Spatial Assembly Networks for Image Representation Learning

Yang Li[1] Shichao Kan[2] Jianhe Yuan[1] Wenming Cao[3] Zhihai He[1]*
[1]University of Missouri, MO, USA [2]Beijing Jiaotong University, Beijing, China
[3]Shenzhen University, Shenzhen, China
yltb5@mail.missouri.edu 16112062@bjtu.edu.cn
{yuanjia, hezhi}@missouri.edu wmcao@szu.edu.cn

Abstract

It has been long recognized that deep neural networks are sensitive to changes in spatial configurations or scene structures. Image augmentations, such as random translation, cropping, and resizing, can be used to improve the robustness of deep neural networks under spatial transforms. However, changes in object part configurations, spatial layout of object, and scene structures of the images may still result in major changes in the their feature representations generated by the network, creating significant challenges for various visual learning tasks, including representation or metric learning, image classification and retrieval. In this work, we introduce a new learnable module, called spatial assembly network (SAN), to address this important issue. This SAN module examines the input image and performs a learned re-organization and assembly of feature points from different spatial locations conditioned by feature maps from previous network layers so as to maximize the discriminative power of the final feature representation. This differentiable module can be flexibly incorporated into existing network architectures, improving their capabilities in handling spatial variations and structural changes of the image scene. We demonstrate that the proposed SAN module is able to significantly improve the performance of various metric / representation learning, image retrieval and classification tasks, in both supervised and unsupervised learning scenarios.

1. Introduction

A key challenge in computer vision and machine learning is to construct or learn discriminative representations for the semantic content of images, which should be invariant to changes in camera positions, perspective transforms, object scales, poses, part deformations, spatial displacement, and scene configurations [8, 25]. Recently, deep neural networks have emerged as a powerful approach for visual learning and representation. With its shared weights for convolution across different spatial locations, average or maximum pooling, coupled with sufficient training image augmentations, they are able to generate relatively invariant features or decisions under small spatial variations or transforms. However, researchers have recognized that deep neural networks are still vulnerable to relatively large geometric transformations and spatial variations [23]. This limitation originates from the fixed geometric structures of deep neural modules. For example, the convolution processes the input image or feature images on a fixed grid structure with a small reception field. The pooling layers then process the outputs from the convolution layers with a fixed spatial mapping or channel structures. There is lack of internal mechanisms to handle the flexible spatial variations, including spatial transforms and changes in object poses, spatial layout, and scene structures [22]. In their recent study [17], Kayhan and Gemert even found out that deep neural networks are exploiting the absolution spatial locations and image boundary conditions for object recognition and image classification, challenging the common assumption that convolution layers in modern CNNs are translation invariant.

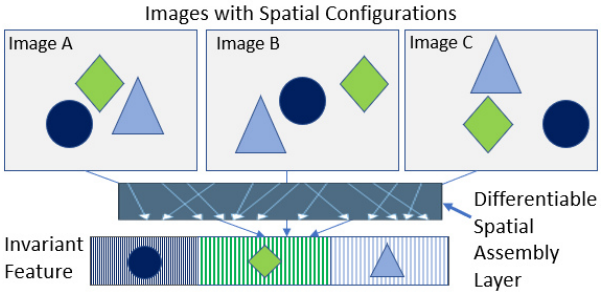

Figure 1. Illustration of invariant image representation learning under generic spatial variations.

Learning invariant features and visual representation with deep neural networks has become an important yet

*Corresponding author: Zhihai He, e-mail: hezhi@missouri.edu.

challenging research problem. Recent research has been focusing on developing various methods on transform-aware data augmentations [4, 11], geometry adversarial training [16], and transform-invariant network modules and structures [14, 33, 35] to improve the robustness of deep neural networks under spatial transforms of images and objects, such as affine or perspective transforms. In this work, we aim to address the challenging problem of invariant feature representation learning under more generic spatial variations, include changes in object poses, part configurations, and scene structures. For example, Figure 1 shows three example images with different spatial configurations of objects due to object motion. Semantically, they should be the same or belong to the same class. However, existing deep neural networks will generate different features for them. Our goal is to design a new spatial assembly network (SAN), which is able to examine the input image and perform a learned re-organization or optimized assembly of feature points from different spatial locations so as to generate invariant features for these three images. This learned spatial assembly is conditioned by feature maps of previous network layers. This differentiable module can be flexibly incorporated into existing network architectures, improving their capabilities in handling spatial variations and structural changes of the image scene, and maximizing the discriminative power of the final feature representation. We will demonstrate that the proposed SAN module is able to improve the performance of various metric / representation learning, image retrieval and classification tasks, in both supervised and unsupervised learning settings.

