.9
Deep Spectral^{512} (Law et al., 2017) *ICML17*	BNI	53.2	66.1	76.7	85.2	59.2	73.1	82.2	89.0	93.0	64.3	67.6	83.7	93.3	89.4
Angular Loss^{512} (Wang et al., 2017) *ICCV17*	G	54.7	66.3	76	83.9	61.1	71.4	81.4	87.5	92.1	63.2	70.9	85.0	93.5	88.6
Proxy-NCA^{64} (Movshovitz-Attias et al., 2017) *ICCV17*	BNI	49.2	61.9	67.9	72.4	59.5	73.2	82.4	86.4	88.7	64.9	73.7	-	-	90.6
Margin Loss^{128} (Manmatha et al., 2017) *ICCV17*	R50	63.6	74.4	83.1	90.0	69.0	79.6	86.5	91.9	95.1	69.1	72.7	86.2	93.8	90.7
Hierarchical triplet^{512} (Ge et al., 2018) *ECCV18*	BNI	57.1	68.8	78.7	86.5	-	81.4	88.0	92.7	95.7	-	74.8	88.3	94.8	-
ABE^{512} (Kim et al., 2018) *ECCV18*	G	60.6	71.5	79.8	87.4	-	85.2	90.5	94.0	96.1	-	76.3	88.4	94.8	-
Normalized Softmax^{512} (Zhai & Wu, 2019) *BMVC19*	R50	61.3	73.9	83.5	90.0	69.7	84.2	90.4	94.4	96.9	**74.0**	78.2	90.6	96.2	91.0
RLL-H^{512} (Wang et al., 2019b) *CVPR19*	BNI	57.4	69.7	79.2	86.9	63.6	74	83.6	90.1	94.1	65.4	76.1	89.1	95.4	89.7
Multi-similarity^{512} (Wang et al., 2019a) *CVPR19*	BNI	65.7	77.0	86.3	91.2	-	84.1	90.4	94.0	96.5	-	78.2	90.5	96.0	-
Relational Knowledge^{512} (Park et al., 2019) *CVPR19*	G	61.4	73.0	81.9	89.0	-	82.3	89.8	94.2	96.6	-	75.1	88.3	95.2	-
Divide and Conquer^{1028} (Sanakoyeu et al., 2019) *CVPR19*	R50	65.9	76.6	84.4	90.6	69.6	84.6	90.7	94.1	96.5	70.3	75.9	88.4	94.9	90.2
SoftTriple Loss^{512} (Qian et al., 2019) *ICCV19*	BNI	65.4	76.4	84.5	90.4	69.3	84.5	90.7	94.5	96.9	70.1	78.3	90.3	95.9	**92.0**
HORDE^{512} (Jacob et al., 2019) *ICCV19*	BNI	66.3	76.7	84.7	90.6	-	83.9	90.3	94.1	96.3	-	**80.1**	**91.3**	96.2	-
MIC^{128} (Brattoli et al., 2019) *ICCV19*	R50	66.1	76.8	85.6	-	69.7	82.6	89.1	93.2	-	68.4	77.2	89.4	95.6	90.0
Easy triplet mining^{512} (Xuan et al., 2020b) *WACV20*	R50	64.9	75.3	83.5	-	-	82.7	89.3	93.0	-	-	78.3	90.7	**96.3**	-
Group Loss^{1024} (Elezi et al., 2020) *ECCV20*	BNI	65.5	77.0	85.0	91.3	69.0	85.6	91.2	94.9	97.0	**72.7**	75.1	87.5	94.2	**90.8**
Proxy NCA++^{512} (Teh et al., 2020) *ECCV20*	R50	66.3	77.8	**87.7**	91.3	**71.3**	84.9	90.6	94.9	97.2	71.5	79.8	**91.4**	**96.4**	-
DiVA^{512} (Milbich et al., 2020) *ECCV20*	R50	**69.2**	**79.3**	-	-	**71.4**	**87.6**	**92.9**	-	-	72.2	79.6	-	-	90.6
PADS^{128} (Roth et al., 2020) *CVPR20*	R50	67.3	78.0	85.9	-	69.9	83.5	89.7	93.8	-	68.8	76.5	89.0	95.4	89.9
Proxy Anchor^{512} (Kim et al., 2020) *CVPR20*	BNI	68.4	79.2	86.8	**91.6**	-	86.1	91.7	95.0	**97.3**	-	79.1	90.8	96.2	-
Proxy Anchor^{512} (Kim et al., 2020) *CVPR20*	R50	**69.7**	**80.0**	**87.0**	**92.4**	-	**87.7**	**92.9**	**95.8**	**97.9**	-	**80.0**	**91.7**	**96.6**	-
Proxy Few^{512} (Zhu et al., 2020) *NeurIPS20*	BNI	66.6	77.6	86.4	-	69.8	85.5	**91.8**	**95.3**	-	72.4	78.0	90.6	96.2	90.2
Ours^{512}	R50	**70.3**	**80.3**	**87.6**	**92.7**	**74.0**	**88.1**	**93.3**	**96.2**	**98.2**	**74.8**	**81.4**	**91.3**	95.9	**92.6**

