Multi-view Harmonized Bilinear Network for 3D Object Recognition

Tan Yu[1], Jingjing Meng[2], Junsong Yuan[2]
[1] Interdisciplinary Graduate School, Nanyang Technological University
[2] Department of CSE, State University of New York at Buffalo
tyu008@ntu.edu.sg, {jmeng2,jsyuan}@buffalo.edu

Abstract

View-based methods have achieved considerable success in 3D object recognition tasks. Different from existing view-based methods pooling the view-wise features, we tackle this problem from the perspective of patches-to-patches similarity measurement. By exploiting the relationship between polynomial kernel and bilinear pooling, we obtain an effective 3D object representation by aggregating local convolutional features through bilinear pooling. Meanwhile, we harmonize different components inherited in the bilinear feature to obtain a more discriminative representation. To achieve an end-to-end trainable framework, we incorporate the harmonized bilinear pooling as a layer of a network, constituting the proposed Multi-view Harmonized Bilinear Network (MHBN). Systematic experiments conducted on two public benchmark datasets demonstrate the efficacy of the proposed methods in 3D object recognition.

1. Introduction

Inspired by the success of deep learning in 2D images, the community has also attempted to exploit the convolutional neural networks for 3D object recognition [34, 29, 22, 24, 15, 1, 18, 27, 23, 25, 16]. These approaches can be coarsely classfied into three categories according to their input: 1) view-based methods [29, 1, 15, 32], 2) volume-based methods [34, 22, 24, 18, 2, 27] and 3) pointset-based methods [23, 16, 25]. 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 network 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 [2] outperforms existing view-based methods, its excellent performance is mainly attributed to the model ensemble and a more advanced base model.

In spite of that the view-base methods [29, 1, 15, 32]

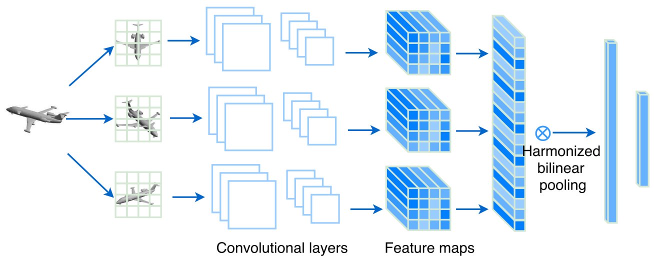


Figure 1. The architecture of the proposed Multi-view Harmonized Bilinear Network (MHBN).

have already achieved good performance in 3D object recognition, there are still some drawbacks existing in current methods. Multi-view CNN (MVCNN) [29] max-pools the view-wise feature into a global feature as the representation of the 3D object. But for each neural node, the max-pooling operation only retains the maximal activation from one specific view whereas the non-maximal elements are ignored, causing the loss of visual information. To enable each view to contribute to the final representation, an alternative solution is to replace max-pooling by sum-pooling. However, the experiments in [29] shows that the performance of sum-pooling is even worse than max-pooling and there is no explanation about the worse performance of sum-pooling. Below we investigate the reason.

We define $\mathcal{A}$ as a 3D object and $\{\mathbf{A}_i\}_{i=1}^n$ as features of n projected views. In the same manner, we define another 3D object $\mathcal{B}$ and $\{\mathbf{B}_j\}_{j=1}^n$. Through sum-pooling, descriptors of two objects are obtained by $\mathbf{A} = \sum_{i=1}^n \mathbf{A}_i$ and $\mathbf{B} = \sum_{j=1}^n \mathbf{B}_j$. We evaluate the discriminative power of sum-pooled features through similarity measurement:

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

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 $(\mathbf{A}_i, \mathbf{B}_j)$. Nevertheless, as shown in Figure 2 (a), only the similarities 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 pairs of corresponding views can be easily contaminated by overnumbered non-corresponding pairs.

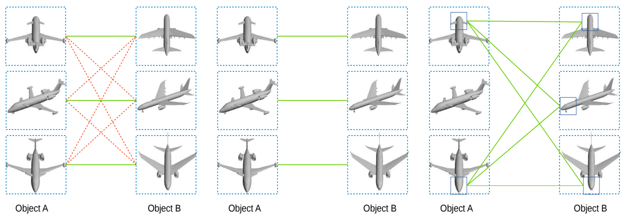
To mitigate the negative influence from pairs of non-corresponding views, GIFT [1] utilizes a robust version of Hausdorff distance defined as

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

As shown in Figure 2 (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 remove the distraction from pairs of non-corresponding views but it also discards the useful information.

