3D Object Representation Learning: A Set-to-Set Matching Perspective

Tan Yu, Jingjing Meng, Ming Yang, *Member, IEEE*, and Junsong Yuan, *Fellow, IEEE*

***Abstract*— In this paper, we tackle the 3D object representation learning from the perspective of set-to-set matching. Given two 3D objects, calculating their similarity is formulated as the problem of set-to-set similarity measurement between two set of local patches. As local convolutional features from convolutional feature maps are natural representations of local patches, the set-to-set matching between sets of local patches is further converted into a local features pooling problem. To highlight good matchings and suppress the bad ones, we exploit two pooling methods: 1) bilinear pooling and 2) VLAD pooling. We analyze their effectiveness in enhancing the set-to-set matching and meanwhile establish their connection. Moreover, to balance different components inherent in a bilinear-pooled feature, we propose the harmonized bilinear pooling operation, which follows the spirits of intra-normalization used in VLAD pooling. To achieve an end-to-end trainable framework, we implement the proposed harmonized bilinear pooling and intra-normalized VLAD as two layers to construct two types of neural network, multi-view harmonized bilinear network (MHBN) and multi-view VLAD network (MVLADN). Systematic experiments conducted on two public benchmark datasets demonstrate the efficacy of the proposed MHBN and MVLADN in 3D object recognition.**

***Index Terms*— 3D object recognition, convolutional neural network, bilinear pooling.**

I. Introduction

INSPIRED by the success of deep learning in 2D images, the community has also attempted to exploit convolutional neural networks for 3D object recognition [1]–[11]. These approaches can be coarsely classified into three categories according to their inputs: 1) view-based methods [2], [5], [6], [12], 2) volume-based methods [1], [3], [4], [7], [8], [13], and 3) pointset-based methods [9]–[11]. View-based methods project 3D objects into multiple 2D views, then the classification is conducted using the features from 2D CNNs. Volume-based approaches apply 3D convolutional neural networks directly on voxelized shapes, while pointset-based methods directly take unordered point sets as input. Among the three categories, the view-based methods generally outperform the other two. Even though one volume-based work, VRN Ensemble [13] outperforms existing view-based methods, its excellent performance is mainly attributed to the model ensemble and a more advanced base model.

Despite that view-based methods [2], [5], [6], [12] have already achieved excellent performance in 3D object recognition, there are still some drawbacks in current methods. For instance, one of the earliest view-based method based on deep neural network, multi-view CNN (MVCNN) [2], max-pools the view-wise feature into a global feature as the representation of the 3D object. Nevertheless, pooling operations generally lose some information in aggregating features of multiple views to the maximal activation (max-pooling) or the sum of all activations (sum-pooling). Meanwhile, the experiments in [2] show that the performance of sum-pooling is even worse than max-pooling. Below we investigate the reason. To be specific, we define $\mathcal{A}$ as a 3D object and $\{\mathbf{X}_i\}_{i=1}^n$ as features of n projected views. In the same manner, we define another 3D object $\mathcal{B}$ and its projected views' features $\{\mathbf{Y}_j\}_{j=1}^n$. Through sum-pooling, descriptors of two objects are obtained by

$$\mathbf{X} = \sum_{i=1}^{n} \mathbf{X}_i, \quad \mathbf{Y} = \sum_{j=1}^{n} \mathbf{Y}_j. \tag{1}$$

Below we analyse the effectiveness of a feature from the perspective of feature similarity. Note that, optimizing the feature similarity, *i.e.*, metric learning, is a common practice to boost the retrieval accuracy. Despite that this paper focuses on classification, the effectiveness of a metric is vital to classification since we adopt a linear classifier. The similarity between two sum-pooled features is

$$sim(\mathcal{A}, \mathcal{B}) = \langle \mathbf{X}, \mathbf{Y} \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{X}_i, \mathbf{Y}_j \rangle, \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner-product operation of two vectors. It is easy to observe from above equation that, the similarity between two sum-pooled global features will be equal to the summation of similarities of every matching pair $(\mathbf{X}_i, \mathbf{Y}_j)$. Nevertheless, as shown in Figure 1(a), only the similarity between two corresponding views (green solid arrows) can reliably capture the relevance between two 3D objects whereas the similarities from non-corresponding views (red dash arrows) are low due to view-point variations. Nevertheless, given two 3D objects represented by n views each, there are only n pairs of corresponding views whereas the other $n^2 - n$ pairs consist of non-corresponding views. Thus the

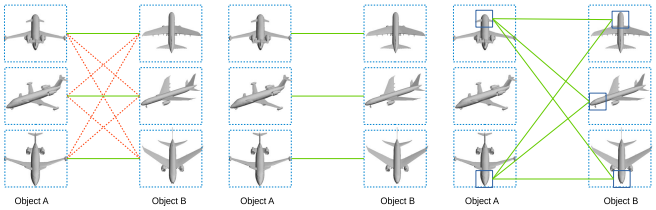
Manuscript received November 18, 2018; revised September 20, 2019; accepted December 22, 2020. Date of publication January 20, 2021; date of current version January 27, 2021. This work was supported in part by the Horizon Robotics and National Science Foundation Grant CNS-1951952. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Laura Toni. *(Corresponding author: Tan Yu.)*

Tan Yu is with the Cognitive Computing Laboratory, Baidu Research, Bellevue, WA 98004 USA (e-mail: tyu008@e.ntu.edu.sg).

Jingjing Meng and Junsong Yuan are with the Department of Computer Science and Engineering, The State University of New York, University at Buffalo, Buffalo, NY 14260 USA.

Ming Yang is with Horizon Robotics, Inc., Cupertino, CA 95014 USA.

Digital Object Identifier 10.1109/TIP.2021.3049968



(a) sum-pool MVCNN (b) GIFT (c) patch-based method

Fig. 1. The comparisons among sum pooling, GIFT and patch-based method. The sum-pooling suffers from being contaminated by overnumbered non-corresponding pairs. GIFT greedily selects the best-matched view and might discard the potentially relevant ones. Our patch-based method hooks up relevant patch pairs no matter if they are from two corresponding views or not, and meanwhile decouples the pairs of irrelevant patches.

pairs of corresponding views can be easily contaminated by overnumbered non-corresponding pairs.

To mitigate the negative influence from pairs of non-corresponding views, GIFT [6] utilizes a robust version of Hausdorff distance defined as

$$sim(\mathcal{A}, \mathcal{B}) = \sum_{i=1}^{N} \max_{j} \langle \mathbf{X}_i, \mathbf{Y}_j \rangle. \tag{3}$$

The intuition behind the above distance is that, for each view of $\mathcal{A}$, it only takes into account the similarity with its most relevant view of $\mathcal{B}$. Therefore, GIFT only keeps the pairs consisting of two corresponding views (green solid arrows) and eliminates the pairs consisting of two non-corresponding views (red dash arrows). The experiments in GIFT [6] shows that it achieves a better performance than MVCNN [2].

However, GIFT [6], [14] requires to store the feature of every view of the 3D objects separately whereas MVCNN [2] only needs to store one global max-pooled feature. Meanwhile, the GIFT requires to cross-match every view of $\mathcal{A}$ with every view of $\mathcal{B}$, taking n^2 times comparisons where n is the number of views. In contrast, MVCNN only needs directly compare two global features representing two 3D objects, which is much faster than GIFT. Using hamming embedding and inverted-indexing, GIFT can achieve comparable efficiency as MVCNN in retrieval tasks. As shown in Figure 1 (b), it only counts the best-matched views and mitigate the negative effect from similarities between non-corresponding views. Nevertheless, even if two views from two 3D objects differ significantly in global appearance due to view-point variations, there might exist some relevant patches between these two views locally. Greedily selecting the best-matched view can remove the distraction from pairs of non-corresponding views but it also discards the useful information.

Observing the limitations of view-wise pooling used in MVCNN and GIFT, we propose to pool feature in *patch* level rather than *view* level to obtain a more reasonable similarity measure between two 3D objects. As shown in Figure 1 (c), ideally, we seek to hook up relevant patch pairs no matter they are from two corresponding views or not and meanwhile decouple the pairs of irrelevant patches. Since a local convolutional feature is a natural and effective representation of a local patch, a 3D object can be characterized through a set of local convolutional features and thus the similarity measurement between 3D objects is converted into a set-to-set matching problem. To achieve an effective set-to-set matching, we exploit two set-to-set matching kernels, polynomial set-to-set matching kernel (PSMK) and local set-to-set matching kernel (LSMK).

PSMK adaptively assigns higher weights to good matching pairs and lower weights to bad ones. Note that directly evaluating PSMK requires comparing each local patch from one object with all local patches from another object, which is costly given the huge number of local patches of each object. To alleviate the computational cost, we exploit the connection between bilinear pooling methods and polynomial kernel, achieving the same functionality of PSMK through bilinear pooling in a more efficient manner. In this case, the set-to-set similarity computing complexity is reduced from $\mathcal{O}(N^2 d)$ to $\mathcal{O}(d^2)$ (where N is the number of patches for each 3D object and d is the dimension of local feature) thanks to global bilinear-pooled representation. To further improve the representation's discriminative power, we extend the original bilinear pooling to harmonized bilinear pooling, which learns an element-wise Box-Cox [15] transformation for each singular value from the training data.