Table 1. Retrieval and Clustering performance on *CUB-200-2011*, *CARS196* and *Stanford Online Products* datasets. Bold indicates best, red second best, and blue third best results. The exponents attached to the method name indicates the embedding dimension. BB=backbone, G=GoogLeNet, BNI=BN-Inception and R50=ResNet50.

Method	BB	**R@1**	**R@10**	**R@20**	**R@40**
FashionNet^{4096} (Liu et al., 2016) *CVPR16*	V	53.0	73.0	76.0	79.0
A-BIER^{512} (Opitz et al., 2020) *PAMI20*	G	83.1	95.1	96.9	97.8
ABE^{512} (Kim et al., 2018) *ECCV18*	G	87.3	96.7	97.9	98.5
Multi-similarity^{512} (Wang et al., 2019a) *CVPR19*	BNI	89.7	**97.9**	98.5	**99.1**
Learning to Rank^{512} (Çakir et al., 2019)	R50	90.9	97.7	98.5	**98.9**
HORDE^{512} (Jacob et al., 2019) *ICCV19*	BNI	90.4	97.8	98.4	**98.9**
MIC^{128} (Brattoli et al., 2019) *ICCV19*	R50	88.2	97.0	98.0	98.8
Proxy NCA++^{512} (Teh et al., 2020) *ECCV20*	R50	90.4	**98.1**	**98.8**	**99.2**
Proxy Anchor^{512} (Kim et al., 2020) *CVPR20*	BNI	**91.5**	**98.1**	**98.8**	**99.1**
Proxy Anchor^{512} (Kim et al., 2020) *CVPR20*	R50	**92.1**	**98.1**	**98.7**	**99.2**
Ours^{512}	R50	**92.8**	**98.5**	**99.1**	**99.2**

Table 2. Retrieval performance on *In Shop Clothes*.

DiVA (Milbich et al., 2020)) scored the highest in at-least one metric, now our method reaches the best results in all Recall@1 and NMI metrics across all four datasets.

Qualitative Results. In Figure 6, we present qualitative results on the retrieval task for all four datasets. In all cases, the query image is given on the left, with the four nearest neighbors given on the right. Green boxes indicate cases where the retrieved image is of the same class as the query image, and red boxes indicate a different class. In supplementary material, we provide qualitative evaluations on the clustering performance using t-SNE (van der Maaten & Hinton, 2012) visualization.

4.4. Ablation Studies and Robustness Analysis

In this section, we use the CUB-200-2011 (Wah et al., 2011) and Cars196 (Krause et al., 2013) datasets to analyze the robustness of our method and show the importance of our design choices.