In light of the limitations of max-pool MVCNN, sum-pool MVCNN and GIFT, we propose to utilize patch-level features rather than view-level features to obtain a more reasonable similarity measure between two 3D objects. As shown in Figure 2 (c), it has following advantages: 1) it can exploit the useful pairs consisting of relevant patches from two non-corresponding views; 2) even in two corresponding views, it only counts the pairs consisting of relevant patches and decouple the pairs of irrelevant patches. Meanwhile, recent studies [6, 21] show the local convolutional feature from convolutional neural network is a natural and effective representation for the local patch, making it feasible to integrate the patch-based method in the network.

To achieve a reasonable patches-to-patches similarity score, we propose to use polynomial set-to-set similarity, which adaptively assign higher weights to good matching pairs and lower weights to bad ones. However, directly computing polynomial similarity 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. By exploiting the connection between bilinear pooling methods and polynomial kernel, we show the same functionality of polynomial set-to-set similarity can be achieved by bilinear pooling in a more efficient manner. To be specific, the set-to-set similarity computing complexity is reduced from $\mathcal{O}(N^2 d)$ to $\mathcal{O}(d^2)$ (where N is 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



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

Figure 2. sum-pool MVCNN, GIFT and patch-based method.

power, we learn an element-wise Box-Cox [26] transformation for each singular value from the data. 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. Through bilinear pooling local convolutional features, we efficiently achieve the functionality of polynomial set-to-set similarity.
- We harmonize singular values of the pooled bilinear features by learning an element-wise Box-Cox transformation on each singular value and obtain a more discriminative representation for the 3D object.
- We integrate our harmonized bilinear pooling into a layer of a network to construct our multi-view harmonized bilinear network (Figure 1). It can be trained in an end-to-end manner for 3D object recognition.

2. Related Work

View-based methods. MVCNN [29] 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 [1] also extracts view-wise features but do not pool them. Instead, it obtains the similarity between two 3D objects by view-wise matching. More recently, Wang *et al.* [32] recurrently cluster the views into multiple sets, pool the features in each set and achieve better performance than the original MVCNN.

Volume-based methods. Some works [34, 22, 24] 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 [2], 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 [8] models and one Inception model whereas most existing view-based methods are based on a single VGG-M model. As shown in [2], using a single ResNet model, its performance is not as good as the view-based methods.

Pointset-based methods. PointNet proposed by Qi *et al.* [23] directly takes unordered point sets as inputs, addressing the sparsity problem encountered in volume-based methods. In parallel, Klokov *et.al* [16] proposed Kd-Networks for the recognition of 3D object represented by 3D point cloud. Recently, Qi *et al.* [25] improve PointNet by exploiting local structures induced by the metric space.

Bilinear pooling is first proposed by Tenenbaum and Freeman [31] to separate style and content. It was traditionally used in pooling hand-crafted features [3]. Recently it is used in pooling local convolutional features [20, 7, 5, 17] 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 [20, 7, 5, 17] 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 a more discriminative representation of 3D object.

3. From set-to-set similarity to bilinear pooling

In this section, we bridge the gap between set-to-set similarity with bilinear pooling. We define $\mathcal{X}_A = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ as the set of local convolutional features from all the views of 3D object $\mathcal{A}$. Each local convolutional feature represents a local patch from one view of the object $\mathcal{A}$. We define $\mathcal{X}_B = \{\mathbf{y}_1, ..., \mathbf{y}_N\}$ as the local convolutional features from another object $\mathcal{B}$. Straightforwardly, we define two simple set similarity measurement: sum set-to-set similarity (SSS) and maximum set-to-set similarity (MSS). $\text{SSS}(\mathcal{X}_A, \mathcal{X}_B)$ measures the sum of similarities between each point in $\mathcal{X}_A$ and each point in $\mathcal{X}_B$:

$$\text{SSS}(\mathcal{X}_A, \mathcal{X}_B) = \sum_{\mathbf{x}\in\mathcal{X}_A}\sum_{\mathbf{y}\in\mathcal{X}_B}\langle\mathbf{x},\mathbf{y}\rangle. \tag{3}$$