2. Related Work and Unique Contributions

Recently, a set of methods have been developed in the literature to improve the robustness of deep neural networks under spatial transforms of images and objects. Analytically, [3] has studied the equivalence and invariance of DNN representations to input image transformations. Lenc and and Vedaldi [23] investigated the linear relationships between representations of the original and transformed images. In [22], the training dataset is augmented with different spatial transformations and used to train different networks. The weights are the shared between networks and their generated features are then fused together using maximum pooling. To increase the robustness of deep neural network under spatial transforms, a random transformation module is developed in [33] to transform the feature maps obtained from the neural network and suppress its sensitivity to spatial transforms in the input image. The spatial transformer network has been developed in [14] which is able to locate and predict the spatial transforms of objects in the scene based on previous feature maps and realign the feature maps of objects based on these transforms. This new spatial transformer layer can be inserted into existing network and used to improve the robustness of deep neural network under spatial transforms. To handle object-level spatial variations, an end-to-end network architecture that perform joint detection, orientation estimation, and feature description has been explored in [49]. To achieve adaptive part localization for objects with different shapes, deformable convolution and pooling are developed in [5], which adds 2D offsets to the grid sampling locations and bin positions in the standard convolution and RoI (region of Interest) pooling. Gens and Domingos [9] proposed a generalization of CNN that forms feature maps over arbitrary symmetry groups based on the theory of symmetry groups in [9], resulting in feature maps that were more invariant to symmetry groups. Sohn and Lee [36] proposed a transform-invariant restricted Boltzmann machine (RBM) which is able to generate compact and invariant representation of the input image using probabilistic max pooling. This framework can also be extended to unsupervised learning. To handle the orientation changes, Wang *et al.* [44] proposes to transform the weighted region features into the final orientation invariant feature vector by clustering key points into four orientation-based region proposals. The feature vectors from these four orientation regions are then fused by the aggregation module that outputs an orientation-invariant feature vector. A Laplacian pyramid network structure has been developed in [7] to produce a set of feature maps with different scales which then fused together to improve the robustness of image features under scale changes.

This work is also related to spatial permutation. Permutation optimization is a long standing problem arising in operations research, graph matching, and other applications [1]. It is also referred to as the linear and quadratic assignment problem [39]. Within the context of deep neural networks, channel shuffling has been explored in ShuffleNet [50] to improve the network performance while minimizing its computational complexity. In [25], Lyu *et al.* have developed a deep neural network approach to learn the permutation for channel shuffling. They introduced Lipschitz continuous non-convex penalty so that it can be incorporated into the stochastic gradient descent to approximate permutation. Exact permutations are then obtained by simple rounding at the end.

Compared to existing methods in the literature, our work has the following **unique novelties and contributions**. (1) Existing methods mainly focus on transform-invariant networks and image feature learning. The proposed spatial assembly goes beyond spatial image transforms. It learns to re-organize or re-assemble the feature maps across different spatial locations with the potential to handle generic spatial variations, including changes in poses, part configurations, relative motion between objects, and scene structures. (2) This work represents one of the first efforts to explore spatial re-organization of feature maps for 2D images. The

proposed spatial assembly represents a more generic feature operation than simple permutation. This differentiable module can be directly incorporated into existing deep neural network for end-to-end training to increase network robustness under spatial variations and improve the discriminative power of image features.

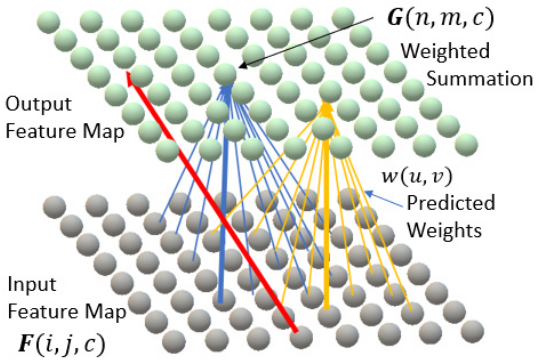

Figure 2. Spatial assembly of feature vectors across different spatial locations to construct the output feature map.

3. Method

In this section, we describe the formulation of spatial assembly and explain its backward error propagation and gradient-based learning process.

3.1. Differentiable Spatial Assembly

It is a differentiable network module which learns the spatial assembly weights from previous feature maps and perform spatial assembly of the feature maps across different spatial locations based on these assembly weights within a single forward pass. The spatial assembly weights are conditioned by the feature maps of the specific input image. In other words, the spatial assembly will be different for different input images. As illustrated in Figure 2, let $F(i, j, c)$ be the feature map at network layer k, spatial location (i, j) and channel c. It serves as the input to the spatial assembly module. Let $G(n, m, c)$ be the output feature map after spatial assembly. The input and output feature maps share the same dimension (W_k, H_k). The output feature map is constructed using the following 2D spatial permutation operation

$$G(n, m, c) = F(i', j', c),\ (i', j') = \mathcal{P}(n, m), \quad (1)$$

where $(i', j') = \mathcal{P}(n, m)$ is a 2D spatial permutation. The 2D spatial permutation can be converted into a 1-D spatial permutation by introducing the location index $u = i \times W_k + j$ and $v = n \times W_k + m$. We have $0 \leq u, v \leq N$ where $N = W_k \times H_k$. With this, (1) can be re-written as

$$G(v, c) = F(u, c), \quad u = \mathcal{P}(v). \quad (2)$$

Let $\boldsymbol{P} = [w(u, v)]_{N \times N}$ be the permutation matrix, which is a binary square matrix with ones at matrix locations $(\mathcal{P}(v), v)$. It should be noted that the permutation matrix $\boldsymbol{P}$ is discrete, which has exactly a single one in every row and each column, and zeros everywhere else. These matrices form discrete points in the Euclidean space, which makes them not differentiable.