MPN Matters. To show the performance improvement when using the MPN during training, we conduct experiments by training the backbone CNN solely with the auxiliary loss, i.e., the cross-entropy loss on the backbone CNN, and without MPN (see the first row in Table 3). For a fair comparison, we use the same implementation details as for the training with MPN. On CUB-200-2011, this leads to a performance drop of $2.8pp$ in Recall@1 (to 67.5) and $4.2pp$ in NMI (to 69.8). On Cars196, it leads to a more significant performance drop of $3.9pp$ in Recall@1 (to 84.2) and $6.1pp$ in NMI (to 68.7), showing the benefit of our proposed formulation.

To give an intuition of how the MPN evolves during the training process, we use GradCam (Selvaraju et al., 2020) to observe which neighbors a sample relies on when computing the final class prediction after the MPN (Selvaraju et al., 2020). To do so, we compare the predictions of an untrained MPN to a trained one. As can be seen in the left part of Figure 7, the untrained MPN takes information from nearly all samples in the batch into account, where red, blue, and green represent different classes. The trained MPN (left part of Figure 7) only relies on the information of samples of the same class. This suggests that using the MPN with self-attention scores as edge weights enforces the embeddings of negative and positive samples to become more dissimilar and similar, respectively. In supplementary, we also provide and compare visualizations of the embedding vectors of a batch of samples after one epoch of training and of all test samples after the whole training.

Number of Message Passing Steps and Attention Heads. In Figure 3, we investigate the robustness of the algorithm when we differ the number of message passing steps and

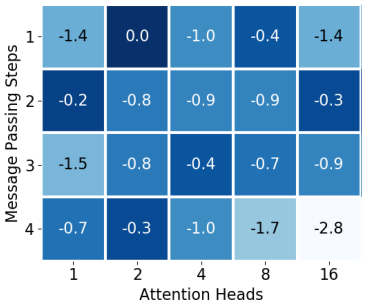

Figure 3. Relative difference to the best model with respect to Recall@1 on CUB-200-2011.

Figure 4. Relative difference to the best model with respect to Recall@1 on Cars196.

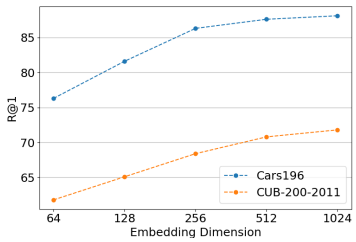

Figure 5. Performance for different embedding dimensions on CUB-200-2011 and Cars196.

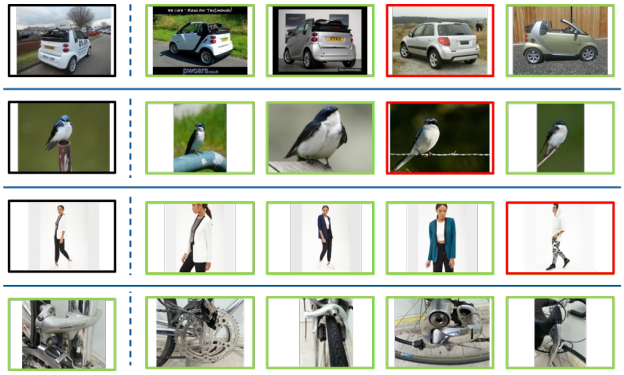

Figure 6. Retrieval results on a set of images from *CUB-200-2011* (top), *Cars196* (second from top), *Stanford Online Products* (second from bottom), and *In-Shop Clothes* (bottom) datasets using our model. The most left column contains query images and the results are ranked by distance. Green frames indicate that the retrieved image is from the same class as the query image, while red frames indicate that the retrieved image is from a different class.

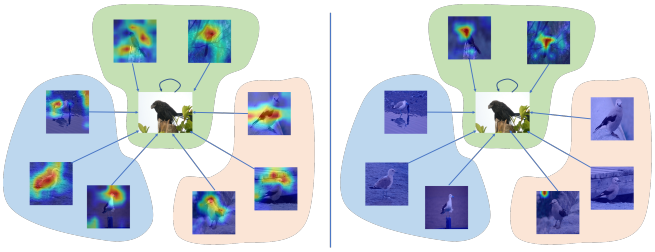

Figure 7. Comparison of the embeddings of a given batch after one epoch of training without and with MPN.