LSMK partitions the feature space into multiple cells through clustering on local convolutional features. The patches are assigned to different clusters according to the similarities between their corresponding local convolutional features with cluster centroids. In this case, when comparing the similarity between two 3D objects, only the local patches which are assigned to the same cluster are compared, which effectively decouple the irrelevant patches since irrelevant patches tend to be assigned to different clusters due to their dissimilarity in appearance. Similar to PSMK, directly evaluating LSMK is also computationally intensive. To boost its efficiency, we exploit the connection between LSMK and vector of locally aggregated descriptor (VLAD), achieving the same functionality of LSMK through aggregating local convolutional features from all projected views through VLAD.

Note that, if the matching between two objects is perfect, we might not need an advanced set-to-set matching kernel and the sum matching kernel might be enough. Nevertheless, in real scenarios, due to misalignment and occlusion, the good matching pairs might only exist in several small regions, which requires a more advanced set-to-set matching kernel such as the PSMK and LSMK proposed in this paper to highlight the good matchings in small regions.

Even though the above two strategies are quite different from the perspective of algorithms, their goals are quite similar. Concretely, both of them aim to highlight pairs of relevant patches and suppress pairs of irrelevant patches. To incorporate them into neural network, we implement these two strategies as two layers, which construct two deep neural networks, respectively. Both of the constructed neural network can be trained in an end-to-end manner. Systematic experiments on two public benchmark datasets, Modelnet10 and Modelnet40, demonstrate the effectiveness of the proposed methods. Note that, this paper is an extended version of our previsouly published work [16]. In summary, our method has following contributions:

- Unlike existing view-based methods pooling view-wise global features, we tackle the 3D object recognition from the perspective of patches-to-patches similarity measurement, which is in essence, a set-to-set matching problem.
- To achieve an effective set-to-set similarity, we seek to highlight pairs consisting of relevant patches whereas suppress pairs of irrelevant patches. To this end, we exploit two set-to-set matching kernels, PSMK and LSMK.
- To boost efficiency of matching kernels and obtain the global representation of 3D objects, we propose to implicitly embed the matching kernels when aggregating local features. We discover that bilinear pooling embeds the PSMK and VLAD embeds LSMK.
- We propose the harmonized bilinear pooling, balancing singular values of the pooled bilinear features by learning a Box-Cox transformation on the singular values.
- We implement harmonized bilinear pooling and VLAD as two layers and use them to construct two multi-view convolutional neural networks, respectively. Both of them can be trained in an end-to-end manner.

II. Related Work

A. View-Based Methods

MVCNN [2] projects a 3D object into multiple views, extracts view-wise CNN features and max-pools them into a global representation of the 3D object. GIFT [6] also extracts view-wise features but does not pool them. Instead, it obtains the similarity between two 3D objects by view-wise matching. Recently, Wang *et al.* [12] recurrently cluster the views into multiple sets, pool the features in each set and achieve better performance than the original MVCNN. In parallel, GVCNN [17] first groups the view-wise features to obtain the group-wise features and further obtains the global feature by fusing all group-wise features, achieving good performance in 3D object recognition.

B. Volume-Based Methods

Some works [1], [3], [4] apply 3D convolutional neural networks directly on voxelized shapes. These methods are constrained by their resolution owing to data sparsity and costly computation of 3D convolution. Generally speaking, the performance of volume-based methods is not as good as view-based methods. Nevertheless, one of volume-based methods, VRN-Ensemble [13], achieves better performance than all existing view-based methods on two public datasets. However, its excellent performance is attributed to model ensemble and a more advanced base model architecture. It ensembles 5 ResNet [18] models and one Inception model whereas most existing view-based methods are based on a single VGG-M model. As shown in [13], using a single ResNet model, its performance is not as good as the view-based methods. More recently, Xie *et al.* [19] proposed a descriptor network following an "analysis by synthesis" scheme, achieving excellent performance in both 3D object recovery and recognition tasks.

C. Pointset-Based Methods

PointNet proposed by Qi *et al.* [9] directly takes unordered point sets as inputs, addressing the sparsity problem encountered in volume-based methods. In parallel, Klokov and Lempitsky [11] propose Kd-Networks for the recognition of 3D object represented by 3D point cloud. To further improve the performance of PointNet, Qi *et al.* [10] propose an improved PointNet++ through exploiting local structures induced by the metric space. Recently, Su *et al.* proposed SPLATNet [20] which conduct hierarchical and spatially-aware feature learning for unordered points, achieving outstanding performance in 3D object segmentation. Meanwhile, PointGrid [21], a 3D convolutional neural network, samples a fixed number of points within each cell, achieving an excellent performance in 3D object recognition and segmentation based on low-resolution mesh grid.

D. Bilinear Pooling

Bilinear pooling was firstly proposed by Tenenbaum and Freeman [22] to separate style and content. It was traditionally used in pooling hand-crafted features [23]. Recently it has been used in pooling local convolutional features [24]–[27] to take the second-order statistics into consideration, achieving state-of-the-art performance in fine-grained image classification. Our work is closely related to bilinear pooling. Nevetheless, different from works [24]–[29] utilizing bilinear pooling to obtain the second-order statistics for fine-grained classification, the bilinear pooling in our work is to more efficiently achieve the functionality of polynomial set-to-set similarity measurement. Moreover, we conduct the harmonizing operation on the pooled bilinear feature to obtain more discriminative representations of 3D objects.

III. Set-to-Set Matching Kernel

In this section, we analyze the effectiveness of several set-to-set matching kernels. We define $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ as the set of local convolutional features from all views of object $\mathcal{A}$. Each local convolutional feature represents a local patch from one view of $\mathcal{A}$. We define $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^N$ as the local convolutional features from another object $\mathcal{B}$. We define four simple set-to-set matching kernel: sum set-to-set matching kernel (SSMK) and maximum set-to-set matching kernel (MSMK), polynomial set-to-set matching kernel (PSMK) and local set-to-set matching kernel (LSMK).

A. Sum Set-to-Set Matching Kernel (SSMK)

$\mathrm{SSMK}(\mathcal{X}, \mathcal{Y})$ measures the sum of similarities between each point in $\mathcal{X}$ and each point in $\mathcal{Y}$:

$$\mathrm{SSMK}(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{x}\in\mathcal{X}} \sum_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{x}, \mathbf{y} \rangle. \tag{4}$$

The SSMK is robust to the noise since it takes similarity of every possible matching pair into consideration. Nevertheless, it treats each matching pair equally and thus good matching pairs could be easily swamped by bad ones.

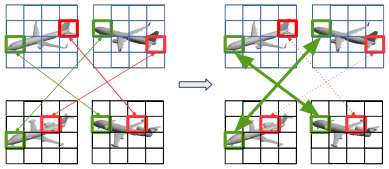


Fig. 2. By adaptively assigning different weights, good matching pairs are highlighted whereas the bad ones are suppressed.

B. Maximum Set-to-Set Matching Kernel (MSMK)

In contrast, $\mathrm{MSMK}(\mathcal{X}, \mathcal{Y})$ measures the maximum similarity between points in $\mathcal{X}$ and points in $\mathcal{Y}$:

$$\mathrm{MSMK}(\mathcal{X}, \mathcal{Y}) = \max_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{x}, \mathbf{y} \rangle. \tag{5}$$

The MSMK only takes the best matching pair into consideration and effectively suppresses bad influence from bad matching pairs. Nevertheless, taking only the best matching pair makes it throw away much useful information and sensitive to noise.

C. Polynomial Set-to-Set Matching Kernel (PSMK)

To overcome drawbacks of SSMK and MSMK, we propose a polynomial set-to-set matching kernel (PSMK_p):

$$\mathrm{PSMK}_p(\mathcal{X}, \mathcal{Y}) = \Big(\sum_{\mathbf{x}\in\mathcal{X}} \sum_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{x}, \mathbf{y} \rangle^p \Big)^{\frac{1}{p}}. \tag{6}$$

It is not difficult to observe that both SSMK and MSMK are special cases of PSMK_p. To be specific, SSMK corresponds to the condition when $p = 1$ whereas MSMK corresponds to the case when $p \to +\infty$. By choosing a not too large value $p \in (1, +\infty)$, the similarity measurement can simultaneously suppress the bad matchings and meanwhile be robust to noise.

Let us consider a a special case by setting $p = 2$ and remove the $\frac{1}{p}$ entry in the original PSMK_p since it does not change the order of the similarities:

$$\mathrm{PSMK}_2(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{x}\in\mathcal{X}} \sum_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{x}, \mathbf{y} \rangle^2 \tag{7}$$

By rewriting $\langle \mathbf{x}, \mathbf{y} \rangle^2 = w(\mathbf{x}, \mathbf{y}) \langle \mathbf{x}, \mathbf{y} \rangle$, it is not difficult to find that larger weights $w(\mathbf{x}, \mathbf{y})$ are assigned to the good matchings with larger value of $\langle \mathbf{x}, \mathbf{y} \rangle$ whereas the smaller weights are assigned to the bad ones as illustrated in Figure 2. It can be regarded as a soft and robust version of MSMK.