To make this spatial permutation module differentiable, we extend this spatial permutation into spatial assembly by relaxing the binary indicator $w(u, v)$ into a continuous weight between [0, 1] which satisfies the following condition

$$\sum_{u=1}^{N} w(u, v) = 1,\ \sum_{v=1}^{N} w(u, v) = 1,\ w(u, v) > 0. \quad (3)$$

In spatial assembly, the output feature map is constructed by the following weighted summation

$$G(v, c) = \sum_{u=0}^{N-1} w(u, v) \cdot F(u, c), \quad (4)$$

as illustrated in Figure 2. Here, every feature vector in the output feature map $G(v, c)$ is computed using the weighted summation of the input feature vectors $F(u, c)$ at all spatial locations.

In order to achieve spatial re-organization of the feature map while maintaining the differentiable property of the spatial assembly weight function $w(u, v)$, we introduce the following two constraints. The first one is the ***minimum entropy constraint*** which aims to ensure locality of the weighting function. From a spatial transform perspective, this will ensure that one object is being moved from one location in the input feature map to another location in the output feature map. Specifically, we define the following entropy function which is the summation of entropies for all rows and all columns of the spatial assembly weight matrix:

$$\begin{aligned} \mathbb{E}[w(u, v)] = &\sum_{v} \sum_{u} \bar{w}_c(u, v) \cdot \log_2 \frac{1}{\bar{w}_c(u, v)}, \\ &+ \sum_{u} \sum_{v} \bar{w}_r(u, v) \cdot \log_2 \frac{1}{\bar{w}_r(u, v)}. \end{aligned} \quad (5)$$

Note that when the entropy is 0, each row or each column will have a single unit value with the rest entries to be 0. During training, this minimum entropy constraint will be used as a part of the loss function to increase the locality of the spatial assembly operation.

The second one is the ***minimum correlation constraint***: during spatial assembly, different input feature vectors are contributing to different output feature vectors. Otherwise, if one input feature vector is contributing significantly to multiple output features, it will result in significant output

information redundancy, or equivalently input information loss. From the spatial transform perspective, this constraint will ensure that different objects are being re-organized to different locations. To this end, we introduce the minimum correlation constraint which aims to minimize the following correlation within the spatial assembly weight map:

$$\begin{aligned}\mathbb{C}[w(u,v)] = &\sum_{u_1 \neq u_2} \sum_{v} w(u_1, v) \cdot w(u_2, v) \\ &+ \sum_{v_1 \neq v_2} \sum_{u} w(u, v_1) \cdot w(u, v_2).\end{aligned} \tag{6}$$

3.2. Spatial Assembly with Local Coherence

The above formulation of spatial assembly aims to achieve spatial re-assembly of the feature map in a differentiable manner. It should be noted that this spatial re-assembly operation is performed on feature vectors at individual spatial locations of the feature map, or individual feature points. Although the spatial assembly is learned by optimizing the target loss function, it is highly likely that feature points from the same object may be dis-assembled into different locations in the output feature map. To address this issue, we propose introduce local coherence into the spatial assembly operation. While it could be more effective to develop a separate network to predict if two feature points belong to the same object or not, we choose to adopt a simple yet efficient measure to enforce the local coherence. Specifically, we define the local coherence $\alpha(i, j; i', j')$ as the correlation (cosine similarity) between feature vector $\boldsymbol{F}_{i,j} = [F(i, j, 1), \cdots, F(i, j, C)]$ and its neighbor $\boldsymbol{F}_{i',j'} = [F(i', j', 1), \cdots, F(i', j', C)]$, i.e.,

$$\alpha(i, j; i', j') = \begin{cases} \frac{\boldsymbol{F}_{i,j} \cdot \boldsymbol{F}_{i',j'}}{||\boldsymbol{F}_{i,j}|| \cdot ||\boldsymbol{F}_{i',j'}||}, & (i', j') \in \Omega_{i,j}, \\ 0, & \text{elsewhere.} \end{cases} \tag{7}$$

where $\Omega_{i,j}$ is the set of 8 direct neighbor points of (i, j). To address the computational complexity, the number of feature vectors can be limited. During coherent spatial assembly, we expect that neighboring feature points with high local coherence should be maintained together in the output feature map. In otherwise words, they should have similar spatial assemble weights. Motivated by this, we introduce the following loss function

$$\mathcal{L}_{LC} = \sum_{(i',j') \in \Omega_{i,j}} \alpha(i, j; i', j') \cdot ||\boldsymbol{W}_{i,j} - \mathcal{S}^{i,j}_{i',j'}[\boldsymbol{W}_{i',j'}]||_2, \tag{8}$$

where $\boldsymbol{W}_{i,j}$ represents the 2-D spatial assembly weight map of size $N_H \times N_W$ for point (i, j). $\mathcal{S}^{i',j'}_{i,j}[\cdot]$ performs a 2-D shift of the whole weight map by one point such that point (i', j') is aligned to point (i, j).

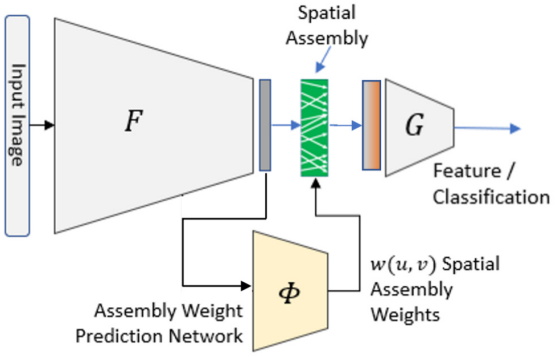

Figure 3. The spatial assembly networks being embedded into the deep neural network.