attention heads of our MPN. On CUB-200-2011 dataset, we reach the best results when we use a single message passing step, containing two attention heads. We see that increasing the number of message passing steps or the number of attention heads, for the most part, does not result in a large drop in performance. The biggest drop in performance happens when we use four message-passing steps, each having sixteen attention heads. In Figure 4, we do a similar robustness analysis for the Cars196 dataset. Unlike CUB-200-2011, the method performs best using two layers and eight attention heads. However, it again performs worst using four message passing steps. This observation is in line with (Velickovic et al., 2018), which also utilizes a few message passing steps when applying graph attention.

Embedding Dimension. In Figure 5, we measure the performance of the model as a function of the embedding size. We observe that the performance of the network increases on both datasets when we increase the size of the embedding layer. This is unlike (Wang et al., 2019a), which reports a drop in performance when the size of the embedding layer gets bigger than 512. While increasing the dimension of the embedding layer results in even better performance, for fairness with the other methods that do not use an embedding size larger than 512, we avoid those comparisons.

Auxiliary Loss Function. Considering that in the default scenario, we do not use the MPN during inference, we investigate the effect of adding the auxiliary loss function at the top of the backbone CNN embedding layer. On CUB-200-2011 dataset, we see that such a loss helps the network improve by $2.2pp$ in Recall@1. Without the loss, the performance of the network drops to 68.1 as shown in the second row of Table 3. On the other hand, removing the auxiliary loss function leads to a performance drop of only $0.9pp$ in Recall@1 on Cars196 to 87.2. However, the NMI performance drops by $2.7pp$ to 72.1 on Cars196 and $2.0pp$ on CUB-200-2011.

Implicit Regularization. We further investigate the training behavior of our proposed approach on CUB-200-2011. As already stated above, Group Loss (Elezi et al., 2020) also utilized contextual classification, with the authors claiming that it introduces implicit regularization and thus less overfitting. However, their approach is based on a handcrafted label propagation rule, while ours takes into account the contextual information in an end-to-end learnable way.

Therefore, we present the training behavior of our approach and compare it to the behavior of the Group Loss (Elezi et al., 2020). As can be seen in Figure 8, Group Loss (Elezi et al., 2020) shows higher overfitting on the training data, while our method is capable of better generalization on the test dataset and has a smaller gap between training and test performance. We argue that by taking into account the global structure of the dataset in an end-to-end learnable way, our approach is able to induce an even stronger implicit regularization.

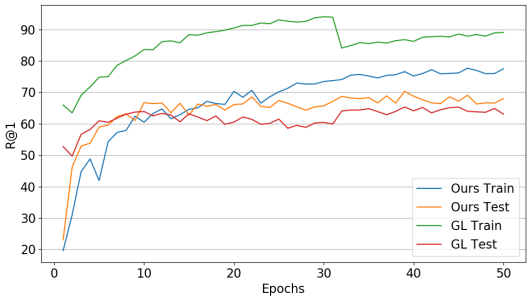

Figure 8. Performance on training and test data of CUB-200-2011 compared to Group Loss (Elezi et al., 2020).

Using MPN During Test Time. In Table 3, we analyze the effect of applying message passing during inference (see row four). On CUB-200-2011 dataset, we improve by $0.5pp$ in Recall@1, and by $0.5pp$ on the NMI metric. On Cars196 dataset, we also gain $0.5pp$ in Recall@1 by using MPN during inference. More impressively, we gain $1.4pp$ in the NMI metric, putting our results $2.2pp$ higher than *Normalized Softmax* (Zhai & Wu, 2019). We gain an improvement in performance in all cases, at the cost of extra parameters.

Note, our method does not require the usage of these extra parameters in inference. As we have shown, for a fair comparison, our method reaches state-of-the-art results even without using MPN during inference (see Tables 1 and 2). We consider the usage of MPN during inference a performance boost, but not a central part of our work.