The SSS is robust to the noise since it takes sum of similarities into consideration. Nevertheless, the good matching pairs could be easily swamped by the bad ones. In contrast, $\text{MSS}(\mathcal{X}_A, \mathcal{X}_B)$ measures the maximum similarity between points in $\mathcal{X}_A$ and points in $\mathcal{X}_B$:

$$\text{MSS}(\mathcal{X}_A, \mathcal{X}_B) = \max_{\mathbf{x}\in\mathcal{X}_A}\max_{\mathbf{y}\in\mathcal{X}_B}\langle\mathbf{x},\mathbf{y}\rangle. \tag{4}$$

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

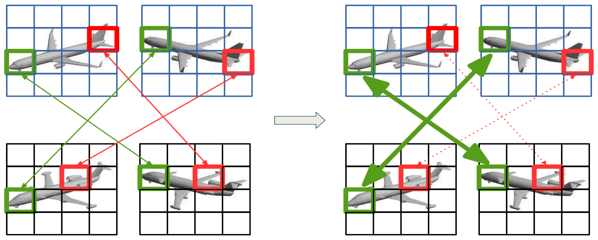


Figure 3. By adaptively assigning different weights, good matching pairs are highlighted whereas the bad ones are suppressed.

To keep the essence and goes the dregs, we propose a more general similarity measurement termed polynomial set-to-set similarity (PSS_p):

$$\text{PSS}_p(\mathcal{X}_A, \mathcal{X}_B) = \Big(\sum_{\mathbf{x}\in\mathcal{X}_A}\sum_{\mathbf{y}\in\mathcal{X}_B}\langle\mathbf{x},\mathbf{y}\rangle^p\Big)^{\frac{1}{p}}. \tag{5}$$

It is not difficult to observe that both SSS and MSS are special cases of PSS_p. To be specific, SSS corresponds to the condition when $p = 1$ whereas MSS 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 the noise. Let us consider a a special case by setting $p = 2$ and remove the $\frac{1}{p}$ entry in the original PSS_p since it does not change the order of the similarities:

$$\text{PSS}_2(\mathcal{X}_A, \mathcal{X}_B) = \sum_{\mathbf{x}\in\mathcal{X}_A}\sum_{\mathbf{y}\in\mathcal{X}_B}\langle\mathbf{x},\mathbf{y}\rangle^2 \tag{6}$$

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 higher weights $w(\mathbf{x},\mathbf{y})$ are assigned to the good matchings with higher value of $\langle\mathbf{x},\mathbf{y}\rangle$ whereas the lower weights are assigned to the bad ones as illustrated in Figure 3. It can be regarded as a soft and robust version of MSS.

In fact, $\langle\mathbf{x},\mathbf{y}\rangle^2$ is a special case of polynomial kernel $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^\top\mathbf{y} + c)^d$ when $c = 0$ and $d = 2$. It possesses explicit feature map $\langle\mathbf{x},\mathbf{y}\rangle^2 = \langle vec(\mathbf{x}\mathbf{x}^\top), vec(\mathbf{y}\mathbf{y}^\top)\rangle$ [28]. Thus we can rewrite Eq. (6) by

$$\text{PSS}_2(\mathcal{X}_A, \mathcal{X}_B) = \langle vec(\sum_{\mathbf{x}\in\mathcal{X}_A}\mathbf{x}\mathbf{x}^\top), vec(\sum_{\mathbf{y}\in\mathcal{X}_B}\mathbf{y}\mathbf{y}^\top)\rangle, \tag{7}$$

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 PSS_2 between two sets of local features are equal to the inner-product similarity between two bilinear pooled features. By exploiting this connection, when computing $\text{PPS}_2(\mathcal{X}_A, \mathcal{X}_B)$, we avoid exhaustive $|\mathcal{X}_A||\mathcal{X}_B|$ times comparisons between every pair of d-dimension local features and just need conduct one time inner product between two d^2-dimension global features. It considerably boosts the efficiency in both computation and memory. Meanwhile, the bilinear-pooled global feature of the 3D object can be used as the input of any classifier such as SVM and softmax whereas the GIFT [1] can only rely on nearest-neighbor classifier due to its dependency to Hausdorff distance.

4. Harmonized bilinear pooling

In the previous section, we have analyzed that bilinear pooling can effectively suppress the bad matching pairs and highlight the good ones. Nevertheless, it ignores the problem caused by components unbalance. The components unbalance problem has been discussed in burstiness phenomenon [14] where repetitive patterns frequently appearing in the image will dominate the image representation. Below we investigate the components unbalance problem inherited in bilinear-pooled features.

