Revisiting Bilinear Pooling: A Coding Perspective

Zhi Gao,[1] Yuwei Wu,[1]* Xiaoxun Zhang,[2] Jindou Dai,[1] Yunde Jia,[1] Mehrtash Harandi[3]
[1]Beijing Laboratory of Intelligent Information Technology
School of Computer Science, Beijing Institute of Technology, Beijing, China
[2]Alibaba Group
[3]Department of Electrical and Computer Systems Eng., Monash University, and Data61, Australia
{gaozhi2017, wuyuwei, daijindou, jiayunde}@bit.edu.cn, xiaoxun.zhang@alibaba-inc.com, mehrtash.harandi@monash.edu

Abstract

Bilinear pooling has achieved state-of-the-art performance on fusing features in various machine learning tasks, owning to its ability to capture complex associations between features. Despite the success, bilinear pooling suffers from redundancy and burstiness issues, mainly due to the rank-one property of the resulting representation. In this paper, we prove that bilinear pooling is indeed a similarity-based coding-pooling formulation. This establishment then enables us to devise a new feature fusion algorithm, the factorized bilinear coding (FBC) method, to overcome the drawbacks of the bilinear pooling. We show that FBC can generate compact and discriminative representations with substantially fewer parameters. Experiments on two challenging tasks, namely image classification and visual question answering, demonstrate that our method surpasses the bilinear pooling technique by a large margin.

Introduction

Bilinear Pooling (BiP) provides an expressive representation to fuse features by exploiting the higher-order information captured in the form of pairwise correlations between features (Lin, RoyChowdhury, and Maji 2015). Various studies show the superiority of bilinear representations over other fusion techniques such as concatenation, element-wise sum, Hadamard product, and Vector of Locally Aggregated Descriptor (VLAD) (Gong et al. 2014).

Despite the success of BiP in many computer vision tasks including image classification (Lin, RoyChowdhury, and Maji 2015) and heterogeneous multi-modal tasks (Fukui et al. 2016), two shortcomings, *i.e.*, the redundancy and the burstiness, hinder the wide-application of BiP. First, BiP creates a redundant representation in an exceptionally high-dimensional space. For example, in the case of image classification, several studies (Lin, RoyChowdhury, and Maji 2015; Ionescu, Vantzos, and Sminchisescu 2015) leveraged the relatively small VGG-16 network to fuse features, yet they have to process $(512 \times 512 = 262,144)$ dimensional representations as the result of BiP, where 512 is the dimensionality of input features. A recent study by Gao *et al.* showed that in such a high-dimensional space, less than 5% of dimensions are informative (Gao et al. 2016). The dimensionality of bilinear representations unnecessarily increases the memory footprint and incurs heavy computational loads.

Second, the bilinear representation could suffer from the burstiness phenomenon (Wei et al. 2018; Lin and Maji 2017; Li et al. 2017a). Generally speaking, the burstiness corresponds to the problem that features are not invariant enough where the feature elements may have large variances within the same class (Wei et al. 2018). This becomes more dramatic in visual recognition when there exist large illumination and appearance changes, and repeated visual elements. To get a feeling about this issue, in Figure 1a and Figure 1c, we plot bilinear representations for two challenging tasks, namely visual question answering (VQA) using the VQA 2.0 dataset (Goyal et al. 2017) and material classification using the MINC dataset (Quattoni and Torralba 2009). These plots suggest that bilinear representations in the same class may have large variances. This, in return, creates intertwined distributions. For example, in Figure 1a, each of the 'batter' and 'umbrellas' classes has two separate clusters with large distances. Besides, bilinear representations of 'store', 'oven' and 'hand' classes have large intra-class distances, leading to a non-discriminative geometry.

To address the shortcomings of BiP, some methods opt for approximating BiP via the tensor sketch (Gao et al. 2016; Fukui et al. 2016) or factorization (Li et al. 2017b; Kim et al. 2016; Ben-Younes et al. 2017; 2019) to generate compact representations. To improve the discriminative power of BiP, several methods investigate normalization strategies (Li et al. 2017a; Lin and Maji 2017) and orthonormal representations (Wei et al. 2018). Despite providing solutions for the targeted issues, studying compact and discriminative BiP in a unified framework has received little attention and is still a challenging and open problem.

In this paper, we prove that BiP is a form of similarity-based coding-pooling (Riesenhuber and Poggio 1999). This new perspective helps us to better analyze and understand the nature of the redundancy and discriminative issues of BiP. In particular, we will see that bilinear features are rank-one matrices, hence overly redundant in high-dimensional spaces (see Corollary 1 of the 'Further Insights' section).

*Corresponding author

Furthermore, unlike well-established coding techniques, we show that BiP uses a varying dictionary to encode inputs (see Corollary 2 of the 'Further Insights' section). This makes the resulting codes fragile, leading to large intra-distances and non-discriminative distributions.