D. Local Set-to-Set Matching Kernel (LSMK)

Meanwhile, we propose another set-to-set matching kernel, which we define as local set-to-set matching kernel (LSMK), achieving the same function as PSMK. LSMK is based on a pre-learned feature space partition. Concretely, as shown in Figure 3, LSMK partitions the feature space into K cells and each local feature $\mathbf{x}$ is assigned to the cells with index $i(\mathbf{x})$ according to its similarity with the cells' centroids $\{\mathbf{c}_k\}_{k=1}^K$. Given two sets of local features $\mathcal{X}$ and $\mathcal{Y}$, LSMK is computed by

$$\mathrm{LSMK}(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{x}\in\mathcal{X}} \sum_{\mathbf{y}\in\mathcal{Y}} \mathbb{1}(i(\mathbf{x}) = i(\mathbf{y})) \langle \mathbf{x}, \mathbf{y} \rangle, \tag{8}$$

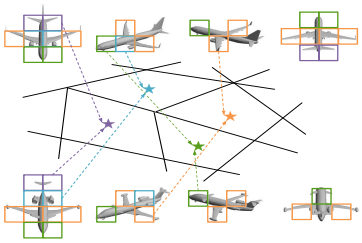


Fig. 3. After the feature space partition, the relevant patches which are closed in feature space will be assigned to the same cell. In contrast, the irrelevant patches will be divided into different cells.

where $\mathbb{1}(i(\mathbf{x}) = i(\mathbf{y}))$ is an indicator function which is non-zero if and only if the local features $\mathbf{x}$ and $\mathbf{y}$ are assigned to the same cell. The feature space partition can be learned through k-means in an unsupervised manner, but later we will show it can be learned in an end-to-end manner.

IV. Kernel Embedding

In the previous section, we analyze the effectiveness of several set-to-set matching kernels. Nevertheless, to evaluate the matching kernel, we need to exhaustively compare every local feature from $\mathcal{X}$ and every local feature from $\mathcal{Y}$, which is considerably time-consuming. To boost the efficiency, we propose to implicitly embed the matching kernel when aggregating the local features. After the kernel embedding, each 3D object is represented by a global feature and the inner-product between two objects' global features is equivalent to the output of embedded matching kernel. Since computing the inner-product between global features are much faster than exhaustively comparing every pair of local features, the efficiency is significantly boosted. Moreover, the aggregated global feature is more flexible in further process compared with original set of discrete local features. Concretely, the global feature can be used as the input of any classifier such as SVM and softmax whereas the original local feature sets based on matching kernels can only rely on nearest neighbour classifier.

A. PSMK Embedding

In fact, $\langle \mathbf{x}, \mathbf{y} \rangle^p$ is a special case of polynomial kernel $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\top \mathbf{y} + c)^p$ when $c = 0$. It has an explicit feature map defined as

$$\langle \mathbf{x}, \mathbf{y} \rangle^p = \langle vec(\underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{p}), vec(\underbrace{\mathbf{y} \otimes \cdots \otimes \mathbf{y}}_{p}) \rangle, \tag{9}$$

where $\otimes$ represents the outer-product operation and $\underbrace{\otimes \cdots \otimes}_{p}$ denotes p times outer product. Concretely, let us consider the case when $p = 2$ and obtain

$$\begin{aligned} \langle \mathbf{x}, \mathbf{y} \rangle^2 &= \langle vec(\mathbf{x} \otimes \mathbf{x}), vec(\mathbf{y} \otimes \mathbf{y}) \rangle \\ &= \langle vec(\mathbf{x}\mathbf{x}^\top), vec(\mathbf{y}\mathbf{y}^\top) \rangle. \end{aligned} \tag{10}$$

Thus we can rewrite Eq. (7) by

$$\mathrm{PSMK}_2(\mathcal{X}, \mathcal{Y}) = \langle vec(\sum_{\mathbf{x}\in\mathcal{X}} \mathbf{x}\mathbf{x}^\top), vec(\sum_{\mathbf{y}\in\mathcal{Y}} \mathbf{y}\mathbf{y}^\top) \rangle, \tag{11}$$

where $\sum_{\mathbf{x}\in\mathcal{X}_A} \mathbf{x}\mathbf{x}^\top$ and $\sum_{\mathbf{y}\in\mathcal{X}_B} \mathbf{y}\mathbf{y}^\top$ are bilinear-pooled features of object $\mathcal{A}$ and $\mathcal{B}$, respectively. That it, the PSMK_2

between two sets of local features are equal to the inner-product similarity between two bilinear pooled features. In other words, bilinear pooling implicitly embeds the PSS_2 set-to-set matching kernel.

Let us define the number of local features within the set as $N = |\mathcal{X}| = |\mathcal{Y}|$. Without kernel embedding, computing $\text{PSMK}_2(\mathcal{X}, \mathcal{Y})$ requires N^2 times comparisons between every pair of d-dimension local features, taking $\mathcal{O}(dN^2)$ time complexity. After kernel embedding, we just need to compute an inner product between two d^2-dimension global features, taking $\mathcal{O}(d^2)$ time complexity. In our case, $d \ll N^2$, thus the efficiency is significantly boosted after kernel embedding. Note that,

$$vec(\underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{p}) \in \mathbb{R}^{d^p}, \tag{12}$$

which means that the dimension of pooled feature increases exponentially as p increases. When $p > 2$, the dimension of the pooled feature will be extremely high, making both memory and computational cost extremely huge. Therefore, considering both effectiveness and efficiency, we select $p = 2$.

Note that, the proposed PSMK kernel resembles the generalized mean pooling proposed in [30]. But they are quite different in the motivation and formulation. To be specific, given two sets of features $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ and $\mathcal{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_N\}$, generalized mean pooling conducts on each set individually:

$$\begin{aligned} \bar{\mathbf{x}}[k] &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i[k]^p, \\ \bar{\mathbf{y}}[k] &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_i[k]^p, \end{aligned} \tag{13}$$

where $\bar{\mathbf{x}}[k]$ and $\bar{\mathbf{y}}[k]$ denotes the k-th element of the pooled global feature of $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$. The similarity between $\mathcal{X}$ and $\mathcal{Y}$ based on the generalized mean pooling (GMP) is

$$\text{sim}_{\text{GMP}}(\mathcal{X}, \mathcal{Y}) = \langle \bar{\mathbf{x}}, \bar{\mathbf{y}} \rangle. \tag{14}$$

It is straightforward to see that $\text{sim}_{\text{GMP}}(\mathcal{X}, \mathcal{Y})$ based on generalized mean pooling is not equivalent to the proposed $\text{PSMK}(\mathcal{X}, \mathcal{Y})$. Meanwhile, GMP cannot achieve the function of highlighting good matchings achieved by our PSMK.

B. LSMK Embedding

Let us consider $\text{LSMK}(\mathcal{X}, \mathcal{Y})$ contributed by the k-th cell:

$$\text{LSMK}^k(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{x}\in\mathcal{X}} \sum_{\mathbf{y}\in\mathcal{Y}} \mathbb{1}(i(\mathbf{x}) = i(\mathbf{y}) = k)\langle \mathbf{x}, \mathbf{y} \rangle. \tag{15}$$

We define the sum-pooled vector in the k-th cell from $\mathcal{X}_A$ and that from $\mathcal{X}_B$ as

$$\begin{aligned} \mathbf{v}_x^k &= \sum_{\mathbf{x}\in\mathcal{X}} \mathbb{1}(i(\mathbf{x}) = k)\mathbf{x}, \\ \mathbf{v}_y^k &= \sum_{\mathbf{y}\in\mathcal{Y}} \mathbb{1}(i(\mathbf{y}) = k)\mathbf{y}. \end{aligned} \tag{16}$$

It is not difficult to observe that

$$\langle \mathbf{v}_x^k, \mathbf{v}_y^k \rangle = \text{LSMK}^k(\mathcal{X}, \mathcal{Y}). \tag{17}$$

To obtain global representation, we concatenate each set of locally pooled features into a global vector

$$\mathbf{V}_x = [\mathbf{v}_x^1, \cdots, \mathbf{v}_x^K], \quad \mathbf{V}_y = [\mathbf{v}_y^1, \cdots, \mathbf{v}_y^K]. \tag{18}$$

It is straightforward to obtain

$$\begin{aligned} \langle \mathbf{V}_x, \mathbf{V}_y \rangle &= \sum_{k=1}^{K} \langle \mathbf{v}_x^k, \mathbf{v}_y^k \rangle = \sum_{k=1}^{K} \text{LSMK}^k(\mathcal{X}, \mathcal{Y}) \\ &= \text{LSMK}(\mathcal{X}, \mathcal{Y}). \end{aligned} \tag{19}$$

Note that, if we replace $\mathbf{v}_x^k = \sum_{\mathbf{x}\in\mathcal{X}} \mathbb{1}(i(\mathbf{x}) = k)\mathbf{x}$ in Eq. (16) by $\mathbf{v}_x^k = \sum_{\mathbf{x}\in\mathcal{X}} \mathbb{1}(i(\mathbf{x}) = k)(\mathbf{x} - \mathbf{c}_k)$, the aggregated global feature $\mathbf{V}_x$ in Eq. (18) will be equivalent to the vector of locally aggregated descriptors (VLAD) [31]. From above analysis, we conclude that VLAD embeds the LSMK kernel.