		CUB-200-2011		CARS196	
Training Losses	Test Time Embeddings	**R@1**	**NMI**	**R@1**	**NMI**
Cross-Entropy	Backbone Embeddings	67.5	69.8	84.2	68.7
MPN Loss	Backbone Embeddings	68.1	72.0	87.2	72.1
MPN Loss + Auxiliary Loss	Backbone Embeddings	70.3	74.0	88.1	74.8
MPN Loss + Auxiliary Loss	MPN Embeddings	**70.8**	**74.5**	**88.6**	**76.2**

Table 3. Performance of the network with and without MPN during training and testing time. We achieved all results using embedding dimension 512.

Ensembles. The Group Loss (Elezi et al., 2020) showed that the performance of their method significantly improves by using an ensemble at test time. The ensemble was built by simply concatenating the features of k independently trained networks. Similarly, we also conduct experiments on ensembles using 2 and 5 networks, respectively, and compare our ensemble with that of (Elezi et al., 2020).

	CUB-200-2011		Cars196		Stanford Online Products		In-Shop Clothes
	R@1	**NMI**	**R@1**	**NMI**	**R@1**	**NMI**	**R@1**
GL	65.5	69.0	85.6	72.7	75.7	91.1	-
Ours	70.3	74.0	88.1	74.8	81.4	92.6	92.8
GL 2	65.8	68.5	86.2	72.6	75.9	91.1	-
Ours 2	72.2	74.3	90.9	74.9	81.8	92.7	92.9
GL 5	66.9	70.0	88.0	74.2	76.3	91.1	-
Ours 5	**73.1**	**74.4**	**91.5**	**75.4**	**82.1**	**92.8**	**93.4**

Table 4. Performance of our ensembles and comparisons with the ensemble models of (Elezi et al., 2020).

In Table 4, we present the results of our ensembles. We see that when we use 2 networks, the performance increases by $1.9pp$ on CUB-200-2011, $3.0pp$ on Cars196, and $0.4pp$ on Stanford Online Products. Similarly, the NMI score also improves by $0.3pp$ on CUB-200-2011, $0.1pp$ on Cars196, and $0.1pp$ on Stanford Online Products. Unfortunately, the Recall@1 performance on In-Shop Clothes only improves by $0.1pp$. Using 5 networks, the performance increases by $2.8pp$ on CUB-200-2011, $3.4pp$ on Cars196, $0.7pp$ on Stanford Online Products, and $0.6pp$ on In-Shop Clothes compared to using a single network. NMI on CUB-200-2011 is improved by $0.4pp$ compared to a single network, on Cars196 it increases by $0.6pp$ more and on Stanford Online Products it increases by $0.6pp$.

Compared to (Elezi et al., 2020), the performance increase of our approach from one network to an ensemble is higher. This is surprising, considering that our network starts from a higher point, and has less room for improvement.

5. Conclusions

In this work, we propose a model that utilizes the power of message passing networks for the task of deep metric learning. Unlike classical metric learning methods, e.g., triplet loss, our model utilizes all the intra-batch relations in the mini-batch to promote similar embeddings for images coming from the same class, and dissimilar embeddings for samples coming from different classes. Our model is fully learnable, end-to-end trainable, and does not utilize any handcrafted rules. Furthermore, our model achieves state-of-the-art results while using the same number of parameters, and compute time, during inference. In future work, we will explore the applicability of our model for the tasks of semi-supervised deep metric learning and deep metric learning in the presence of only relative labels.

Acknowledgements. This research was partially funded by the Humboldt Foundation through the Sofia Kovalevskaja Award. We thank Guillem Brasó for useful discussions.

References

Alemu, L. T., Shah, M., and Pelillo, M. Deep constrained dominant sets for person re-identification. In *International Conference on Computer Vision (ICCV)*, 2019.

Ba, L. J., Kiros, J. R., and Hinton, G. E. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V. F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, H. F., Ballard, A. J., Gilmer, J., Dahl, G. E., Vaswani, A., Allen, K. R., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research (JMLR)*, 13:281–305, 2012.