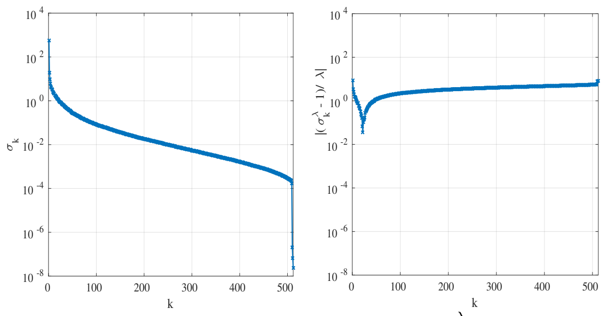
Following the previous definition, we obtain the bilinear-pooled feature of $\mathcal{A}$ and $\mathcal{B}$ by $\mathbf{F}_A = \sum_{\mathbf{x}\in\mathcal{X}_A} \mathbf{x}\mathbf{x}^\top$ and $\mathbf{F}_B = \sum_{\mathbf{y}\in\mathcal{X}_B} \mathbf{y}_j\mathbf{y}_j{}^\top$. Since $\mathbf{F}_A$ and $\mathbf{F}_B$ are symmetric, their left-singular vectors are also the right-singular vectors. By singular value decomposition (SVD), we obtain

$$\begin{aligned}\mathbf{F}_A &= \mathbf{U}_A\Sigma_A\mathbf{U}_A^\top = \sum_{s=1}^{d} \sigma_A^s \mathbf{u}_A^s \mathbf{u}_A^s{}^\top \\ \mathbf{F}_B &= \mathbf{U}_B\Sigma_B\mathbf{U}_B^\top = \sum_{t=1}^{d} \sigma_B^t \mathbf{u}_B^t \mathbf{u}_B^t{}^\top\end{aligned} \tag{8}$$

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

$$\begin{aligned} sim(\mathcal{A},\mathcal{B}) &= \langle vec(\mathbf{F}_A), vec(\mathbf{F}_B)\rangle \\ &= \sum_{s=1}^{d}\sum_{t=1}^{d} \sigma_A^s\sigma_B^t \langle \mathbf{u}_A^s, \mathbf{u}_B^t\rangle^2. \end{aligned} \tag{9}$$

For the convenience of illustration, we term a singular vector as a component. From Eq. (9), 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.



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

Figure 4. After Box-Cox transformation, the scales of components are more balanced.

The Box-Cox transformation [26] 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

$$\sigma^{(\lambda)} = \begin{cases} \frac{\sigma^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ ln(\sigma), & \lambda = 0 \end{cases} \tag{10}$$

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

Relation to existing work There are some existing works [12, 11] 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.* [19] 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 [19] corresponds to the Box-Cox transformation when $\lambda = 1/2$. Different from matrix-logarithm [12, 11] and improved bilinear pooling [19], we adaptively learn the $\{\lambda_k\}_{k=1}^{d}$ from the data, possessing higher flexibility and modelling ability.

5. 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-dimension 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. The proposed harmonized bilinear-pooling layers consists of 6 sub-layers:

- **Early sqrt sub-layer** simply normalizes each local feature by root normalization:

$$\bar{\mathbf{x}}_i = sign(\mathbf{x}_i) \odot \sqrt{|\mathbf{x}_i|}, \tag{11}$$

  where $\odot$ represents the element-wise multiplication. $\sqrt{\cdot}$ and $|\cdot|$ represents the element-wise square root and absolute. We can compute $\frac{\partial L}{\partial \mathbf{x}_i}$ through back-propagation by

$$\frac{\partial L}{\partial \mathbf{x}_i} = \frac{1}{2\sqrt{|\mathbf{x}_i|}} \odot \frac{\partial L}{\partial \bar{\mathbf{x}}_i}. \tag{12}$$

- **Conv sub-layer** 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

$$\hat{\mathbf{x}}_i = \mathbf{W}\bar{\mathbf{x}}_i + \mathbf{b}, \tag{13}$$

  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) reducing the number of parameters to mitigate over-fitting.