The coding perspective inspires us to propose a novel fusion technique based on BiP, which we will refer to as factorized bilinear coding (FBC) henceforth. FBC makes use of the concept of sparse coding to reduce the redundancy and obtain a compact representation. Furthermore, by learning a dictionary in an end-to-end manner for coding, FBC generates more discriminative representations. As the name suggests, FBC factorizes the dictionary atoms into low-rank matrices, eliminating the need to explicitly compute high-dimensional bilinear features. This immensely reduces the number of parameters of the model and results in a scalable solution. Our thorough empirical study on the challenging tasks of image classification and VQA demonstrates that the proposed FBC outperforms the BiP technique comfortably, while performing competitively or even exceeding various state-of-the-art algorithms. The code is available at https://github.com/ZhiGaomcislab/FactorizedBilinearCoding.

Contributions. Our main contributions are three-fold.

1. Theoretically, we prove that BiP is indeed a similarity-based coding-pooling formulation. Under this formulation, we provide reasons behind properties that affect the performance of BiP.

2. Based on the coding perspective, we design a new feature fusion algorithm, namely FBC, to encourage compactness and discriminative power of the representation.

3. By factorizing dictionary atoms into low-rank matrices and by avoiding massive matrix operations (*e.g.*, inversion), we achieve a highly scalable solution with small memory footprint.

Notations. Throughout this paper, scalars are denoted by lower-case letters, such as z; vectors are represented by bold lower-case letters, such as $\boldsymbol{z}$, and matrices are denoted by bold upper-case letters, such as $\boldsymbol{Z}$. $\mathrm{Vec}(\cdot)$ and $\top$ show matrix vectorization (*i.e.*, reshaping a matrix to a vector by stacking its columns on top of one another) and the transpose operation, respectively. We denote a p dimensional Hilbert space by $\mathcal{H}_p$.

Bilinear Pooling as Similarity-based Coding

In many problems, fusing features into a combined representation for further processing is important. Let $\{\boldsymbol{x}_s \in \mathbb{R}^p\}_{s=1}^m$ and $\{\boldsymbol{y}_t \in \mathbb{R}^q\}_{t=1}^n$ be two groups of features. In BiP, the fusion is achieved by

$$\begin{aligned} \boldsymbol{Z} &= \sum_{(s,t)\in\mathcal{S}} \boldsymbol{x}_s \boldsymbol{y}_t^\top \in \mathbb{R}^{p\times q}, \\ \boldsymbol{z} &= \mathrm{Vec}(\boldsymbol{Z}) = \sum_{(s,t)\in\mathcal{S}} \mathrm{Vec}(\boldsymbol{x}_s \boldsymbol{y}_t^\top) \in \mathbb{R}^{pq}, \end{aligned} \tag{1}$$

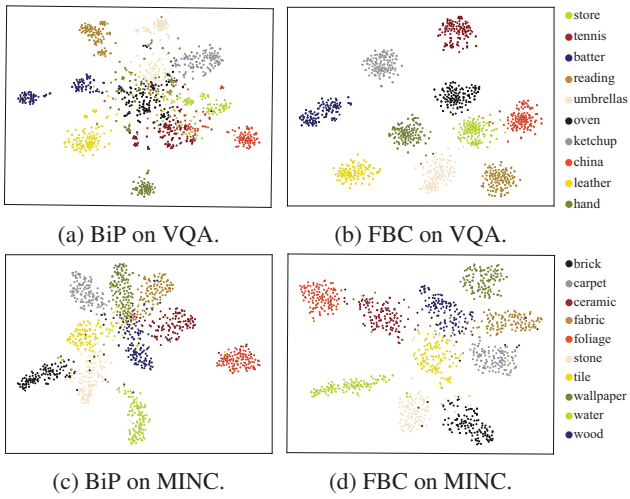
where $\boldsymbol{z}$ is the combined representation, $\boldsymbol{Z}$ is its matrix from, and $\mathcal{S}$ is the feature pair set of the two groups of features. The feature pair set $\mathcal{S}$ has two common forms. In some tasks (*e.g.*, image classification and action recognition (Lin, RoyChowdhury, and Maji 2015; Zhang et al. 2019)), the two groups $\{\boldsymbol{x}_s\}_{s=1}^m$ and $\{\boldsymbol{y}_t\}_{t=1}^n$ have the same number of features, $m = n$, and $\boldsymbol{x}_s$ and $\boldsymbol{y}_s$ are extracted from the same spatial or temporal location of the data. In this case, BiP takes the form

$$\boldsymbol{z} = \sum_{s=1}^{m} \mathrm{Vec}(\boldsymbol{x}_s \boldsymbol{y}_s^\top). \tag{2}$$

In the other form, $\mathcal{S}$ contains all pairs of features from the two groups. This form is commonly used in multi-modal tasks, such as VQA (Kim, Jun, and Zhang 2018) with the fusion being

$$\boldsymbol{z} = \sum_{s=1}^{m}\sum_{t=1}^{n} \mathrm{Vec}(\boldsymbol{x}_s \boldsymbol{y}_t^\top). \tag{3}$$

The dimensionality of the final output $\boldsymbol{z} \in \mathbb{R}^{pq}$ can easily become overwhelming. Existing methods generate compact representations using the aforementioned explicit BiP formulation (Gao et al. 2016; Li et al. 2017b; Yu et al. 2018). Different from them, we show that BiP is a form of similarity-based coding-pooling. We then make use of the coding perspective to develop compact and discriminative representations.