Brasó, G. and Leal-Taixé, L. Learning a neural solver for multiple object tracking. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Brattoli, B., Roth, K., and Ommer, B. MIC: mining interclass characteristics for improved metric learning. In *International Conference on Computer Vision (ICCV)*, 2019.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems (NIPS)*, 1994.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Çakir, F., He, K., Xia, X., Kulis, B., and Sclaroff, S. Deep metric learning to rank. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020.

Chopra, S., Hadsell, R., and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

Duan, Y., Zheng, W., Lin, X., Lu, J., and Zhou, J. Deep adversarial metric learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Duan, Y., Chen, L., Lu, J., and Zhou, J. Deep embedding learning with discriminative sampling policy. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Elezi, I., Torcinovich, A., Vascon, S., and Pelillo, M. Transductive label augmentation for improved deep network learning. In *International Conference on Pattern Recognition (ICPR)*, 2018.

Elezi, I., Vascon, S., Torchinovich, A., Pelillo, M., and Leal-Taixé, L. The group loss for deep metric learning. In *European Conference in Computer Vision (ECCV)*, 2020.

Ge, W., Huang, W., Dong, D., and Scott, M. R. Deep metric learning with hierarchical triplet loss. In *European Conference in Computer Vision (ECCV)*, 2018.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, 2017.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Guo, M., Chou, E., Huang, D., Song, S., Yeung, S., and Fei-Fei, L. Neural graph matching networks for fewshot 3d action recognition. In *European Conference on Computer Vision (ECCV)*, 2018.

He, K., Zhang, X., Ren, S., and ian Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

He, K., Çakir, F., Bargal, S. A., and Sclaroff, S. Hashing as tie-aware learning to rank. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018a.

He, K., Lu, Y., and Sclaroff, S. Local descriptors optimized for average precision. In *Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2018b.

Jacob, P., Picard, D., Histace, A., and Klein, E. Metric learning with HORDE: high-order regularizer for deep embeddings. In *International Conference on Computer Vision (ICCV)*, 2019.

Jégou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell. (tPAMI)*, 33(1):117–128, 2011.

Kim, S., Kim, D., Cho, M., and Kwak, S. Proxy anchor loss for deep metric learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Kim, W., Goyal, B., Chawla, K., Lee, J., and Kwon, K. Attention-based ensemble for deep metric learning. In *European Conference on Computer Vision (ECCV)*, 2018.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *International IEEE Workshop on 3D Representation and Recognition*, 2013.

Law, M. T., Urtasun, R., and Zemel, R. S. Deep spectral clustering learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Lin, X., Duan, Y., Dong, Q., Lu, J., and Zhou, J. Deep variational metric learning. In *European Conference in Computer Vision (ECCV)*, 2018.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations (ICLR)*, 2020.

Liu, Z., Luo, P., Qiu, S., Wang, X., and Tang, X. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2016.

MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1*, pp. 281–297, 1967.

Manmatha, R., Wu, C., Smola, A. J., and Krähenbühl, P. Sampling matters in deep embedding learning. In *International Conference on Computer Vision (ICCV)*, 2017.

McDaid, A. F., Greene, D., and Hurley, N. J. Normalized mutual information to evaluate overlapping community finding algorithms. *CoRR*, abs/1110.2515, 2011.

Milbich, T., Roth, K., Bharadhwaj, H., Sinha, S., Bengio, Y., Ommer, B., and Cohen, J. P. Diva: Diverse visual feature aggregation for deep metric learning. In *European Conference in Computer Vision (ECCV)*, 2020.

Min, Y., Wenkel, F., and Wolf, G. Scattering GCN: overcoming oversmoothness in graph convolutional networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Movshovitz-Attias, Y., Toshev, A., Leung, T. K., Ioffe, S., and Singh, S. No fuss distance metric learning using proxies. In *International Conference on Computer Vision (ICCV)*, 2017.

Narasimhan, M., Lazebnik, S., and Schwing, A. G. Out of the box: Reasoning with graph convolution nets for factual visual question answering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Opitz, M., Waltner, G., Possegger, H., and Bischof, H. BIER - boosting independent embeddings robustly. In *International Conference on Computer Vision (ICCV)*, 2017.