V. Harmonized Bilinear Pooling

Below we investigate the components unbalance problem inherited in bilinear-pooled features. Following the previous definition, we obtain the bilinear-pooled feature of $\mathcal{A}$ and $\mathcal{B}$ by $\mathbf{F}_A = \sum_{\mathbf{x}\in\mathcal{X}} \mathbf{x}\mathbf{x}^\top$ and $\mathbf{F}_B = \sum_{\mathbf{y}\in\mathcal{Y}} \mathbf{y}_j \mathbf{y}_j{}^\top$. By singular value decomposition (SVD), we obtain

$$\begin{aligned} \mathbf{F}_A &= \mathbf{U}_A \Sigma_A \mathbf{U}_A^\top = \sum_{s=1}^{d} \sigma_A^s \mathbf{u}_A^s \mathbf{u}_A^{s\top} \\ \mathbf{F}_B &= \mathbf{U}_B \Sigma_B \mathbf{U}_B^\top = \sum_{t=1}^{d} \sigma_B^t \mathbf{u}_B^t \mathbf{u}_B^{t\top} \end{aligned} \tag{20}$$

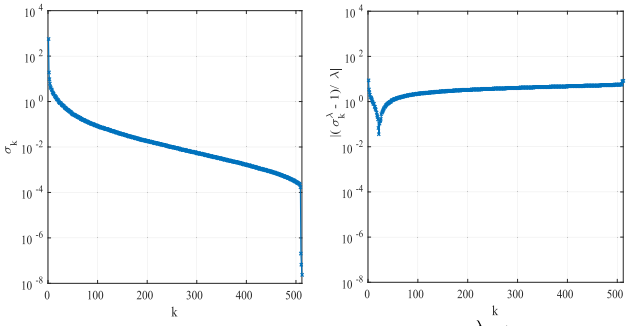
where $\{\mathbf{u}_A^s\}_{s=1}^d$ are singular vectors and $\{\sigma_A^t\}_{t=1}^d$ are singular values. The similarity between $\mathcal{A}$ and $\mathcal{B}$ is

$$\begin{aligned} sim(\mathcal{A}, \mathcal{B}) &= \langle vec(\mathbf{F}_A), vec(\mathbf{F}_B) \rangle \\ &= \sum_{s=1}^{d} \sum_{t=1}^{d} \sigma_A^s \sigma_B^t \langle \mathbf{u}_A^s, \mathbf{u}_B^t \rangle^2. \end{aligned} \tag{21}$$

For the convenience of illustration, we term a singular vector as a component. From Eq. (21), we observe that the similarity between $\mathcal{A}$ and $\mathcal{B}$ is equal to the weighted summation of squares of similarities of all pairs of components $(\mathbf{u}_A^s, \mathbf{u}_B^t)$. The weight corresponds to the product of the corresponding singular values $\sigma_A^s \sigma_B^t$. Nevertheless, the scales of singular values vary significantly. As illustrated in Figure 4(a), the largest singular value is above 10^2 whereas the smallest singular value is below 10^{-4}. Therefore, the weight $\sigma_A^s \sigma_B^t$ from two large singular values will be much larger than that from two small singular values, leading to vanishing of contributions from singular vectors corresponding to small singular values. This problem motivates us to conduct equalization on different singular values to make every component contribute to the final scores between two 3D objects in a more democratic manner.

To better understand the motivation of the proposed equalization method, let us consider an object represented by a feature $\mathbf{x}$ which can be decomposed into a linear combination of two components $10\mathbf{c}_1 + \mathbf{c}_2$ where $\mathbf{c}_1$ and $\mathbf{c}_2$ are orthogonal and unit-norm. Given another object represented by the feature $\mathbf{y} = a_1\mathbf{c}_1 + a_2\mathbf{c}_2$, the similarity between $\mathbf{x}$ and $\mathbf{y}$ is

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle 10\mathbf{c}_1 + \mathbf{c}_2, a_1\mathbf{c}_1 + a_2\mathbf{c}_2 \rangle = 10a_1 + a_2. \tag{22}$$



(a) σ_k (b) $|\frac{\sigma_k^{\lambda}-1}{\lambda}|$

Fig. 4. After Box-Cox transformation, the scales of components are more balanced. Before the transformation, The singular value $\sigma \in [10^{-8}, 10^{3}]$. After transformation, the singular values $\sigma \in [-10, 10]$.

In this case, the similarity between $\mathbf{x}$ and $\mathbf{y}$ is mainly determined by a_1 and the contribution from a_2 is marginal. Our method balances the contributions from a_1 and a_2 and thus makes the similarity more effective.

The Box-Cox transformation [15] is a data stabilization technique widely used in statistics and economics. Inspired by its success in stabilizing variance, we utilize it to normalize the singular values to mitigate the problem caused by burstiness. The Box-Cox transformation is defined as

$$\sigma^{(\lambda)} = \begin{cases} \dfrac{\sigma^{\lambda}-1}{\lambda}, & \lambda \neq 0, \\ ln(\sigma), & \lambda = 0. \end{cases} \tag{23}$$

The condition when $\lambda = 0$ is based on the fact that $\lim_{\lambda \to 0} \frac{\sigma^{\lambda}-1}{\lambda} = ln(\sigma)$. Note that the singular value σ is non-negative since the bilinear matrix is semi-definite. We visualize the values of singular values before and after Box-Cox transformation in Figure 4, in which we set $\lambda = 0.1$. As we can see, after the Box-Cox transformation, the singular values will be transformed to be at comparable scales. Nevertheless, how to choose λ is a nontrivial problem and there is no evidence showing that we should assign the same λ to different singular values $\{\sigma_k\}_{k=1}^{d}$. Therefore, we propose to learn a λ_k for each σ_k from the data. To be specific, we incorporate $\{\lambda_k\}_{k=1}^{d}$ as the weights of one layer of the neural network, which can be trained in an end-to-end manner. The readers can refer to next section for details. Below we summarize the pipeline of the harmonized bilinear pooling in the forward path:

1) $\sum_{i=1}^{N} \mathbf{x}^i \mathbf{x}^{i\top} \rightarrow \mathbf{F}$
2) $\mathbf{F} \rightarrow \sum_{k=1}^{d} \sigma_k \mathbf{u}^k \mathbf{u}^{k\top}$
3) $\sum_{k=1}^{d} \frac{\sigma_k^{\lambda_k}-1}{\lambda_k} \mathbf{u}^k \mathbf{u}^{k\top} \rightarrow \mathbf{H}$

A. Relation to Existing Bilinear Pooling Methods

There are some existing works [32], [33] utilizing the matrix-logarithm operation to map the covariance matrix from Symmetric Postive Definite (SPD) manifold to the tangent Euclidean space. Interestingly, the matrix-logarithm operation is equivalent to Box-Cox transformation conducted on the singular values when $\lambda = 0$. But Meanwhile, Lin *et al.* [34] recently propose the improved bilinear pooling which normalizes the singular values by element-wise signed square-root normalization given by $sign(\sigma_k)\sqrt{|\sigma_k|}$. It is not difficult to observe that the square root normalization used in [34] corresponds to the Cox-Box transformation when $\lambda = 1/2$. Different from matrix-logarithm [32], [33] and improved bilinear pooling [34], we adaptively learn the $\{\lambda_k\}_{k=1}^{d}$ from the data, possessing higher flexibility and modelling ability. Meanwhile, several works [35]–[37] show that higher-order pooling achieves better performance than bilinear pooling. If we use higher-order SVD [38], our harmonized normalization can also be applied in higher-order pooling. Nevertheless, conducting higher-order SVD is very time-consuming, limiting the efficiency.

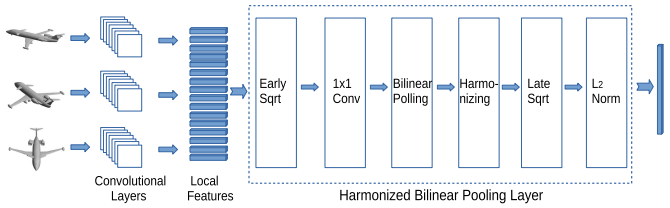


Fig. 5. The architecture of the proposed Multi-view Harmonized Bilinear Network (MHBN).