3.3. Spatial Assembly Networks

Figure 3 shows the design of the spatial assembly network and how it is embedded into existing deep neural networks for feature learning and image classification. The spatial assembly module is integrated into an intermediate layer of the network. The feature map $F(i, j, c)$ generated by network $\mathbb{F}$ is used as input to the spatial assembly network $\mathbf{\Phi}$, which predicts the spatial assembly weight map $w(u, v)$, as defined in the above section. Using this weight map, the input feature map $F(i, j, c)$ is re-assembled into a new feature map $G(n, m, c)$, which will be further processed by the upper network $\mathbb{G}$. In the following, we use the supervised metric learning as an example to explain the loss function design and learning process. This learning processing can be naturally extended to unsupervised feature learning, and image classification and will be evaluated in our experiments.

In supervised metric or feature learning, the network aims to generate discriminative features such that intra-class image feature distance is minimized and the inter-class feature distance is maximized. As illustrated in Figure 1, the proposed spatial assembly has the capability to handle spatial variations caused by changes in object poses, part configurations, spatial layout, and scene structures, and significantly reduce the intra-class feature variations. This can be driven by the metric loss defined at the network output. For example, in our experiments, our baseline system includes the multi-similarity (MS) loss [42]. The MS method computes the similarity scores between image samples in the current mini-batch. The similarity matrix between features of the current mini-batch $\mathbf{S}$. For each sample I_k, we determine the set of positive pairs $\mathcal{P}_k$ and the set of hard negative pairs $\mathcal{N}_k$ based on their similarity scores. $\mathbf{S}_{kp}$ and $\mathbf{S}_{kp}$ are similarity scores of the positive and negative pairs. We define the loss for all samples $\{I_k\}$ in the mini-batch as

follows

$$\mathcal{L}_{FEN} = \frac{1}{N_B}\sum_{k=1}^{N_B}\{\frac{1}{\lambda_P}\log[1+\sum_{p\in\mathcal{P}_k}(e^{-\lambda_P(\mathbf{S}_{kp}-\delta)})] + \frac{1}{\lambda_N}\log[1+\sum_{q\in\mathcal{N}_k}(e^{\lambda_N(\mathbf{S}_{kq}-\delta)})]\}, \quad (9)$$

where δ is a margin threshold, λ_P and λ_N are hyper-parameters for positive and negative pairs. We follow [42] for the setting of these hyper-parameters.

During training, the error gradients from metric loss will back propagated through network $\mathbb{G}$ to the spatial assembly layer, which will be further propagated to the spatial assembly network and the bottom network $\mathbb{F}$. According to (4), the gradients of the output feature map $G(n,m,c)$ with respect to the input feature map $F(i,j,c)$ and the spatial assembly weights are given by

$$\frac{\partial G(n,m,c)}{\partial F(i,j,c)} = w(i\times N+j,\ n\times N+m), \quad (10)$$

and

$$\frac{\partial G(n,m,c)}{\partial w(u,v)} = F(i,j,c), \quad u = i\times N+j, \quad v = n\times N+j. \quad (11)$$

In addition to the error gradients back propagated from the network output, we also use the minimum entropy, minimum correlation constraints, and the local coherence penalty to regulate the training of the spatial assembly weight prediction network $\mathbf{\Phi}$ through the following combined loss

$$\mathcal{L}_{\mathbf{\Phi}} = \lambda_1\cdot\mathbb{E}[w(u,v)] + \lambda_2\cdot\mathbb{C}[w(u,v)] + \lambda_3\cdot\mathcal{L}_{LC}, \quad (12)$$

as illustrated in Figure 3. λ_i are the weighting parameters. Once successfully trained, the SAN module will analyze the incoming feature map, predict the spatial assembly weight map. Figure 4 shows two examples of predicted spatial assembly weight map. In each example, we show the maximum weight of the first two rows from the weight map $w(u,v)$. It should be noted that we have re-organized 1-D weights of each row into an 2-D vector. Each 2-D vector represents the assembly weight vector for one output feature point. We only mark the maximum weight point in each 2-D vector by red for a better visualization. The whole weight map is used to re-assembly the feature map to generate the output feature map, which will be further analyzed by the network to produce the feature or decision.

4. Experimental Results

In this section, we conduct experiments on three different settings: (1) supervised metric learning, (2) unsupervised metric learning, and (3) image classification to evaluate the performance of the spatial assembly network.

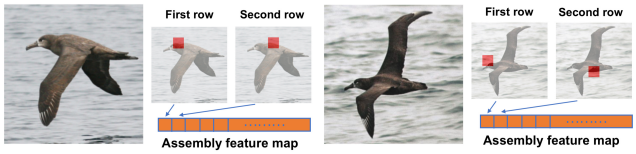

Figure 4. Examples of the first two rows in the predicted spatial assembly weight map.

4.1. Datasets

For supervised and unsupervised deep metric learning, we use the following four benchmark datasets, following the same procedure used by existing papers [30, 43, 46, 47]. **(1) CUB-200-2011 (CUB)** [40] is a fine-grained bird dataset. It contains 11,788 images of birds from 200 categories. The first 100 classes are used for training, the remaining 100 classes are used for testing. **(2) Cars-196 (Cars)** [20] consists of 16,185 car model images (196 classes). We split the first 98 classes (8,054 images) for training, and remaining 98 classes (8,131 images) for testing. **(3) Stanford Online Products (SOP)** [28] consists of 120,053 online product images (22,634 classes) from Ebay. The first 11,318 classes are used for training and the remaining 11,316 classes are used for testing. **(4) In-Shop Clothes Retrieval (In-Shop)** [24] contains 54,642 images with 11,735 clothing classes. We use the predefined 25,882 training images of 3,997 classes for training. The testing set includes 14,218 query images of 3,985 classes and 12,612 gallery images of 3,985 classes.