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

$$\mathbf{F} = \hat{\mathbf{X}}\hat{\mathbf{X}}^\top. \tag{14}$$

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

$$\frac{\partial L}{\partial \hat{\mathbf{X}}} = (\frac{\partial L}{\partial \mathbf{F}} + \frac{\partial L}{\partial \mathbf{F}}^\top)\hat{\mathbf{X}}. \tag{15}$$

- **Harmonizing sub-layer** first decomposes the bilinear pooled matrix $\mathbf{F}$ by SVD:

$$\mathbf{F} \rightarrow \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top. \tag{16}$$

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

$$\mathbf{H} \leftarrow \mathbf{U}h(\mathbf{\Sigma})\mathbf{U}^\top. \tag{17}$$

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

$$h(\mathbf{\Sigma})(i,j) = \begin{cases} \frac{\sigma_i^{\lambda_i}-1}{\lambda_i}, & i = j \\ 0, & i \neq j \end{cases} \tag{18}$$

  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 the parameters of the harmonizing layer to be trained. In the back-propagation phase, $\frac{\partial L}{\partial \lambda_k}$ and $\frac{\partial L}{\partial \mathbf{F}}$ are computed by

$$\frac{\partial L}{\partial \lambda_k} = \frac{\lambda_k \sigma_k^{\lambda_k} ln(\sigma_k) - \sigma_k^{\lambda_k} + 1}{\lambda_k^2} \mathbf{u}_k{}^\top \frac{\partial L}{\partial \mathbf{H}} \mathbf{u}_k, \tag{19}$$

$$\frac{\partial L}{\partial \mathbf{F}} = \mathbf{U}\Big\{\Big(\mathbf{K} \odot \Big(\mathbf{U}^\top \frac{\partial L}{\partial \mathbf{U}}\Big)\Big) + \Big(\frac{\partial L}{\partial \mathbf{\Sigma}}\Big)_{diag}\Big\}\mathbf{U}^\top, \tag{20}$$

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

$$\mathbf{K}(i,j) = \begin{cases} \frac{1}{\sigma_j - \sigma_i}, & i \neq j \\ 0, & i = j \end{cases} \tag{21}$$

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

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{U}} &= \Big\{\frac{\partial L}{\partial \mathbf{H}} + \Big(\frac{\partial L}{\partial \mathbf{H}}\Big)^\top\Big\}\mathbf{U}h(\mathbf{\Sigma}), \\ \frac{\partial L}{\partial \mathbf{\Sigma}} &= h'(\mathbf{\Sigma})\mathbf{U}^\top \frac{\partial L}{\partial \mathbf{H}}\mathbf{U}. \end{aligned} \tag{22}$$

  $h(\mathbf{\Sigma})$ is defined in Eq. (18) and $h'(\mathbf{\Sigma})$ is given by

$$h'(\mathbf{\Sigma})(i,j) = \begin{cases} \sigma_i^{\lambda_i - 1}, & i = j \\ 0, & i \neq j \end{cases} \tag{23}$$

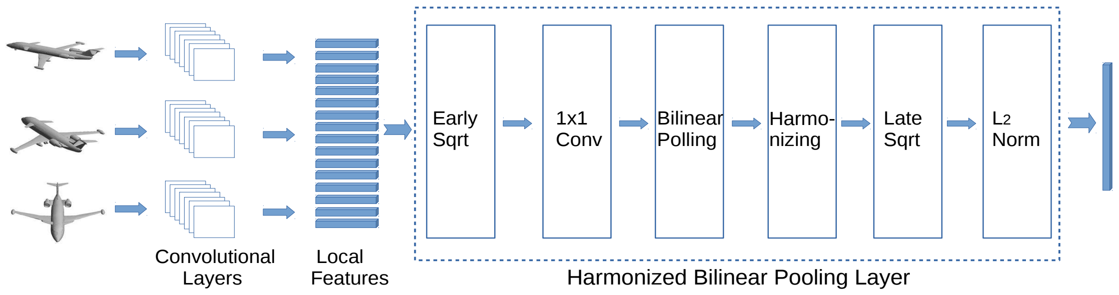


Figure 5. The architecture of the proposed Multi-view Harmonized Bilinear Network (MHBN).

The readers can refer to [13] for detailed derivation of Eq. (20) (21) (22). We derive Eq. (19) here. By plugging Eq. (18) into Eq. (17), we obtain

$$\mathbf{H} = \sum_{k=1}^{d} \frac{\sigma_k^{\lambda_k} - 1}{\lambda_k} \mathbf{u}_k \mathbf{u}_k^\top . \tag{24}$$

Therefore,

$$\begin{aligned} \frac{\partial L}{\partial \lambda_k} &= vec(\frac{\partial L}{\partial \mathbf{H}})^\top vec(\frac{\partial \mathbf{H}}{\partial \lambda_k}) \\ &= \frac{\lambda_k \sigma_k^{\lambda_k} ln(\lambda_k) - \sigma_k^{\lambda_k} + 1}{\lambda_k^2} \mathbf{u}_k{}^\top \frac{\partial L}{\partial \mathbf{H}} \mathbf{u}_k . \end{aligned} \tag{25}$$