(a) BiP on VQA. (b) FBC on VQA.

(c) BiP on MINC. (d) FBC on MINC.

Figure 1: Feature distributions of BiP and FBC on VQA 2.0 and MINC datasets (best viewed in color) using the t-SNE. Different colors represent different classes. We observe that BiP creates scatter clusters and does not have the locality. It has large intra-distances and the confused distribution. On the contrary, FBC generates discriminative clusters.

The Coding-pooling Formulation

Generally speaking, the coding is mapping a feature from a p dimensional Hilbert space $\mathcal{H}_p$ into a d dimensional Hilbert space $\mathcal{H}_d$. A graceful coding is uniquely defined through a finite set of signals, a.k.a., a dictionary $\boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_k] \in \mathbb{R}^{p\times k}$. For example, in sparse coding, the mapping is identified by solving $\arg\min_{\boldsymbol{z}} \|\boldsymbol{x} - \boldsymbol{B}\boldsymbol{z}\|^2 +$

$\lambda\|\boldsymbol{z}\|_1$. One can seamlessly generalize the above concept to coding on two features. Here, the dictionary $\boldsymbol{B}$ is used to jointly map inputs from two different Hilbert spaces, a p dimensional space $\mathcal{H}_p$ and a q dimensional space $\mathcal{H}_q$ into $\mathcal{H}_d$. A straight-forward approach to implement such coding is to separately encode the two features in $\mathcal{H}_p$ and $\mathcal{H}_q$, followed by concatenating the results. However, such an approach is blind to possible cross-correlations of data points in $\mathcal{H}_p$ and $\mathcal{H}_q$. BiP, to some degree, addresses this shortcoming via Eq. (1).

Eq. (1) shows that the representation $\boldsymbol{z}$ is the sum of bilinear features $\{\mathrm{Vec}(\boldsymbol{x}_s\boldsymbol{y}_t^\top)\}$. It is now clear that it can be interpreted as a coding-pooling scheme of $\{\boldsymbol{x}_s\}_{s=1}^m$ encoded by the dictionary $\boldsymbol{B} = \{\boldsymbol{y}_t\}_{t=1}^n$ or vice versa (*i.e.,* $\{\boldsymbol{y}_t\}_{t=1}^n$ is encoded by the dictionary $\boldsymbol{B} = \{\boldsymbol{x}_s\}_{s=1}^m$). All codes are then sum-pooled into the representation $\boldsymbol{z}$. Considering BiP operates on two inputs, in this paper, we can further derive BiP to another coding-pooling formulation.

Lemma 1. *Given features* $\{\boldsymbol{x}_s \in \mathbb{R}^p\}_{s=1}^m$ *and* $\{\boldsymbol{y}_t \in \mathbb{R}^q\}_{t=1}^n$*, BiP in Eq.* (1) *is equal to a coding-pooling formulation.*

Proof. Via the singular value decomposition of the matrix $\boldsymbol{Z}$, we have

$$\boldsymbol{Z} = \sum_{j=1}^{o} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j{}^\top \tag{4}$$

$$\boldsymbol{Z}\boldsymbol{v}_j = \sigma_j \boldsymbol{u}_j \tag{5}$$

$$\boldsymbol{u}_j^\top \boldsymbol{u}_j = \boldsymbol{v}_j^\top \boldsymbol{v}_j = \mathrm{Tr}(\boldsymbol{u}_j\boldsymbol{u}_j^\top) = \mathrm{Tr}(\boldsymbol{v}_j\boldsymbol{v}_j^\top) = 1, \tag{6}$$

where σ_j is the j-th singular value, $\boldsymbol{u}_j$ is the corresponding left singular vector, $\boldsymbol{v}_j$ is the corresponding right singular vector, o is the rank of $\boldsymbol{Z}$, and $\mathrm{Tr}(\cdot)$ is the matrix trace. We can write the singular value σ_j as

$$\begin{aligned}
\sigma_j &= \sigma_j \mathrm{Tr}(\boldsymbol{u}_j\boldsymbol{u}_j^\top) = \mathrm{Tr}(\sigma_j\boldsymbol{u}_j\boldsymbol{u}_j^\top) = \mathrm{Tr}(\boldsymbol{Z}\boldsymbol{v}_j\boldsymbol{u}_j^\top) \\
&= \mathrm{Tr}\Big(\big(\sum_{(s,t)\in\mathcal{S}} \boldsymbol{x}_s\boldsymbol{y}_t^\top\big)\boldsymbol{v}_j\boldsymbol{u}_j^\top\Big) \\
&= \sum_{(s,t)\in\mathcal{S}} \mathrm{Tr}(\boldsymbol{x_s}\boldsymbol{y_t}{}^\top\boldsymbol{v}_j\boldsymbol{u}_j^\top) \\
&= \sum_{(s,t)\in\mathcal{S}} \langle \mathrm{Vec}(\boldsymbol{x_s}\boldsymbol{y_t}{}^\top), \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\rangle \\
&= \sum_{i=1}^{N} \langle \boldsymbol{f}_i, \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\rangle,
\end{aligned} \tag{7}$$