To evaluate the performance of our method on classification in the supervised setting, we use the following two datasets: **(5) CIFAR-10** [21] consists of 32×32 pixel RGB images in 10 classes. It includes 50,000 training images and 10,000 testing images. **(6) CIFAR-100** [21] is an extension of CIFAR-10. It contains 50,000 training images (100 classes) and 10,000 testing images.

4.2. Supervised Metric Learning

We follow the recent state-of-the-art methods [42, 51, 30, 43] and conduct the experiments on the fine-grained CUB , Cars, SOP, and In-Shop datasets which are challenging for learning discriminative features. We utilize the GoogleNet network [37] pre-trained on ImageNet [32] as the backbone network with an one-layer embedding head to embed feature representation to the 512-dimensional feature space on all datasets for the benchmark performance comparison. We implement our algorithm with PyTorch. The Adam optimizer [19] is used in all experiments with $5e^{-4}$ weight decay. In the following experiments, we use the standard image retrieval performance metric (Recall@K), for performance evaluations and comparisons. Note that the major challenge here is that the training classes are totally differ-

ent from the test classes.

The performance comparisons with existing state-of-the-art supervised metric learning methods on the CUB, Cars, and In-Shop datasets are summarized in Table 1. These methods include: LiftedStruct Loss [28], Histogram Loss [38], N-Pair Loss [35], Clustering [27], BIER (boosting independent embeddings robustly) [29], Angular Loss [41], MS (Multi-Similarity) Loss [42], HDML (hardness-aware deep metric learning) [51], ABIER [30] and XBM (Cross-Batch Memory) [43]. We use the multi-similarity loss [42] with momentum memory bank [10] as the baseline system. The momentum memory bank has a contrastive-based loss [15]. Our proposed method is the baseline system with SAN module. From Table 1, we can see that our method outperforms the state-of-the-art methods by up to 2.6% on the Recall@1, 2, 4, and 8 rates on the CUB dataset. We evaluate the performance of the In-Shop dataset in two settings. One setting uses the whole testing set as the query set and gallery set, the other setting splits the testing set into query set (14,218 query images) and gallery set (12,612 gallery images). The performance of the second setting shows in the brackets.

4.3. Unsupervised Deep Metric Learning

In the following experiments, we evaluate the performance of the spatial assembly network for unsupervised metric learning where image labels are not available. We compare the performance of our proposed methods with the state-of-the-art unsupervised methods: MOM (mining on manifolds) [13], AND (anchor neighborhood discovery) [12], CBSwR (center-based softmax with reconstruction) [26], PSLR (probabilistic structural latent representation) [46], ISIF [48], and aISIF [47] (augmentation invariant and spreading instance feature). For fair comparisons, the authors of the aISIF paper [47] have implemented three other state-of-the-art methods developed for feature learning and adapted them to unsupervised deep metric learning tasks: Examplar [6], NCE (Noise-Contrastive Estimation) [45], and DeepCluster [2], which are included into our comparison. We use the same baseline system in the supervised metric learning.

We use the ImageNet [32] pre-trained GoogleNet [37] as the backbone network and set the embedding feature dimension to 128 on CUB, Cars, and SOP datasets for performance comparison. We use the k-means clustering to cluster the embedding features of training samples and assign pseudo labels to them. We set the cluster number K to be 100 for the CUB and Cars datasets, and set K to 10,000 for the SOP dataset. From Table 2, we can see that our proposed method outperforms the state-of-the-art unsupervised methods by a large margin.

Following the recent state-of-the-art PSLR [46], ISIF [48], and aISIF [47] methods, we also evaluate the performance of our proposed method on Resnet-18 without pre-trained parameters. In this experiment, we use the randomly initialized Resnet-18 network with an one-layer embedding head to verify the effectiveness of our proposed SAN module. We set the feature embedding dimension to 128 and conduct experiments on the large-scale SOP dataset. The results in Table 3 show that our proposed method has improved the Recall@1, Recall@10, and Recall@100 rates by 4.0%, 4.2%, and 4.5%, respectively.

4.4. Image Classification

To verify the generalization capability of our spatial assembly network module in different tasks, we conduct experiments on the image classification task. Specifically, in this experiment, we train the VGG-19 network [34] with the cross-entropy loss to perform the image classification task on the CIFAR-10 and CIFAR-100 datasets [31]. Table 4 shows the classification performance comparison on the CIFAR-10 and CIFAR-100 dataset. We can see that the SAN module is able to improve the classification accuracy by 0.68% and 0.64% on the CIFAR-10 and CIFAR-100, respectively. We notice that this improvement is small since the space for performance improvement on these two well-studied datasets is already very limited. In addition, when applied to image classification, we used the SAN module directly without any modification for this classification task. We observe that, unlike the metric loss, the image label does not provide very efficient supervision on the spatial assembly learning.

4.5. Ablation Studies

In the following, we perform ablation studies to further understand the performance of the proposed spatial assembly network.