- **Late sqrt sub-layer** aims to further suppress the burstiness and reshape the 2D matrix into a 1D vector:

$$\mathbf{v} = sign(vec(\mathbf{H})) \odot \sqrt{vec(|\mathbf{H}|)}. \tag{26}$$

In back-propagation phase, we can compute $\frac{\partial L}{\partial \mathbf{H}}$ by

$$\frac{\partial L}{\partial \mathbf{H}} = mat\Big(\frac{1}{2\sqrt{|vec(\mathbf{H})|}} \odot \frac{\partial L}{\partial \mathbf{v}}\Big), \tag{27}$$

where $mat(\cdot)$ is the function reshaping the input 1D vector into 2D matrix.

- ℓ_2**-norm sub-layer** conducts the ℓ_2-normalization:

$$\hat{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|_2. \tag{28}$$

In the back-propagation phase, $\frac{\partial L}{\partial \mathbf{v}}$ is computed by

$$\frac{\partial L}{\partial \mathbf{v}} = \frac{1}{\|\mathbf{v}\|_2}\Big(\frac{\partial L}{\partial \hat{\mathbf{v}}} - \hat{\mathbf{v}}\Big(\hat{\mathbf{v}}^\top \frac{\partial L}{\partial \hat{\mathbf{v}}}\Big)\Big). \tag{29}$$

6. Experiment

6.1. 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 [4] as our base model. Despite that some more advanced networks are proposed such as ResNet [8] and DenseNet [10], 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 layer after conv5. We initialize the weights of harmonizing sub-layer $\{\lambda_k\}_{k=1}^d$ by 0.5 and the weights of $1 \times 1 \times d \times D$ conv sub-layer by PCA. A dropout layer is added after harmonized bilinear-pooling layer and we set the dropout ratio as 0.5. Like MVCNN [29] and RCPCNN [32], after training, we replace the last layer of MHBN with a linear SVM as the classifier in the testing phase.

6.2. Datasets and evaluation metrics

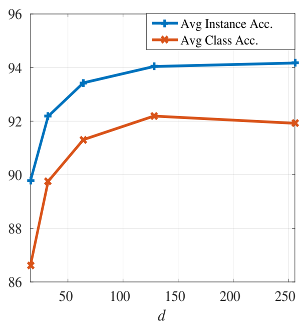
ModelNet40 [34] consists of 12311 3D models from 40 categories. The models are split into 9843 training samples and 2468 testing samples. ModelNet10 [34] consists of 4899 3D models split into 3991 training samples and 908 testing samples from 10 categories. Following MVCNN-MultiRes [24], we report both average instance accuracy and average class accuracy. Average instance accuracy counts the percentage of the correctly recognized testing samples among all the testing samples whereas the average class accuracy is the average accuracy cross all the classes.

6.3. Influence of the number of views

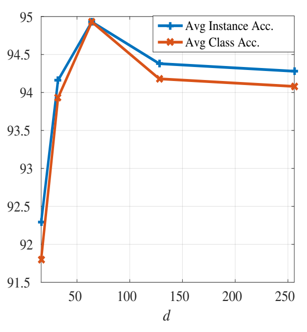
Method	3 vews	6 views	12 views
MVCNN [29]	91.33	92.01	91.49
RCPCNN [32]	92.10	92.22	92.18
MHBN(ours)	**93.78**	**94.12**	**93.42**

Table 1. Influence of the number of views.

We evaluate the effect of the number of views on the average instance accuracy of our MHBN on the Modelnet40 dataset. Meanwhile, we compare it with that from MVCNN [29] and recurrent clustering and pooling (RCPCNN) [32]. The accuracies of MVCNN and RCPCNN shown in Table 1 are taken from Table 3 of Wang *et al.* [32]. As shown



(a) ModelNet40



(b) ModelNet10

Figure 6. Influence of local convolutional feature dimension d.

in Table 1, our MHBN consistently outperforms 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.

6.4. Influence of dimension of local features

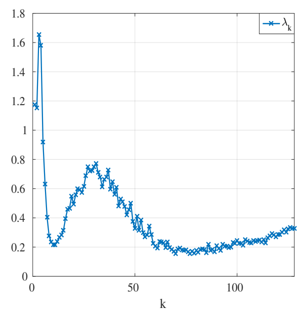
The conv sub-layer aims to reduce the dimension of original local convolutional features from D (512) to d. We evaluate the influence of d on the performance of our MHBN. As shown in Figure 6, on the ModelNet40 dataset, the average instance/class accuracy generally improves as the dimension of local 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.