B. Relation to Normalization Method in VLAD

As for VLAD aggregation method, we also need normalization to balance the contribution from different components. One of most widely used normalization method is intra-normalization proposed in [39]. To be specific, given a VLAD feature $\mathbf{V}_x = [\mathbf{v}_x^1, \cdots, \mathbf{v}_x^K]$, intra-normalization obtains the normalized feature $\bar{\mathbf{V}}_x$ by conducting ℓ_2-normalization on each locally-pooled feature $\mathbf{v}_x^k$ individually:

$$\bar{\mathbf{V}}_x = [\frac{\mathbf{v}_x^1}{\|\mathbf{v}_x^1\|_2}, \cdots, \frac{\mathbf{v}_x^K}{\|\mathbf{v}_x^K\|_2}]. \tag{24}$$

Despite the intra-normalization used in VLAD is quite different from the harmonizing operation proposed for bilinear pooling from the perspective of algorithms, their purposes are quite similar, that is, to make different components inherited in the global feature more balanced.

VI. Multi-View Harmonized Bilinear Network

In this section, we will introduce how to incorporate harmonized bilinear-pooling as a layer of the proposed multi-view harmonized bilinear network and derive the back-propagation equations for training the network. As shown in Figure 5, the proposed harmonized bilinear-pooling layer is concatenated after the last convolutional layer. All views share the weights of the convolutional layers. For each view V_j, the output of the last convolutional layer is a three dimensional tensor $\mathbf{X}_j \in \mathbb{R}^{W \times H \times D}$. We define a local convolutional feature $\mathbf{x} \in \mathbb{R}^{D}$ as a super-column of tensor generated from a specific view. Therefore, given a 3D object represented by M views, we will obtain $N = MWH$ local features. We define $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ as

the set of all local features of all views. We define L as the cross-entropy loss used in training the network. The proposed harmonized bilinear-pooling layer consists of 6 sub-layers:

A. Early Sqrt Sub-Layer

It simply normalizes each local feature by root normalization:

$$\bar{\mathbf{x}}_i = sign(\mathbf{x}_i) \odot \sqrt{|\mathbf{x}_i|}, \tag{25}$$

where $\odot$ represents the element-wise multiplication. $\sqrt{\cdot}$ and $|\cdot|$ represent the element-wise square root and absolute. This layer is actually conducting root-normalization. Previous works like RootSIFT [40] have shown that the root-normalization can considerably improve the performance of local features. We can compute $\frac{\partial L}{\partial \mathbf{x}_i}$ through back-propagation by

$$\frac{\partial L}{\partial \mathbf{x}_i} = \frac{1}{2\sqrt{|\mathbf{x}_i|}} \odot \frac{\partial L}{\partial \bar{\mathbf{x}}_i}. \tag{26}$$

B. Conv Sub-Layer

It is a convolutional layer with $1 \times 1 \times d \times D$ kernel size. It reduces the dimension of local convolutional features from D to d ($d < D$) by

$$\hat{\mathbf{x}}_i = \mathbf{W}\bar{\mathbf{x}}_i + \mathbf{b}, \tag{27}$$

where $\mathbf{W}$ and $\mathbf{b}$ are initialized by PCA. The dimension reduction serves two purposes: 1) improving the efficiency in training and testing; 2) mitigating over-fitting.

C. Bilinear Pooling Sub-Layer

$\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n] \in \mathbb{R}^{d \times N}$ are defined as all the compact local convolutional features after conv sub-layer. The output of the bilinear pooling sub-layer is computed by

$$\mathbf{F} = \hat{\mathbf{X}}\hat{\mathbf{X}}^\top. \tag{28}$$

We can compute $\frac{\partial L}{\partial \hat{\mathbf{X}}}$ through back-propagation by

$$\frac{\partial L}{\partial \hat{\mathbf{X}}} = (\frac{\partial L}{\partial \mathbf{F}} + \frac{\partial L}{\partial \mathbf{F}}^\top)\hat{\mathbf{X}}. \tag{29}$$

D. Harmonizing Sub-Layer

It decomposes the bilinear pooled matrix $\mathbf{F}$ by SVD:

$$\mathbf{F} \rightarrow \mathbf{U}\Sigma\mathbf{U}^\top. \tag{30}$$

The the output of harmonizing layer $\mathbf{H}$ is obtained by

$$\mathbf{H} \leftarrow \mathbf{U}h(\Sigma)\mathbf{U}^\top. \tag{31}$$

$h(\Sigma)$ is the matrix containing the harmonized singular values defined by

$$h(\Sigma)(i,j) = \begin{cases} \dfrac{\sigma_i^{\lambda_i} - 1}{\lambda_i}, & i = j, \\ 0, & i \neq j. \end{cases} \tag{32}$$

where $\{\sigma_k\}_{k=1}^d$ are the singular values of input bilinear-pooled matrix $\mathbf{F}$. The singular values are harmonized by coefficients $\{\lambda_k\}_{k=1}^d$, which are parameters of the harmonizing sub-layer. In the back-propagation phase, $\frac{\partial L}{\partial \lambda_k}$ is computed by

$$\frac{\partial L}{\partial \lambda_k} = \frac{\lambda_k \sigma_k^{\lambda_k} ln(\sigma_k) - \sigma_k^{\lambda_k} + 1}{\lambda_k^2} \mathbf{u}_k{}^\top \frac{\partial L}{\partial \mathbf{H}} \mathbf{u}_k, \tag{33}$$

Below we derive Eq. (33). By plugging Eq. (32) into Eq. (31), we obtain

$$\mathbf{H} = \sum_{k=1}^{d} \frac{\sigma_k^{\lambda_k} - 1}{\lambda_k} \mathbf{u}_k \mathbf{u}_k^\top. \tag{34}$$

Therefore,

$$\begin{aligned} \frac{\partial L}{\partial \lambda_k} &= vec(\frac{\partial L}{\partial \mathbf{H}})^\top vec(\frac{\partial \mathbf{H}}{\partial \lambda_k}) \\ &= \frac{\lambda_k \sigma_k^{\lambda_k} ln(\lambda_k) - \sigma_k^{\lambda_k} + 1}{\lambda_k^2} \mathbf{u}_k{}^\top \frac{\partial L}{\partial \mathbf{H}} \mathbf{u}_k. \end{aligned} \tag{35}$$

Meanwhile, $\frac{\partial L}{\partial \mathbf{F}}$ is computed by

$$\frac{\partial L}{\partial \mathbf{F}} = \mathbf{U}\Big\{\Big(\mathbf{K} \odot \Big(\mathbf{U}^\top \frac{\partial L}{\partial \mathbf{U}}\Big)\Big) + \Big(\frac{\partial L}{\partial \Sigma}\Big)_{diag}\Big\}\mathbf{U}^\top, \tag{36}$$

where $\mathbf{u}_k$ is the k-th singular vector and matrix $\mathbf{K}$ is defined as

$$\mathbf{K}(i,j) = \begin{cases} \dfrac{1}{\sigma_j - \sigma_i}, & i \neq j, \\ 0, & i = j. \end{cases} \tag{37}$$

$\frac{\partial L}{\partial \Sigma}$ and $\frac{\partial L}{\partial \mathbf{U}}$ are computed by

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{U}} &= \Big\{\frac{\partial L}{\partial \mathbf{H}} + \Big(\frac{\partial L}{\partial \mathbf{H}}\Big)^\top\Big\}\mathbf{U}h(\Sigma), \\ \frac{\partial L}{\partial \Sigma} &= h'(\Sigma)\mathbf{U}^\top \frac{\partial L}{\partial \mathbf{H}}\mathbf{U}. \end{aligned} \tag{38}$$

$h(\Sigma)$ is defined in Eq. (32) and $h'(\Sigma)$ is given by

$$h'(\Sigma)(i,j) = \begin{cases} \sigma_i^{\lambda_i - 1}, & i = j, \\ 0, & i \neq j. \end{cases} \tag{39}$$

The readers can refer to [41] for detailed derivation of Eq. (36) (37) (38).

E. Late Sqrt Sub-Layer

It aims to further suppress the burstiness and reshape the 2D matrix into a 1D vector:

$$\mathbf{v} = sign(vec(\mathbf{H})) \odot \sqrt{vec(|\mathbf{H}|)}. \tag{40}$$

F. ℓ_2-Norm Sub-Layer

It conducts the ℓ_2-normalization:

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|_2}. \tag{41}$$

VII. Multi-View VLAD Network

In the previous section, we have analyzed that the VLAD implicitly embeds LSMK, which automatically decouples the irrelevant patches and only match between relevant patches. Straightforwardly, we can directly utilize VLAD to aggregate the local convolutional features of pre-trained CNNs. Nevertheless, the off-the-shelf pre-trained network might not be optimal for new datasets. Inspired by the success of NetVLAD [42], we also incorporate the VLAD as a layer into the neural network to further boost the performance. We propose our multi-view VLAD Network (MVLADN) to achieve end-to-end training for 3D object recognition task. Our experiments show the representation learned from the end-to-end trained network can outperform the aggregated VLAD representation based on the pre-trained CNN model.