(1) Performance contribution of the SAN module. In this ablation study, we aim to identify the contribution of our proposed SAN module on different datasets. Table 5 summarizes the performance results on the CUB and SOP datasets with and without using the SAN module in both supervised and unsupervised deep metric learning. The baseline system is using the multi-similarity [42] with momentum memory bank [10]. The momentum memory bank has a contrastive-based loss [15]. We can see that our proposed SAN module significantly improves the performance by a large margin. Figure 5 shows the retrieval examples by the baseline with and without our SAN module on the CUB, Cars, SOP, and In-Shop datasets from supervised metric learning. The top row shows the retrieval results by the baseline, and the bottom row shows the results for the baseline plus the SAN module. Samples highlighted with blue and red boxes are query images and incorrect retrieval results. We can see that, using the SAN module, the number of incorrect retrieval results have been significantly reduced

Table 1. Recall@K (%) performance on the CUB and Cars, and In-Shop datasets with **GoogleNet** in comparison with other supervised metric learning methods. Some papers did not report results on specific datasets, which are marked with -.

Methods	CUB				Cars				In-Shop			
	R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8	R@1	R@10	R@20	R@30
LiftedStruct [28] CVPR16	47.2	58.9	70.2	80.2	49.0	60.3	72.1	81.5	-	-	-	-
Histogram Loss [38] NIPS16	50.3	61.9	72.6	82.4	-	-	-	-	-	-	-	-
N-Pair Loss [35] NIPS16	51.0	63.3	74.3	83.2	71.1	79.7	86.5	91.6	-	-	-	-
Clustering [27] CVPR17	48.2	61.4	71.8	81.9	58.1	70.6	80.3	87.8	-	-	-	-
BIER [29] ICCV17	55.3	67.2	76.9	85.1	78.0	85.8	91.1	95.1	76.9	92.8	95.2	96.2
Angular Loss [41] ICCV17	54.7	66.3	76.0	83.9	71.4	81.4	87.5	92.1	-	-	-	-
MS [42] CVPR19	58.2	69.8	79.9	87.3	75.7	84.6	90.1	94.4	85.1	96.7	97.8	98.3
HDML [51] CVPR19	53.7	65.7	76.7	85.7	79.1	87.1	92.1	95.5	-	-	-	-
A-BIER [30] TPAMI18	57.5	68.7	78.3	86.2	82.0	89.0	93.2	96.1	83.1	95.1	96.9	97.5
XBM [43] CVPR20	61.9	72.9	81.2	88.6	80.3	87.1	91.9	95.1	89.1	97.3	98.1	98.4
Proposed	**63.3**	**74.5**	**83.8**	**90.4**	**83.5**	**89.7**	**93.4**	**96.1**	**92.5**(88.5)	**98.9**(97.5)	**99.3**(98.2)	**99.5**(98.6)
Gain	**+1.4**	**+1.6**	**+2.6**	**+1.8**	**+1.5**	**+0.7**	**+0.2**	**+0.0**	**+3.4**(-)	**+1.6**(0.2)	**+1.2**(0.1)	**+1.1**(0.2)

Table 2. Recall@K (%) performance on the CUB, Cars, and SOP datasets with **GoogleNet** in comparison with other unsupervised metric learning methods.

Methods	CUB				Cars				SOP		
	R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8	R@1	R@10	R@100
Examplar [6] *TPAMI16*	38.2	50.3	62.8	75.0	36.5	48.1	59.2	71.0	45.0	60.3	75.2
NCE [45] CVPR18	39.2	51.4	63.7	75.8	37.5	48.7	59.8	71.5	46.6	62.3	76.8
DeepCluster [2] ECCV18	42.9	54.1	65.6	76.2	32.6	43.8	57.0	69.5	34.6	52.6	66.8
MOM [13] CVPR18	45.3	57.8	68.6	78.4	35.5	48.2	60.6	72.4	43.3	57.2	73.2
AND [12] ICML19	47.3	59.4	71.0	80.0	38.4	49.6	60.2	72.9	47.4	62.6	77.1
ISIF [48] CVPR19	46.2	59.0	70.1	80.2	41.3	52.3	63.6	74.9	48.9	64.0	78.0
aISIF [47] TPAMI20	47.7	59.9	71.2	81.4	41.2	52.6	63.8	75.1	49.7	65.4	79.5
CBSwR [26] BMVC20	47.5	59.6	70.6	80.5	42.6	54.4	65.4	76.0	-	-	-
PSLR [46] CVPR20	48.1	60.1	71.8	81.6	43.7	54.8	66.1	76.2	51.1	66.5	79.8
Proposed	**55.9**	**68.0**	**78.6**	**86.8**	**44.2**	**55.5**	**66.8**	**76.9**	**58.7**	**73.1**	**84.6**
Gain	**+7.8**	**+7.9**	**+6.2**	**+5.2**	**+0.5**	**+0.7**	**+0.7**	**+0.7**	**+7.6**	**+6.6**	**+4.8**

Table 3. Recall@K (%) performance on the SOP dataset using **Resnet-18** network without pre-trained parameters.

Methods	SOP		
	R@1	R@10	R@100
Random	18.4	29.4	46.0
Examplar [6] TPAMI16	31.5	46.7	64.2
NCE [45] CVPR18	34.4	49.0	65.2
MOM [13] CVPR18	16.3	27.6	44.5
AND [12] ICML19	36.4	52.8	67.2
ISIF [48] CVPR19	39.7	54.9	71.0
aISIF [47] TPAMI20	40.7	55.9	72.2
PSLR [46] CVPR20	42.3	57.7	72.5
Proposed	**46.3**	**61.9**	**77.0**
Gain	**+4.0**	**+4.2**	**+4.5**

because the learned feature is much more discriminative.