Opitz, M., Waltner, G., Possegger, H., and Bischof, H. Deep metric learning with BIER: boosting independent embeddings robustly. *IEEE Trans. Pattern Anal. Mach. Intell. (tPAMI)*, 42(2):276–290, 2020.

Park, W., Kim, D., Lu, Y., and Cho, M. Relational knowledge distillation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. *NIPS Workshops*, 2017.

Qian, Q., Shang, L., Sun, B., Hu, J., Tacoma, T., Li, H., and Jin, R. Softtriple loss: Deep metric learning without triplet sampling. In *International Conference on Computer Vision (ICCV)*, 2019.

Revaud, J., Almazán, J., Rezende, R. S., and de Souza, C. R. Learning with average precision: Training image retrieval with a listwise loss. In *International Conference on Computer Vision (ICCV)*, 2019.

Roth, K., Milbich, T., and Ommer, B. PADS: policy-adapted sampling for visual similarity learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis. (IJCV)*, 115 (3):211–252, 2015.

Sanakoyeu, A., Tschernezki, V., Büchler, U., and Ommer, B. Divide and conquer the embedding space for metric learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Schultz, M. and Joachims, T. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vis. (IJCV)*, 128(2):336–359, 2020.

Sohn, K. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Song, H. O., Xiang, Y., Jegelka, S., and Savarese, S. Deep metric learning via lifted structured feature embedding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Teh, E. W., DeVries, T., and Taylor, G. W. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *European Conference on Computer Vision (ECCV)*, 2020.

van der Maaten, L. and Hinton, G. E. Visualizing non-metric similarities in multiple maps. *Machine Learning*, 87(1): 33–55, 2012.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Wang, J., Zhou, F., Wen, S., Liu, X., and Lin, Y. Deep metric learning with angular loss. In *International Conference on Computer Vision (ICCV)*, 2017.

Wang, X., Han, X., Huang, W., Dong, D., and Scott, M. R. Multi-similarity loss with general pair weighting for deep metric learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019a.

Wang, X., Hua, Y., Kodirov, E., Hu, G., Garnier, R., and Robertson, N. M. Ranked list loss for deep metric learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019b.

Weinberger, K. Q. and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research (JMLR)*, 10:207–244, 2009.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, 2015.

Xu, X., Yang, Y., Deng, C., and Zheng, F. Deep asymmetric metric learning via rich relationship mining. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Xuan, H., Souvenir, R., and Pless, R. Deep randomized ensembles for metric learning. In *European Conference Computer Vision (ECCV)*, 2018.

Xuan, H., Stylianou, A., Liu, X., and Pless, R. Hard negative examples are hard, but useful. In *European Conference in Computer Vision (ECCV)*, 2020a.

Xuan, H., Stylianou, A., and Pless, R. Improved embeddings with easy positive triplet mining. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020b.

Yuan, Y., Yang, K., and Zhang, C. Hard-aware deeply cascaded embedding. In *International Conference on Computer Vision (ICCV)*, 2017.

Zhai, A. and Wu, H. Classification is a strong baseline for deep metric learning. In *British Machine Vision Conference (BMVC)*, 2019.

Zhang, X., Zhou, F., Lin, Y., and Zhang, S. Embedding label structures for fine-grained feature representation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Zhao, K., Xu, J., and Cheng, M. Regularface: Deep face recognition via exclusive regularization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Zheng, X., Ji, R., Sun, X., Zhang, B., Wu, Y., and Huang, F. Towards optimal fine grained retrieval via decorrelated centralized loss with normalize-scale layer. In *Conference on Artificial Intelligence (AAAI)*, 2019.

Zhong, Z., Zheng, L., Cao, D., and Li, S. Re-ranking person re-identification with k-reciprocal encoding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Zhu, Y., Yang, M., Deng, C., and Liu, W. Fewer is more: A deep graph metric learning perspective using fewer proxies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.