Following the definition in the last section, for each view V_i, the output of a convolutional layer is a three-dimension tensor $\mathbf{X}_i^{H\times W\times D}$. We define local convolutional feature $\mathbf{x}\in\mathbb{R}^D$ as a super-column of the tensor. The tensor is further unfolded into $N = nWH$ local convolutional features $\{\mathbf{x}_j\}_{j=1}^N$. As shown in Fig. 6, VLAD layer consists of 4 sub-layers:

A. Square Root Sub-Layer

The first sub-layer is a square root layer, it conducts the root normalization for each local convolutional feature $\mathbf{x}$ by

$$\bar{\mathbf{x}}_j = sign(\mathbf{x}_j)\odot\sqrt{|\mathbf{x}_j|}, \tag{42}$$

where $\odot$ denotes the element-wise product operation and $||$ denotes the element-wise absolute operation. In the back-propagation, we can compute $\frac{\partial L}{\partial \mathbf{x}_i}$ by

$$\frac{\partial L}{\partial \mathbf{x}_j} = \frac{1}{2\sqrt{|\mathbf{x}_j|}}\odot\frac{\partial L}{\partial \bar{\mathbf{x}}_j} \tag{43}$$

B. Conv Sub-Layer

Similar to the one used in MHBN, it is also a $1\times 1\times d\times D$ convolutional layer for dimension reduction. It reduces the dimension of local features from D to d where $d < D$ by

$$\hat{\mathbf{x}}_j = \mathbf{W}\bar{\mathbf{x}}_j + \mathbf{b}. \tag{44}$$

It improves the efficiency and meanwhile smooth the over-fitting. $\mathbf{W}$ and $\mathbf{b}$ are initialized by PCA.

C. Soft VLAD Sub-Layer

The soft-VLAD is an extension of original VLAD by changing original hard cluster assignment to soft assignment to make it differentiable [42]. It computes each sub-vector $\mathbf{v}_k$ by

$$\mathbf{v}_k = \sum_{j=1}^{N}\frac{e^{-\alpha\|\hat{\mathbf{x}}_j-\mu_k\|_2^2}}{\sum_{k'=1}^{K}e^{-\alpha\|\hat{\mathbf{x}}_j-\mu_k\|_2^2}}(\hat{\mathbf{x}}_j-\mu_k). \tag{45}$$

To make the gradients easier to compute, a trick is to replace the similarity measured by Euclidean distance $-\|\hat{\mathbf{x}}_j-\mu_k\|_2^2$ by the inter-product similarity $\hat{\mathbf{x}}_j^\top\mu_k$. Therefore, we replace $e^{-\alpha\|\hat{\mathbf{x}}_j-\mu_k\|_2^2}$ by $e^{\alpha\hat{\mathbf{x}}_j\mathbf{w}_k^\top}$ where $\mathbf{w}_k=\mu_k$ and above equation will be equivalent to

$$\mathbf{v}_k = \sum_{j=1}^{N}\frac{e^{\alpha\hat{\mathbf{x}}_j^\top\mathbf{w}_k}}{\sum_{k'=1}^{K}e^{\alpha\hat{\mathbf{x}}_j^\top\mathbf{w}_{k'}}}(\hat{\mathbf{x}}_j-\mu_k). \tag{46}$$

All the sub-vectors are concatenated into a global vector $\mathbf{v} = [\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_K]$. To enable more flexibility, we decouple $\mathbf{w}_k$ from μ_k and thus $\{\mathbf{w}_k\}_{k=1}^K$ as well as $\{\mu_k\}_{k=1}^K$ consist of the weights of the Soft-VLAD layer.

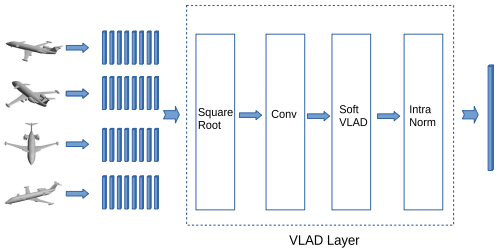


Fig. 6. The structure of VLAD layer.

D. Intra-Normalization [39] Sub-Layer

It conducts the ℓ_2-normalization on each subvector $\mathbf{v}_k$ in order to balance the contribution of each subvector:

$$\hat{\mathbf{v}} = [\mathbf{v}_1/\|\mathbf{v}_1\|_2;\mathbf{v}_2/\|\mathbf{v}_2\|_2;\ldots;\mathbf{v}_K/\|\mathbf{v}_K\|_2] \tag{47}$$

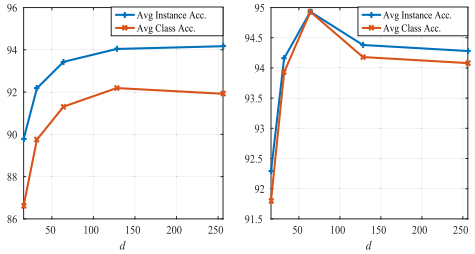
VIII. Experiment

A. Implementation Details

We render the 3D mesh models by placing 6 centroid pointing virtual cameras around the mesh every 60 degrees with an elevation of 30 degrees from the ground plane. We adopt VGG-M network [43] as our base model. Despite that some more advanced networks are proposed such as ResNet [18] and DenseNet [44], we use VGG-M to make a fair comparison with other existing methods which are mostly based on VGG-M. We remove all the layers of original VGG-M after conv5 layer and concatenate the proposed harmonized bilinear pooling/VLAD layer after conv5. We initialize the weights of harmonizing sub-layer $\{\lambda_k\}_{k=1}^d$ by 0.5, the weight of VLAD layer through k-means, the weights of $1\times 1\times d\times D$ conv sub-layer by PCA. A dropout layer is added after harmonized bilinear-pooling/VLAD layer and we set the dropout ratio as 0.5. Likewise MVCNN [2] and RCPCNN [12], after training, we replace the softmax classifier with a linear SVM as the classifier in the testing phase.

B. Datasets and Evaluation Metrics

ModelNet40 [1] consists of 12311 3D models from 40 categories. The models are split into 9843 training samples and 2468 testing samples. ModelNet10 [1] consists of 4899 3D models split into 3991 training samples and 908 testing samples from 10 categories. Following MVCNN-MultiRes [4], we report both average instance accuracy and average class accuracy. Average instance accuracy counts the percentage of



(a) ModelNet40 (b) ModelNet10

Fig. 7. Influence of local convolutional feature dimension d.

TABLE I
INFLUENCE OF THE NUMBER OF VIEWS

Method	3 vews	6 views	12 views
MVCNN [2]	91.33	92.01	91.49
RCPCNN [12]	92.10	92.22	92.18
MHBN(ours)	**93.78**	**94.12**	**93.42**
MVLADN(ours)	**93.48**	**93.96**	**93.02**

TABLE II
INFLUENCE OF EARLY SQRT AND LATE SQRT SUBLAYERS

early sqrt			✓
late sqrt		✓	✓
ModelNet40 class ac.	90.00	91.09	**92.23**
ModelNet40 instance ac.	93.24	93.63	**94.12**
ModelNet10 class ac.	93.03	93.75	**94.91**
ModelNet10 instance ac.	93.39	93.94	**94.93**

TABLE III
COMPARISON WITH OTHER POOLING METHODS

Method	ModelNet10			ModelNet40		
	Class	Inst.	mAP	Class	Inst.	mAP
Sum pooling	90.2	91.2	85.2	85.2	88.0	84.2
Max pooling	91.0	91.4	85.7	85.7	87.4	84.3
DeepSets [45]	91.5	91.7	85.9	86.3	88.5	85.0
Bilinear [24]	88.7	89.8	84.4	84.8	87.0	83.8
IBP [34]	93.1	93.2	88.4	91.2	93.2	87.2
Log-cov [32]	93.4	93.5	88.7	90.6	93.0	87.4
MHBP (ours)	**94.9**	**94.9**	**90.6**	**92.2**	**94.1**	**88.9**

the correctly recognized testing samples among all the testing samples whereas the average class accuracy is the average accuracy cross all the classes.

C. Influence of the Number of Views

We evaluate the effect of the number of views on the average instance accuracy of our MHBN and MVLADN on the Modelnet40 dataset. We compare them with that from MVCNN [2] and recurrent clustering and pooling (RCPCNN) [12]. The accuracies of MVCNN and RCPCNN shown in Table I are taken from Table 3 of Wang *et al.* [12]. As shown in Table I, our MHBN and MVLADN consistently outperform MVCNN and RCPCNN with a large margin. Note that a performance drop is observed when the number of views increases from 6 to 12 in MVCNN, RCPCNN and ours. This performance drop might be attributed to the fact that sampling too densely will enlarge the joint area of two adjacent views, making the representation dominated by the overlaps.

D. Ablation Study on MHBN

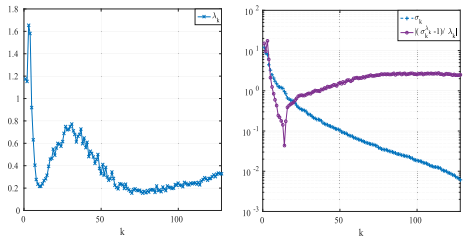
1) Influence of Dimension of Local Features: The conv sub-layer aims to reduce the dimension of original local convolutional features from D (512) to d. We evaluate the influence of d on the performance of our MHBN. As shown in Figure 7, on the ModelNet40 dataset, the average instance/class accuracy generally improves as the dimension of local features d increases. To balance the efficiency and effectiveness, we set $d = 128$ on the ModelNet40 dataset. In contrast, on the ModelNet10 dataset, the accuracy drops when $d > 64$. This is owing to that the scale of ModelNet10 is small and thus a larger d tends to cause over-fitting. We set $d = 64$ on the ModelNet10 dataset.