6.5. Influence of early sqrt and late sqrt sub-layers

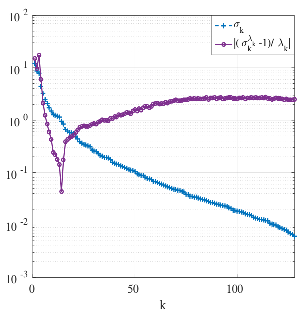
Early sqrt ?			✓
Late sqrt ?		✓	✓
ModelNet40 class ac.	90.00	91.09	**92.23**
ModelNet40 instance ac.	93.24	93.63	**94.12**
ModelNet10 class ac.	93.03	93.75	**94.91**
ModelNet10 instance ac.	93.39	93.94	**94.93**

Table 2. Influence of early sqrt and late sqrt sublayers.

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.



(a) λ_k



(b) σ_k and $|(\sigma_k^{\lambda_k} - 1)/\lambda_k|$

Figure 7. The visualization of λ_k, σ_k and $|(\sigma_k^{\lambda_k} - 1)/\lambda_k|$.

Method	ModelNet10		ModelNet40	
	Class	Inst.	Class	Inst.
Sum pooling	90.24	91.18	85.20	88.04
Max pooling	91.02	91.40	85.65	87.35
Bilinear pooling [20]	88.71	89.76	84.81	87.03
Improved BP [19]	93.11	93.17	91.21	93.23
Log-covariance [12]	93.43	93.50	90.57	93.03
Ours	**94.91**	**94.93**	**92.23**	**94.12**

Table 3. Comparison with other pooling methods.

6.6. Comparison with other pooling methods

In this section, we compare the performance of the proposed harmonized bilinear-pooling with sum-pooling, max-pooling, bilinear pooling [20], improved bilinear pooling [19], log-covariance pooling [12]. 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 3, ours consistently outperforms all other pooling methods on both ModelNet10 and ModelNet40 datasets. In Figure 7, we visualize $\{\lambda_k\}_{k=1}^d$ as well as the scales of components before and after harmonization on the ModelNet10 dataset.

6.7. Additional modality

Note that, all the experiments in the previous sections are conducted using only RGB images. Analogous to [32, 15], 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 for RGB views and depth views separately. We concatenate the output of the harmonized bilinear-pooling layer of RGB-MHBN and that of Depth-MHBN as the final feature of a 3D object, and train a linear SVM as classifier. As shown in Table 5, incorporating the additional depth modality indeed improves average instance/class accuracy.

Method	Views #	Modality	ModelNet40		ModelNet10	
			Instance	Class	Instance	Class
3DShapeNets [34]	-	Volume	-	77.3	-	83.5
VoxNet [22]	-	Volume	-	83.0	-	92.0
3DGAN [33]	-	Volume	-	83.3	-	91.0
Subvolume [24]	-	Volume	89.2	86.0	-	-
VRN [2]	-	Volume	91.3	-	93.6	-
VRN-Ensemble [2]	-	Volume	95.5	-	97.1	-
GIFT [1]	64	RGB	-	83.1	-	92.4
MVCNN [29]	12	RGB	92.1	89.9	-	-
MVCNN-MultiRes [24]	20	3-Resolution RGB	93.8	91.4	-	-
Pairwise [15]	12	RGB	-	90.7	-	92.8
Pairwise [15]	12	RGB + Dep	-	91.1	-	93.2
FusionNet [9]	20	RGB + Volume	-	90.8	-	93.1
RCPCNN [32]	12	RGB	92.2	-	-	-
RCPCNN [32]	12	RGB + Dep + Surf	93.8	-	-	-
PointNet [23]	-	Points	89.2	86.2	-	-
Kd-Network [16]	-	Points	91.8	88.5	94.0	93.5
PointNet++ [25]	-	Points	-	90.4	-	-
PointNet++ [25]	-	Points + Normal	-	91.7	-	-
MHBN (Ours)	6	RGB	**94.1**	**92.2**	**94.9**	**94.9**
MHBN (Ours)	6	RGB + Dep	**94.7**	**93.1**	**95.0**	**95.0**

Table 4. Comparison with state-of-the-art methods.

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 5. The influence of additional modality.