Here, we denote N the number of pairs in $\mathcal{S}$. For convenience, we reorganize N bilinear features $\{\mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\}$ into an ordered set indexed by i, and denote the bilinear feature as $\boldsymbol{f}_i = \mathrm{Vec}(\boldsymbol{x_s}\boldsymbol{y_t}{}^\top) \in \mathbb{R}^{pq}$, $i \in [1, N]$. We substitute σ_j of Eq. (7) into Eq. (4) and have

$$\begin{aligned}
\boldsymbol{Z} &= \sum_{j=1}^{o}\sum_{i=1}^{N} \langle \boldsymbol{f}_i, \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\rangle \boldsymbol{u}_j\boldsymbol{v}_j^\top \\
&= \sum_{i=1}^{N}\sum_{j=1}^{o} \langle \boldsymbol{f}_i, \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\rangle \boldsymbol{u}_j\boldsymbol{v}_j^\top = \sum_{i=1}^{N} \boldsymbol{C}_i,
\end{aligned} \tag{8}$$

where

$$\boldsymbol{C}_i = \sum_{j=1}^{o} \langle \boldsymbol{f}_i, \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\rangle \boldsymbol{u}_j\boldsymbol{v}_j^\top. \tag{9}$$

By plugging Eq. (8) into the representation $\boldsymbol{z}$ of Eq. (1), we arrive at

$$\boldsymbol{z} = \mathrm{Vec}(\boldsymbol{Z}) = \sum_{i=1}^{N} \mathrm{Vec}(\boldsymbol{C}_i) = \sum_{i=1}^{N} \boldsymbol{c}_i, \tag{10}$$

where $\boldsymbol{c}_i = \mathrm{Vec}(\boldsymbol{C}_i)$. Based on Eq. (9), we have

$$\begin{aligned}
\boldsymbol{c}_i = \mathrm{Vec}(\boldsymbol{C}_i) &= \sum_{j=1}^{o} \langle \boldsymbol{f}_i, \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\rangle \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top) \\
&= \sum_{j=1}^{o} \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)^\top \boldsymbol{f}_i \\
&= \big(\sum_{j=1}^{o} \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)^\top\big)\boldsymbol{f}_i.
\end{aligned} \tag{11}$$

Here we denote

$$\sum_{j=1}^{o} \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)^\top = \boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_{pq}], \tag{12}$$

where $\boldsymbol{B} \in \mathbb{R}^{pq\times pq}$, $\boldsymbol{b}_l \in \mathbb{R}^{pq}$ is the l-th column of $\boldsymbol{B}$, and $\boldsymbol{B} = \boldsymbol{B}^\top$. Thus, BiP of Eq. (1) can be rewritten as

$$\boldsymbol{z} = \sum_{i=1}^{N} \boldsymbol{c}_i, \tag{13}$$

where

$$\boldsymbol{c}_i = [c_i^1, c_i^2, \cdots, c_i^k] = \boldsymbol{B}^\top \boldsymbol{f}_i, \tag{14}$$

$$c_i^l = \langle \boldsymbol{b}_l, \boldsymbol{f}_i\rangle = \boldsymbol{b}_l^\top \boldsymbol{f}_i, \tag{15}$$

$\boldsymbol{c}_i \in \mathbb{R}^{pq}$, and c_i^l is the l-th element of $\boldsymbol{c}_i$. In Eq. (15), Eq. (14), and Eq. (13), $\boldsymbol{B}$ can be understood as a dictionary that contains $k = pq$ atoms. As such, BiP encodes the bilinear features $\boldsymbol{f}_i$ via computing inner product similarities between $\boldsymbol{f}_i$ and the dictionary atoms $\{\boldsymbol{b}_l\}_{l=1}^k$. This concludes the proof that BiP on features $\{\boldsymbol{x}_s\}_{s=1}^m$ and $\{\boldsymbol{y}_t\}_{t=1}^n$ is equivalent to a coding-pooling formulation (Riesenhuber and Poggio 1999) with the dictionary being $\boldsymbol{B} = \sum_{j=1}^{o} \mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)\mathrm{Vec}(\boldsymbol{u}_j\boldsymbol{v}_j^\top)^\top$. □

To summarize and with the dictionary $\boldsymbol{B}$ as above, BiP encodes $\boldsymbol{f}_i = \mathrm{Vec}(\boldsymbol{x_s}\boldsymbol{y_t}{}^\top)$ into $\boldsymbol{c}_i \in \mathbb{R}^k$ with c_i^l, the l-th element of the code, obtained by Eq. (15). Codes are finally sum-pooled into a global representation $\boldsymbol{z}$ by Eq. (13).