2) Influence of Early Sqrt and Late Sqrt Sub-Layers: By removing these two sub-layers, MHBN achieves 93.24 average instance accuracy on ModelNet40 and 93.39 on ModelNet10. After we add on late sqrt sub-layer, it improves average instance accuracy from 93.24 to 93.63 on ModelNet40. If we add on early sqrt and late sqrt sub-layers together, the average class/instance accuracy is improved from 90.00/93.24 to 92.19/94.04 on ModelNet40 and from 93.03/93.39 to 94.93/94.93 on ModelNet10.

3) Comparison With Other Pooling Methods: In this section, we compare the classification and retrieval performance of the proposed harmonized bilinear-pooling with sum-pooling, max-pooling, bilinear pooling [24], improved bilinear pooling [34], log-covariance pooling [32] and an off-the-shelf set-wise pooling method DeepSets [45]. In implementation of DeepSets [45], we use a stack of two layers define in Eq. (6) of [45] to process the last convolutional feature map and the global feature is obtained by sum-pooling. We use mean average precision (mAP) to evaluate the retrieval accuracy. Note that, bilinear pooling corresponds to the special case of the proposed harmonized bilinear pooling by fixing $\lambda = 1$, improved bilinear pooling is the special case when $\lambda = 1/2$ and log-covariance pooling is the condition when $\lambda = 0$. To make a fair comparison, we also add early sqrt and late sqrt layers in bilinear pooling, improved bilinear pooling and log-covariance pooling. As shown in Table III, ours consistently outperforms all other pooling methods on both ModelNet10 and ModelNet40 datasets. Meanwhile, the classification performance of compared methods are in accordance with their retrieval performance, that is, the method achieving higher classification accuracy tends to have a higher retrieval mAP. In Figure 8, we visualize $\{\lambda_k\}_{k=1}^{d}$ and components' scales before and after harmonization on the ModelNet10 dataset.

E. Ablation Study on MVLADN

1) Influence of the Dimension of Local Features: As mentioned in the previous section, the second sub-layer in the VLAD layer is a $1 \times 1 \times D \times d$ convolutional layer. It aims to reduce the dimension of local features from D(512) to d. We test the performance of the proposed MVLADN when



(a) λ_k (b) σ_k

Fig. 8. The visualization of λ_k, σ_k and $|(\sigma_k^{\lambda_k}-1)/\lambda_k|$.

TABLE IV

THE INFLUENCE OF SQUARE ROOT LAYER AND INTRA-NORM LAYER ON THE PERFORMANCE OF THE PROPOSED MVLADN

Square-root ?		✓		✓
Intra-norm ?			✓	✓
ModelNet40 class ac.	89.84	89.85	90.16	**91.68**
ModelNet40 instance ac.	91.77	92.38	93.15	**93.96**
ModelNet10 class ac.	91.98	92.55	93.67	**94.93**
ModelNet10 instance ac.	92.40	92.73	93.72	**94.93**

the reduced dimension d varies among 32, 64, 128 and 256. As we can see from Figure 9, the performance of the pre-trained MVLADN generally increases as the feature dimension increases. Meanwhile, the finetuned MVLADN achieved the best performance when $d = 128$. We set the default value of $d = 128$ on both datasets.

2) Influence of the Number of Clusters: The dimension of the output of the VLAD layer is dK, where d is the dimension of local features and K is the number of clusters. We evaluate the influence of the number of clusters on the performance of the proposed MVLADN in Figure 10. As shown in Figure 10, the proposed MVLADN achieves excellent performance when $K = 64/128$. We set default $K = 64$ and thus the default dimension of the output of VLAD layer is $64 \times 128 = 8192$.

3) Square-Root Sub-Layer and Intra-Norm Sub-Layer: We evaluate the influence of square-root sub-layer and the intra-norm sub-layer. As we can see from Table IV that, square-root sublayer and intra-norm sub-layer indeed boost the performance of the proposed MVLADN. For instance, by removing both square-root and intra-norm sub-layers, the average instance/class accuracy achieved on the ModelNet40 dataset is only 91.77/89.84. By incorporating intra-norm sub-layer, our MVLADN achieves a 93.15/90.16 average instance/class accuracy. Moreover, by incorporating both square-root and intra-norm sub-layers, our MVLADN achieves an 93.96/91.68 average instance/class accuracy on the ModelNet40 dataset.

F. Efficiency

To demonstrate the efficiency of the proposed MHBN, we compare the time cost with MVCNN [2]. We set the batch size as 16 and test on a single P40 Nvidia GPU card. Table V reports the total time cost on the whole neural network and the time cost on pooling per batch. As shown, despite the pooling time having increased considerably, the total time cost only increases marginally, since the cost of pooling layers is quite marginal compared to that of convolutional layers.

TABLE V

TIME COST COMPARISON BETWEEN MVCNN [2] AND OUR MHBN

	Pooling Time	Total Time
MVCNN [2]	0.2 ms	150 ms
MHBN (ours)	3 ms	155 ms

TABLE VI

THE INFLUENCE OF ADDITIONAL MODALITY. BY ADDING ADDITIONAL DEPTH MODALITY, THE PERFORMANCE GETS IMPROVED

RGB	✓		✓
Depth		✓	✓
ModelNet40	94.12/92.23	93.52/91.16	94.73/93.06
ModelNet10	94.93/94.91	94.27/93.93	95.04/95.03

TABLE VII

THE INFLUENCE OF ADDITIONAL MODALITY ON MVLADN. BY ADDING ADDITIONAL DEPTH MODALITY, THE PERFORMANCE GETS IMPROVED

RGB	✓		✓
Depth		✓	✓
ModelNet40	93.96/91.68	92.82/90.05	94.57/92.34
ModelNet10	94.93/94.93	94.05/93.73	94.71/94.53

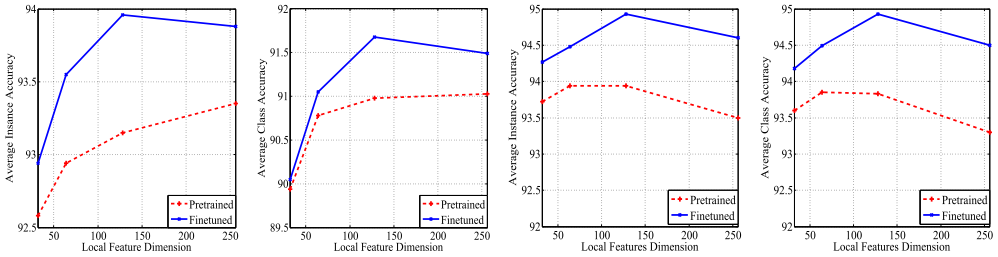
G. Additional Modality

Note that, all the experiments in the previous sections are conducted using only RGB images. Analogous to [5], [12], we evaluate the influence of additional depth modality as well. For each 3D object, we obtain 6 depth views in addition to 6 RGB views. We train two MHBNs/MVLADNs for RGB views and depth views separately. We concatenate the features from RGB-MHBN/MVLADN and that of Depth-MHBN/MVLADN as the final feature of a 3D object, and train a linear SVM as classifier, respectively. As shown in Table VI and VII, incorporating the additional depth modality generally improves recognition performance of MHBN and MVLADN. For instance, using MHBN, incorporating additional depth modality improves the performance from 94.12/92.23 to 94.73/93.06 on the ModelNet40 dataset and improve that from 94.93/94.91 to 95.04/95.03 on the ModelNet10 dataset. Nevertheless, we witness a slight performance drop using MVLADN on ModelNet10 dataset when adding depth modality.

H. Comparison With State-of-the-Art Methods

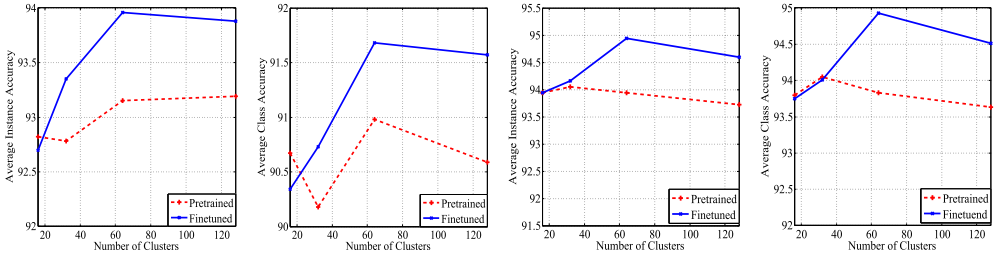
In this section, we compare ours with existing state-of-the-art methods. Sequentially, we compare ours with three categories of methods, that is, volume-based methods, view-based methods and pointset-based methods.