(2) Performance of SAN module with different metric learning losses. In order to verify the generalization capability of our method, we conduct experiments to show the performance of our proposed SAN module with different metric learning losses and different backbone networks. It should be noted that the momentum memory bank [10, 15] in the baseline system is not included in this experiment. We evaluate the MS loss [42] with SAN on GoogleNet backbone and Proxy-Anchor [18] loss with SAN on BN-inception backbone. From the Table 6, we can see that the MS loss [42] with SAN has improved the Recall@1 rate by 1.5% and the Proxy-Anchor [18] with SAN has improved the Recall@1 rate by 1.1%.

Table 4. Classification accuracy (%) on the CIFAR-10 and CIFAR-100 dataset using **VGG-19** network with and without the SAN module.

Methods	CIFAR-10	CIFAR-100
VGG-19	93.23 %	72.13%
VGG-19 with SAN	**93.91**%	**72.77**%

In the **Supplementary Materials**, we provide additional experimental results, implementation details, and ablation studies.

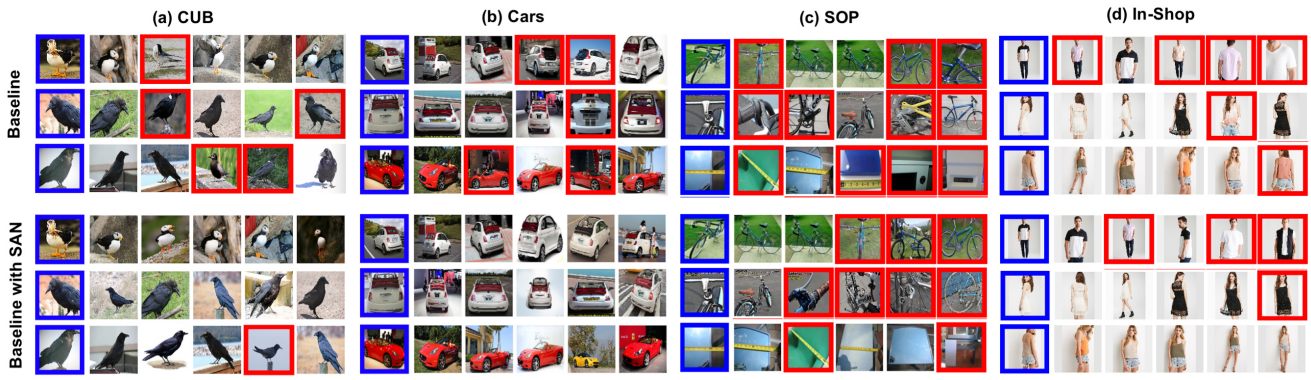

Figure 5. Retrieval examples by the baseline with and without our SAN module on the CUB, Cars, SOP, and In-Shop datasets. The query images and the incorrect retrieved images are highlighted with *blue* and *red*.

Table 5. The Recall@K performance of the baseline and baseline with our proposed SAN module on the CUB and SOP datasets.

Supervised Metric Learning				
Methods	**CUB**			
	R@1	R@2	R@4	R@8
Baseline	61.8	73.2	82.3	88.9
+ SAN	**63.3**	**74.5**	**83.8**	**90.4**
Gain	**+1.5**	**+1.3**	**+1.5**	**+1.5**
Methods	**SOP**			
	R@1	R@10	R@100	R@1000
Baseline	73.7	87.9	95.0	98.4
+ SAN	**75.8**	**89.2**	**95.5**	**98.6**
Gain	**+2.1**	**+1.3**	**+0.5**	**+0.2**
Unsupervised Metric Learning				
Methods	**CUB**			
	R@1	R@2	R@4	R@8
Baseline	53.3	66.1	77.4	85.6
+ SAN	**55.9**	**68.0**	**78.6**	**86.8**
Gain	**+2.6**	**+1.9**	**+1.2**	**+1.2**
Methods	**SOP**			
	R@1	R@10	R@100	
Baseline	56.9	71.2	82.7	
+ SAN	**58.7**	**73.1**	**84.6**	
Gain	**+1.8**	**+1.9**	**+1.9**	

Table 6. Recall@K (%) performance on SAN with Multi-Similarity (MS) loss and Proxy-Anchor loss on the CUB dataset. 'G' denotess **GoogleNet**, 'BN' denotes **BN-inception**.