Further Insights

From the discussion above, one can identify the following properties that affect the performance of BiP: **(1)** bilinear features $\{\boldsymbol{x}_s\boldsymbol{y}_t^\top\}$ are rank-one matrices, incurring high amount of information redundancy (see Corollary 1); **(2)** the

dictionary $\boldsymbol{B}$ is determined by the input and hence is inconsistent for all data (see Corollary 2); **(3)** when the set $\mathcal{S}$ contains all pairs of features, the representation $\boldsymbol{Z}$ is also a rank-one matrix, and it further makes dictionary atoms $\{\boldsymbol{b}_l\}_{l=1}^k$ collinear and codes $\{\boldsymbol{c}_i\}_{i=1}^N$ collinear (see Corollary 3). The imperfect dictionary causes BiP to have large intra-distances and leads to a non-discriminative distribution.

Corollary 1. *For any given two matrices $\boldsymbol{A} \in \mathbb{R}^{m\times p}$ and $\boldsymbol{B} \in \mathbb{R}^{p\times n}$, the rank of their multiplication $\boldsymbol{AB}$ satisfies* $\mathrm{rank}(\boldsymbol{AB}) \leq \min\big(\mathrm{rank}(\boldsymbol{A}), \mathrm{rank}(\boldsymbol{B})\big)$*, where* $\mathrm{rank}(\cdot)$ *denotes the rank of a matrix. As such, the bilinear feature $\boldsymbol{x}_s\boldsymbol{y}_t^\top$ is a rank-one matrix, as it is the multiplication of two vectors in Eq.* (1)*. In the rank-one matrix, only one row and one column preserve the necessary information, and other elements in the matrix can be represented by this row and column. This clearly shows that the bilinear feature $\boldsymbol{x}_s\boldsymbol{y}_t^\top$ is very redundant, especially in high-dimensional spaces. The redundancy unnecessarily increases the memory footprint and incurs heavy computational loads.*

Corollary 2. *From Eq.* (12)*, we know that the dictionary $\boldsymbol{B}$ is determined by the singular vectors of $\boldsymbol{Z}$, meaning that the dictionary $\boldsymbol{B}$ contains the individual characteristic of the input and varies with the input. As such, BiP codes for different inputs are constructed by disparate dictionaries. This may deteriorate the discriminative power of BiP codes as one should at least implicitly consider variations in the dictionary in the follow-up analysis. We can also associate the large intra-class variations observed in Figure 1a and Figure 1c to variations in the dictionaries (as points in the same class will be processed with different and perhaps contradictory dictionaries). Add to this the fact that the dictionary is local and may not be able to capture the global geometry of the whole data space.*

Corollary 3. *Consider the general form of BiP, replicated for clarity below*

$$\boldsymbol{Z} = \sum_{s=1}^{m}\sum_{t=1}^{n} \boldsymbol{x}_s\boldsymbol{y}_t^\top. \tag{16}$$

Although it may imply that $\boldsymbol{Z}$ is the sum of rank-one matrices, the rank of $\boldsymbol{Z}$ is also one, as $\boldsymbol{Z}$ is the outer product of two vectors: $\sum_{s=1}^m\sum_{t=1}^n \boldsymbol{x}_s\boldsymbol{y}_t^\top = (\sum_{s=1}^m \boldsymbol{x}_s)(\sum_{t=1}^n \boldsymbol{y}_t^\top)$. The rank-one property of $\boldsymbol{Z}$ results in the dictionary atoms $\{\boldsymbol{b}_l\}_{l=1}^k$ in Eq. (12) *to become collinear, and codes $\{\boldsymbol{c}_i\}_{i=1}^N$ in Eq.* (14) *are also collinear.*

Proof. $\boldsymbol{Z}$ being a rank-one matrix implies that $o = 1$ and hence $\boldsymbol{Z} = \sigma\boldsymbol{u}\boldsymbol{v}^\top$. Therefore, the dictionary $\boldsymbol{B}$ and the atom $\boldsymbol{b}_l$ in Eq. (12) are

$$\begin{aligned}\boldsymbol{B} &= \mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)^\top = [\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_k],\\ \boldsymbol{b}_l &= \big(\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\big)_l \mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top),\end{aligned} \tag{17}$$

where $\big(\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\big)_l$ is the l-th element of $\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)$. In the resulting dictionary, atoms only vary in their coefficients while being aligned with $\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)$ and collinear. A dictionary with highly correlated atoms cannot span a big enough space.