6.8. Comparison with state-of-the-art methods

We first compare with the volume-based methods [33, 34, 22, 24]. As shown in Table 4, the volume-based methods are generally not as good as ours and other view-based methods. One exception is VRN-Ensemble [2], which beats all the view-based methods and ours. Nevertheless, the excellent performance of VRN-Ensemble is owing to the model ensembles and a more advanced base model. It ensembles 5 ResNet models and 1 Inception model [30]. Using a single ResNet model, VRN only achieves 91.3 accuracy on the ModelNet40 dataset, which is worse than MVCNN and ours using a single simpler VGG-M model.

As to view-based methods, MVCNN [29] achieves 92.1/89.9 average instance/class accuracy using 12 views. MVCNN-MultiRes [24] improves MVCNN by exploiting 3 resolutions with 20 views in each resolution. It trains 3 networks for each resolution and achieves 93.8/91.4 average instance/class accuracy. In contrast, our MHBN achieves 94.7/93.1 average instance/class accuracy using 6 RGB views and 6 depth views using a single resolution and two networks. Three recent works, RCPCNN [32] , Kd-Network [16] and PointNet++ [25] are compared as well. As shown in Table 4, our MHBN outperforms them in both single-modality mode and multi-modality mode.

7. Conclusion

In this paper, we propose multi-view harmonized bilinear network for 3D object recognition. By exploiting the relation between bilinear pooling and the polynomial kernel, we obtain a compact global representation through bilinear pooling local convolutional features, which can emphasize the pairs consisting of relevant patches. Meanwhile, we harmonize the singular values of the pooled bilinear feature to generate a more discriminative 3D object representation. Moreover, we implement our balanced bilinear pooling as a layer of a network, which is trainable in an end-to-end manner. Systematic experiments conducted on public benchmark datasets demonstrate the effectiveness of our method.

Acknowledgment: This work is supported in part by start-up grants of University at Buffalo, Computer Science and Engineering Department. This work is supported in part by Singapore Ministry of Education Academic Research Fund Tier 2 MOE2015-T2-2-114. This research was carried out at the ROSE Lab at the Nanyang Technological University, Singapore. The ROSE Lab is supported by the National Research Foundation, Prime Ministers Office, Singapore. We gratefully acknowledge the support of NVAITC (NVIDIA AI Technology Centre) for their donation for our research.

References

[1] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. Jan Latecki. Gift: A real-time and scalable 3d shape search engine. In *CVPR*, 2016.

[2] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016.

[3] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. *ECCV*, 2012.

[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[5] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie. Kernel pooling for convolutional neural networks. In *CVPR*, 2017.

[6] P. Fischer, A. Dosovitskiy, and T. Brox. Descriptor matching with convolutional neural networks: a comparison to sift. *arXiv preprint arXiv:1405.5769*, 2014.

[7] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *CVPR*, pages 317–326, 2016.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[9] V. Hegde and R. Zadeh. Fusionnet: 3d object classification using multiple data representations. *arXiv preprint arXiv:1607.05695*, 2016.

[10] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[11] 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 *ICML*, 2015.

[12] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *ICCV*, 2015.

[13] C. Ionescu, O. Vantzos, and C. Sminchisescu. Training deep networks with structured layers by matrix backpropagation. *CoRR*, abs/1509.07838, 2015.

[14] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *CVPR*, 2009.

[15] E. Johns, L. Stefan, A. Davison, and A. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR*, 2016.

[16] R. Klokov and V. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *ICCV*, 2017.

[17] S. Kong and C. Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *CVPR*, 2017.

[18] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas. Fpnn: Field probing neural networks for 3d data. In *NIPS*, 2016.

[19] T.-Y. Lin and S. Maji. Improved bilinear pooling with cnns. In *BMVC*, 2017.

[20] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015.

[21] J. L. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? In *NIPS*, 2014.

[22] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015.

[23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[24] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, 2016.

[25] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. 2017.

[26] R. Sakia. The box-cox transformation technique: a review. *The statistician*, pages 169–178, 1992.

[27] N. Sedaghat, M. Zolfaghari, and T. Brox. Orientation-boosted voxel nets for 3d object recognition. *arXiv preprint arXiv:1604.03351*, 2016.

[28] A. Shashua. Introduction to machine learning: Class notes 67577. *CoRR*, abs/0904.3664, 2009.

[29] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[31] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models, 2000.

[32] C. Wang, M. Pelillo, and K. Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. In *BMVC*, 2017.

[33] 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 *NIPS*, 2016.

[34] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015.