Methods		**CUB**			
		R@1	R@2	R@4	R@8
MS [18] CVPR19	G	58.2	69.8	79.9	87.3
MS with SAN	G	**59.7**	**72.0**	**81.4**	**88.4**
Gain		**+1.5**	**+2.2**	**+1.5**	**+1.1**
Proxy-Anchor [18] CVPR20	BN	68.4	79.2	**86.8**	91.6
Proxy-Anchor with SAN	BN	**69.5**	**79.3**	86.7	**92.0**
Gain		**+1.1**	**+0.1**	**-**	**+0.4**

5. Conclusion

In this work, we have successfully developed a new spatial assembly network to explore the spatial variations caused by changes in object part configurations, spatial layout of object, and scene structures of the images. This SAN module examines the input image and perform a learned re-organization and assembly of feature points from different spatial locations conditioned by feature maps from previous network layers so as to maximize the discriminative power of the final feature representation. The proposed spatial assembly goes beyond spatial image transforms. It learns to reorganize or re-assemble the feature maps across different spatial locations. This work represents one of the first efforts to explore spatial reorganization of feature maps for 2D images. The proposed spatial assembly represents a more generic feature operation than simple permutation. This differentiable module can be directly incorporated into existing deep neural network for end-to-end training to increase network robustness under spatial variations and improve the discriminative power of image features In our experiments, we have demonstrated that the proposed SAN module is able to significantly improve the performance of various metric / representation learning, image retrieval and classification tasks, in both supervised and unsupervised learning scenarios.

Acknowledgement

This work was supported in part by National Science Foundation under grants 1647213 and 1646065. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] RE Burkard and E Cela. The quadratic assignment problem, in "handbook of combinatorial optimization" vol. 3. edited by dz du, pm pardalos, 1999. 2

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018. 6, 7

[3] Taco S Cohen and Max Welling. Transformation properties of learned visual representations. *arXiv preprint arXiv:1412.7659*, 2014. 2

[4] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. 2

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 2

[6] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747, 2015. 6, 7

[7] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012. 2

[8] Alhussein Fawzi and Pascal Frossard. Manitest: Are classifiers really invariant? *arXiv preprint arXiv:1507.06535*, 2015. 1

[9] Robert Gens and Pedro M Domingos. Deep symmetry networks. In *Advances in neural information processing systems*, pages 2537–2545, 2014. 2

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019. 6, 7

[11] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pages 2731–2741. PMLR, 2019. 2

[12] Jiabo Huang, Qi Dong, Shaogang Gong, and Xiatian Zhu. Unsupervised deep learning by neighbourhood discovery. In *International Conference on Machine Learning*, pages 2849–2858, 2019. 6, 7

[13] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Mining on manifolds: Metric learning without labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7642–7651, 2018. 6, 7

[14] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 2

[15] Shichao Kan, Yigang Cen, Yang Li, Mladenovic Vladimir, , and Zhihai He. Contrastive bayesian analysis for supervised deep metric learning. In *Github*, 2020. 6, 7

[16] Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Geometric robustness of deep networks: analysis and improvement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4441–4449, 2018. 2

[17] Osman Semih Kayhan and Jan C van Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285, 2020. 1

[18] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3238–3247, 2020. 7, 8

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[20] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 5

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[22] Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 289–297, 2016. 1, 2

[23] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015. 1, 2

[24] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1096–1104, 2016. 5

[25] Jiancheng Lyu, Shuai Zhang, Yingyong Qi, and Jack Xin. Autoshufflenet: Learning permutation matrices via an exact lipschitz continuous penalty in deep convolutional neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 608–616, 2020. 1, 2

[26] Binh X Nguyen, Binh D Nguyen, Gustavo Carneiro, Erman Tjiputra, Quang D Tran, and Thanh-Toan Do. Deep metric learning meets deep clustering: An novel unsupervised approach for feature embedding. *arXiv preprint arXiv:2009.04091*, 2020. 6, 7

[27] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5382–5390, 2017. 6, 7

[28] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on*

computer vision and pattern recognition, pages 4004–4012, 2016. 5, 6, 7

[29] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Bier-boosting independent embeddings robustly. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5189–5198, 2017. 6, 7

[30] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 5, 6, 7

[31] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 6

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 5, 6

[33] Xu Shen, Xinmei Tian, Anfeng He, Shaoyan Sun, and Dacheng Tao. Transform-invariant convolutional neural networks for image classification and search. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 1345–1354, 2016. 2

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[35] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*, pages 1857–1865, 2016. 2, 6, 7

[36] Kihyuk Sohn and Honglak Lee. Learning invariant representations with local transformations. *arXiv preprint arXiv:1206.6418*, 2012. 2

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 5, 6

[38] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016. 6, 7

[39] Joshua T Vogelstein, John M Conroy, Vince Lyzinski, Louis J Podrazik, Steven G Kratzer, Eric T Harley, Donniell E Fishkind, R Jacob Vogelstein, and Carey E Priebe. Fast approximate quadratic programming for graph matching. *PLOS one*, 10(4):e0121002, 2015. 2

[40] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 5

[41] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017. 6, 7

[42] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019. 4, 5, 6, 7

[43] Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. Cross-batch memory for embedding learning. *arXiv preprint arXiv:1912.06798*, 2019. 5, 6, 7

[44] Zhongdao Wang, Luming Tang, Xihui Liu, Zhuliang Yao, Shuai Yi, Jing Shao, Junjie Yan, Shengjin Wang, Hongsheng Li, and Xiaogang Wang. Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 379–387, 2017. 2

[45] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018. 6, 7

[46] Mang Ye and Jianbing Shen. Probabilistic structural latent representation for unsupervised embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5457–5466, 2020. 5, 6, 7

[47] Mang Ye, Jianbing Shen, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Augmentation invariant and instance spreading feature for softmax embedding. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 5, 6, 7

[48] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6210–6219, 2019. 6, 7

[49] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016. 2

[50] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. 2

[51] Wenzhao Zheng, Zhaodong Chen, Jiwen Lu, and Jie Zhou. Hardness-aware deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 72–81, 2019. 5, 6, 7