The collinearity of atoms results in any two codes $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ to become collinear as well. We recall that in a code $\boldsymbol{c}_i$, c_i^l and $c_i^{l'}$ are similarities between $\boldsymbol{f}_i$ and $\boldsymbol{b}_l$, $\boldsymbol{f}_i$ and $\boldsymbol{b}_{l'}$, respectively. The proportion of c_i^l to $c_i^{l'}$ is

$$\frac{c_i^l}{c_i^{l'}} = \frac{\boldsymbol{b}_l^\top\boldsymbol{f}_i}{\boldsymbol{b}_{l'}^\top\boldsymbol{f}_i} = \frac{\big(\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\big)_l \mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)^\top\boldsymbol{f}_i}{\big(\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\big)_{l'} \mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)^\top\boldsymbol{f}_i} = \frac{\big(\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\big)_l}{\big(\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\big)_{l'}},$$

which only depends on the atom coefficients rather the input features. Thus, the proportion is a constant for codes encoded by the same dictionary. For codes $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$, we assume $c_i^1 = \beta c_j^1$, and have

$$\frac{c_i^l}{c_j^l} = \frac{c_i^l}{c_i^1}\cdot\frac{c_i^1}{c_j^1}\cdot\frac{c_j^1}{c_j^l} = \frac{\big(\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\big)_l}{\big(\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\big)_1}\cdot\beta\cdot\frac{\big(\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\big)_1}{\big(\mathrm{Vec}(\boldsymbol{u}\boldsymbol{v}^\top)\big)_l} = \beta.$$

Thus $\boldsymbol{c}_i = \beta\boldsymbol{c}_j$, and codes are collinear. Such coding formulation encodes inputs into collinear codes and cannot reflect the differences between inputs well. □

Factorized Bilinear Coding

In this section, we present the factorized bilinear coding (FBC) to fuse features based on BiP from the coding perspective. Specifically and in contrast to BiP that uses a dynamic dictionary, we propose of learning a dictionary $\boldsymbol{B}$ to capture the structure of the whole data space. Being vigilant to the redundancy issue of bilinear features, we propose to replace the similarity-based coding with sparse coding to generate a compact representation, which can preserve as much information as possible and activate as few dictionary atoms as possible, reducing unnecessary information and further enhancing fusion results.

Due to the high-dimensionality of bilinear features, naively coding bilinear features of size $p \times q$ using a dictionary $\boldsymbol{B}$ with k atoms demands storing $k \times p \times q$ elements which quickly becomes overwhelming. Our idea here is to factorize dictionary atoms into low-rank matrices. Given a pair of features $(\boldsymbol{x}_s, \boldsymbol{y}_t)$ as the input, FBC encodes $(\boldsymbol{x}_s, \boldsymbol{y}_t)$ into $\boldsymbol{c}_i$ by solving the following optimization problem,

$$\min_{\boldsymbol{c}_i} \left\|\boldsymbol{x}_s\boldsymbol{y}_t^\top - \sum_{l=1}^{k} c_i^l\boldsymbol{U}_l\boldsymbol{V}_l^\top\right\|^2 + \lambda\|\boldsymbol{c}_i\|_1. \tag{18}$$

Here, λ is a trade-off between the reconstruction error and the sparsity. Each dictionary atom $\boldsymbol{b}_l$ is factorized into $\boldsymbol{U}_l\boldsymbol{V}_l^\top$, where $\boldsymbol{U}_l \in \mathbb{R}^{p\times r}$ and $\boldsymbol{V}_l \in \mathbb{R}^{q\times r}$ are low-rank matrices. The rank of decomposition $r \ll p, q$ is a hyper-parameter of the algorithm. In essence, we reconstruct the bilinear feature $\boldsymbol{x}_s\boldsymbol{y}_t^\top$ by $\sum_{l=1}^k c_i^l\boldsymbol{U}_l\boldsymbol{V}_l^\top$, with $\boldsymbol{c}_i \in \mathbb{R}^k$ being the FBC code, and c_i^l representing the l-th element of $\boldsymbol{c}_i$. The l_1-norm $\|\cdot\|_1$ is used to impose the sparsity constraint on $\boldsymbol{c}_i$.

We adopt the LASSO method (Tibshirani 1996) to obtain the FBC code $\boldsymbol{c}_i$ from Eq. (18):

$$\begin{cases}\boldsymbol{c}_i' = \big(\boldsymbol{P}(\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{P}^\top \circ \boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{P}^\top)\big)^{-1}\boldsymbol{P}(\boldsymbol{U}^\top\boldsymbol{x}_s \circ \boldsymbol{V}^\top\boldsymbol{y}_t),\\ \boldsymbol{c}_i = \mathrm{sign}(\boldsymbol{c}_i') \circ \max\big(\mathrm{abs}(\boldsymbol{c}_i') - \dfrac{\lambda}{2}, 0\big).\end{cases} \tag{19}$$

Here, $\circ$ denotes the Hadamard product, $\boldsymbol{U} = [\boldsymbol{U}_1, \cdots, \boldsymbol{U}_k] \in \mathbb{R}^{p \times rk}$ and $\boldsymbol{V} = [\boldsymbol{V}_1, \cdots, \boldsymbol{V}_k] \in \mathbb{R}^{q \times rk}$ are learnable parameters of the dictionary. $\boldsymbol{P} \in \mathbb{R}^{k \times rk}$ is a fixed binary matrix with only elements in the row l, columns $\big((l-1) \times r + 1\big)$ to (lr) being "1", where $l \in [1, k]$.