We first compare with the volume-based methods [1], [3], [4], [46]. As shown in Table VIII, the volume-based methods are generally not as good as ours and other view-based methods. As to view-based methods, MVCNN [2] achieves a 92.1/89.9 average instance/class accuracy using



(a) ModelNet40 AIA (b) ModelNet40 ACA (c) ModelNet10 AIA (d) ModelNet10 ACA

Fig. 9. The influence of d, local feature dimension, on the recognition performance of the proposed MVLADN.



(a) ModelNet40 AIA (b) ModelNet40 ACA (c) ModelNet10 AIA (d) ModelNet10 ACA

Fig. 10. The influence of K, the number of clusters, on the recognition performance of the proposed MVLADN.

TABLE VIII

COMPARISON WITH STATE-OF-THE-ART METHODS

Method	Views #	Modality	ModelNet40		ModelNet10	
			Instance	Class	Instance	Class
3DShapeNets [1]	-	Volume	-	77.3	-	83.5
VoxNet [3]	-	Volume	-	83.0	-	92.0
3DGAN [46]	-	Volume	-	83.3	-	91.0
Subvolume [4]	-	Volume	89.2	86.0	-	-
3D DescriptorNet [19]	-	Volume	-	-	92.4	-
DeepPano [47]	-	RGB	-	77.6	-	88.5
GIFT [6], [14]	64	RGB	-	89.5	-	91.5
MVCNN [2]	12	RGB	92.1	89.9	-	-
MVCNN-MultiRes [4]	20	3-Resolution RGB	93.8	91.4	-	-
Pairwise [5]	12	RGB + Dep	-	91.1	-	93.2
FusionNet [48]	20	RGB + Volume	-	90.8	-	93.1
RCPCNN [12]	12	RGB	92.2	-	-	-
GVCNN [17]	8	RGB	93.1	-	-	-
MLH-MV [49]	3	MLH	93.1	-	94.8	-
PointNet [9]	-	Points	89.2	86.2	-	-
Kd-Network [11]	-	Points	91.8	88.5	94.0	93.5
PointNet++ [10]	-	Points + Normal	-	91.7	-	-
PointGrid [21]	-	Points	92.0	88.9	-	-
MHBN (Ours)	6	RGB	94.1	92.2	94.9	94.9
MVLADN (Ours)	6	RGB	94.0	91.7	94.9	94.9
MHBN (Ours)	6	RGB + Dep	94.7	**93.1**	95.0	**95.0**
MVLADN (Ours)	6	RGB + Dep	94.6	92.3	94.7	94.5

12 views, which are considerably better than volume-based methods. In contrast, our MHBN achieves a 94.7/93.1 average instance/class accuracy and our MVLADN achieves a 94.6/92.3 average instance/class accuracy using a combination of 6 RGB views and 6 depth views. Several recently proposed pointset-based methods, Kd-Network [11], PointNet++ [10] and PointGrid [21] are compared as well. As shown in Table VIII, our MHBN and MVLADN consistently outperform all of them in both single-modality mode and multi-modality mode.

IX. CONCLUSION

In this paper, we characterize a 3D object by a set of local patches from multiple projected views. 3D object representation learning is tackled from the perspective of set-to-set matching. We exploit two set-to-set matching kernels, PSMK and LSMK, both of which emphasize the pairs consisting of relevant patches and suppress the pairs of irrelevant patches. To further boost the efficiency and obtain global features, we embed the set-to-set matching kernel in local feature aggregation phase. By exploiting the connection between local feature pooling and set-to-set matching kernel, we discover bilinear pooling embeds PSMK and VLAD embeds LSMK. To balance the contribution of different components in aggregated global feature, we propose the harmonized bilinear pooling. For end-to-end training, we use harmonized bilinear pooling and VLAD as two layers to construct MHBN and MVLADN. Systematic experiments conducted on

public benchmark datasets demonstrate the effectiveness of our MHBN and MVLADN.

REFERENCES

[1] Z. Wu *et al.*, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[2] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 945–953.

[3] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 922–928.

[4] C. R. Qi, H. Su, M. NieBner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5648–5656.

[5] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3813–3822.

[6] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. J. Latecki, "GIFT: A real-time and scalable 3D shape search engine," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5023–5032.

[7] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas, "FPNN: Field probing neural networks for 3D data," in *Proc. NIPS*, 2016, pp. 307–315.

[8] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, "Orientation-boosted voxel nets for 3D object recognition," 2016, *arXiv:1604.03351*. [Online]. Available: http://arxiv.org/abs/1604.03351

[9] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[10] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[11] R. Klokov and V. Lempitsky, "Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 863–872.

[12] C. Wang, M. Pelillo, and K. Siddiqi, "Dominant set clustering and pooling for multi-view 3D object recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–12.

[13] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," 2016, *arXiv:1608.04236*. [Online]. Available: http://arxiv.org/abs/1608.04236

[14] S. Bai, X. Bai, Z. Zhou, Z. Zhang, Q. Tian, and L. J. Latecki, "GIFT: Towards scalable 3D shape retrieval," *IEEE Trans. Multimedia*, vol. 19, no. 6, pp. 1257–1271, Jun. 2017.

[15] R. Sakia, "The Box-Cox transformation technique: A review," *Statistician*, vol. 41, pp. 169–178, 1992.

[16] T. Yu, J. Meng, and J. Yuan, "Multi-view harmonized bilinear network for 3D object recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 186–194.

[17] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, "GVCNN: Group-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1345–1353.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[19] J. Xie, Z. Zheng, R. Gao, W. Wang, S.-C. Zhu, and Y. N. Wu, "Learning descriptor networks for 3D shape synthesis and analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8629–8638.

[20] H. Su *et al.*, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2530–2539.

[21] T. Le and Y. Duan, "PointGrid: A deep network for 3D shape understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9204–9214.

[22] J. B. Tenenbaum and W. T. Freeman, "Separating style and content with bilinear models," *Neural Comput.*, vol. 12, no. 6, pp. 1247–1283, 2000.

[23] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, "Semantic segmentation with second-order pooling," in *Proc. ECCV*, 2012, pp. 430–443.

[24] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1449–1457.

[25] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, "Compact bilinear pooling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 317–326.

[26] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie, "Kernel pooling for convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2921–2930.

[27] S. Kong and C. Fowlkes, "Low-rank bilinear pooling for fine-grained classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 365–374.

[28] T. Yu, X. Li, and P. Li, "Fast and compact bilinear pooling by shifted random maclaurin," in *Proc. AAAI*, 2021.

[29] T. Yu, X. Li, and P. Li, "3D object representation learning: A set-to-set matching perspective," in *Proc. AAAI Conf. Artif. Intell.*, 2021.

[30] F. Radenovic, G. Tolias, and O. Chum, "Fine-tuning CNN image retrieval with no human annotation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 7, pp. 1655–1668, Jul. 2019.

[31] H. Jegou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3304–3311.

[32] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Matrix backpropagation for deep networks with structured layers," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2965–2973.

[33] Z. Huang, R. Wang, S. Shan, X. Li, and X. Chen, "Log-Euclidean metric learning on symmetric positive definite manifold with application to image set classification," in *Proc. ICML*, 2015, pp. 720–729.

[34] T.-Y. Lin and S. Maji, "Improved bilinear pooling with CNNs," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–12.

[35] P. Koniusz and A. Cherian, "Sparse coding for third-order super-symmetric tensor descriptors with application to texture recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5395–5403.

[36] A. Cherian, P. Koniusz, and S. Gould, "Higher-order pooling of CNN features via kernel linearization for action recognition," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 130–138.

[37] P. Koniusz, F. Yan, P.-H. Gosselin, and K. Mikolajczyk, "Higher-order occurrence pooling for bags-of-words: Visual concept detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 2, pp. 313–326, Feb. 2017.

[38] G. Bergqvist and E. Larsson, "The higher-order singular value decomposition: Theory and an application [lecture notes]," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 151–154, May 2010.

[39] R. Arandjelovic and A. Zisserman, "All about VLAD," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1578–1585.

[40] R. Arandjelovic and A. Zisserman, "Three things everyone should know to improve object retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2911–2918.

[41] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Training deep networks with structured layers by matrix backpropagation," *CoRR*, vol. abs/1509.07838, 2015. [Online]. Available: http://arxiv.org/abs/1509.07838

[42] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5297–5307.

[43] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. Brit. Mach. Vis. Conf.*, 2014, pp. 1–12.

[44] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[45] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3391–3401.

[46] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Proc. NIPS*, 2016, pp. 82–90.

[47] B. Shi, S. Bai, Z. Zhou, and X. Bai, "DeepPano: Deep panoramic representation for 3-D shape recognition," *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2339–2343, Dec. 2015.

[48] V. Hegde and R. Zadeh, "FusionNet: 3D object classification using multiple data representations," 2016, *arXiv:1607.05695*. [Online]. Available: http://arxiv.org/abs/1607.05695

[49] K. Sarkar, B. Hampiholi, K. Varanasi, and D. Stricker, "Learning 3D shapes as multi-layered height-maps using 2D convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 71–86.