In Eq. (19), the matrix inversion (*i.e.*, $\big(\boldsymbol{P}(\boldsymbol{U}^\top \boldsymbol{U} \boldsymbol{P}^\top \circ \boldsymbol{V}^\top \boldsymbol{V} \boldsymbol{P}^\top)\big)^{-1}$) is required. In high-dimensional spaces, this can become computationally expensive. Fortunately, there is a workaround here. Let $\boldsymbol{Q}^\top = \big(\boldsymbol{P}(\boldsymbol{U}^\top \boldsymbol{U} \boldsymbol{P}^\top \circ \boldsymbol{V}^\top \boldsymbol{V} \boldsymbol{P}^\top)\big)^{-1} \boldsymbol{P}$. We can rewrite the first part of Eq. (19) as

$$\boldsymbol{c}'_i = \boldsymbol{Q}^\top (\boldsymbol{U}^\top \boldsymbol{x}_s \circ \boldsymbol{V}^\top \boldsymbol{y}_t). \tag{20}$$

Now, the l-th element of $\boldsymbol{c}'_i$ is

$$\boldsymbol{c}'^{l}_i = \boldsymbol{q}_l^\top (\boldsymbol{U}^\top \boldsymbol{x}_s \circ \boldsymbol{V}^\top \boldsymbol{y}_t), \tag{21}$$

where $\boldsymbol{q}_l$ is the l-th column of $\boldsymbol{Q}$. We can further rewrite $\boldsymbol{c}'^{l}_i$ in Eq. (21) as

$$\boldsymbol{c}'^{l}_i = \mathbf{1}_r^\top \left(\left(\tfrac{1}{r} \cdot \boldsymbol{O}\big((\boldsymbol{q}_l \mathbf{1}_{rk}^\top) \circ \boldsymbol{U}^\top\big) \boldsymbol{x}_s \right) \circ \left(\tfrac{1}{r} \cdot \boldsymbol{O} \boldsymbol{V}^\top \boldsymbol{y}_t \right) \right), \tag{22}$$

where $\boldsymbol{O} \in \mathbb{R}^{r \times rk}$ is an all "1" matrix. Similarly, $\mathbf{1}_r \in \mathbb{R}^r$ and $\mathbf{1}_{rk} \in \mathbb{R}^{rk}$ are vectors whose elements are all "1"s. We denote $\widetilde{\boldsymbol{U}}_l^\top = \frac{1}{r} \cdot \boldsymbol{O}\big((\boldsymbol{q}_l \mathbf{1}_{rk}^\top) \circ \boldsymbol{U}^\top\big) \in \mathbb{R}^{r \times p}$ and $\widetilde{\boldsymbol{V}}_l^\top = \frac{1}{r} \cdot \boldsymbol{O} \boldsymbol{V}^\top \in \mathbb{R}^{r \times q}$ to have a simplified form of Eq. (22) as,

$$\boldsymbol{c}'^{l}_i = \mathbf{1}_r^\top (\widetilde{\boldsymbol{U}}_l^\top \boldsymbol{x}_s \circ \widetilde{\boldsymbol{V}}_l^\top \boldsymbol{y}_t). \tag{23}$$

With this, we can now introduce new parameters $\widetilde{\boldsymbol{U}} = [\widetilde{\boldsymbol{U}}_1, \cdots, \widetilde{\boldsymbol{U}}_k] \in \mathbb{R}^{p \times rk}$ and $\widetilde{\boldsymbol{V}} = [\widetilde{\boldsymbol{V}}_1, \cdots, \widetilde{\boldsymbol{V}}_k] \in \mathbb{R}^{q \times rk}$ to replace $\boldsymbol{U}$ and $\boldsymbol{V}$, and the solution of Eq. (18) becomes

$$\begin{cases} \boldsymbol{c}'_i = \boldsymbol{P}(\widetilde{\boldsymbol{U}}^\top \boldsymbol{x}_s \circ \widetilde{\boldsymbol{V}}^\top \boldsymbol{y}_t), \\ \boldsymbol{c}_i = \operatorname{sign}(\boldsymbol{c}'_i) \circ \max\left(\big(\operatorname{abs}(\boldsymbol{c}'_i) - \dfrac{\lambda}{2}\big), 0 \right). \end{cases} \tag{24}$$

As such, we just need to learn $\widetilde{\boldsymbol{U}}$ and $\widetilde{\boldsymbol{V}}$, and the inversion is avoided. The above derivation applies to two individual input features. If two groups of features $\{\boldsymbol{x}_s\}_{s=1}^m$ and $\{\boldsymbol{y}_t\}_{t=1}^n$ are at our disposal, we compute all FBC codes $\{\boldsymbol{c}_i\}_{i=1}^N$ and simply fuse them using the max operation to attain the global representation $\boldsymbol{z}$,

$$\boldsymbol{z} = \max\{\boldsymbol{c}_i\}_{i=1}^N. \tag{25}$$

The whole FBC module is shown in Figure 2c.

Our method leads to large save in the memory footprint and the computational load. For example, in the VQA task, $p = 1024$, $q = 2048$, and there are 3000 classes. In BiP, the classifier needs to store $3000pq > 10^{10}$ parameters. If we first compute the bilinear feature $\boldsymbol{f}_i$, and then deliver it to the non-factorized coding process, we need to store a dictionary $\boldsymbol{B}$ with k complete dictionary atoms, where each dictionary atom contains pq elements. In totally, there are kpq elements in the dictionary, $k \approx 1000$, and we need to store $(kpq + 3000k) > 10^9$ parameters. In contrast to these two solutions, our FBC does not need to compute the high-dimensional bilinear feature $\boldsymbol{f}_i$ explicitly, and the spatial complexity of the dictionary atom is reduced from $\mathcal{O}(pq)$ to $\mathcal{O}((p+q)r)$. Since $r \ll p, q$, our method requires quite fewer parameters. In our implementation, we set $r = 5$. Thus, we just need to store about $(p+q)rk + 3000k \approx 10^7$ parameters. Furthermore, in Eq. (24) we just need to compute three matrix multiplications and a Hadamard product, rather than seven matrix multiplications, two Hadamard products, and a matrix inversion in Eq. (19).

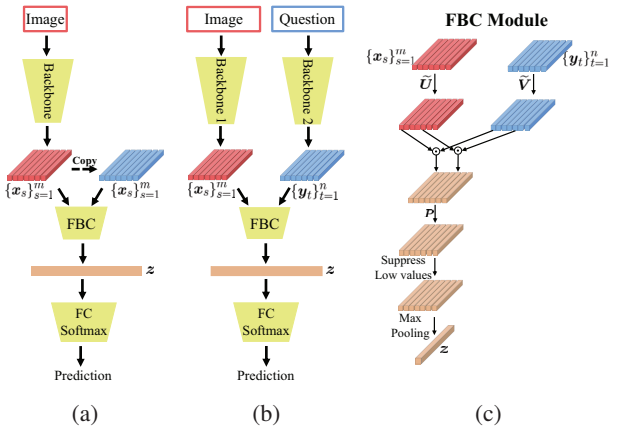


Figure 2: Our network architecture. Figure 2a shows the network architecture on the image classification task. Figure 2b shows the network architecture on the VQA task. Figure 2c shows the architecture of our FBC module.

Experiments

In order to evaluate the performance of the proposed FBC module, we conduct experiments on the challenging image classification and VQA tasks.

Implementation

FBC can be readily incorporated into Deep Neural Networks (DNNs). For the task of image classification and VQA, the architecture of DNNs used in this work is shown in Figure 2a and Figure 2b. In the image classification task, features $\{\boldsymbol{x}_s\}_{s=1}^m$ are extracted using a DNN (*i.e.*, the VGG-16 network), and then $\{\boldsymbol{x}_s\}_{s=1}^m$ and their copies are sent to our FBC module to generate the representation $\boldsymbol{z}$ (see Eq. 25). In the VQA task, features $\{\boldsymbol{x}_s\}_{s=1}^m$ and $\{\boldsymbol{y}_t\}_{t=1}^n$ are first extracted from the image and the question, respectively. Then $\{\boldsymbol{x}_s\}_{s=1}^m$ and $\{\boldsymbol{y}_t\}_{t=1}^n$ are fed to our FBC module to generate the representation $\boldsymbol{z}$. Finally, the representations $\boldsymbol{z}$ from the FBC module are sent to the softmax classifier. Updating parameters $\widetilde{\boldsymbol{U}}$ and $\widetilde{\boldsymbol{V}}$ of the FBC module is simply achieved by the backpropagation algorithm.

Evaluation on the Image Classification Task

We conduct experiments to compare FBC with existing BiP methods on the image classification task. Four datasets are used: Describing Texture Dataset (DTD) (Cimpoi et al. 2014), MINC-2500 (MINC) (Bell et al. 2015), MIT-Indoor

Table 1: Comparisons for BiP methods in terms of Average Precision (%).

Method	Feature dim.	param	DTD	Indoor	MINC	CUB
VGG-16 (Simonyan and Zisserman 2014)	4096	120.4M	60.1	64.5	73.0	66.1
B-CNN (Lin, RoyChowdhury, and Maji 2015)	2.6×10^5	52.4M	67.5	77.6	74.5	84.0
DeepO2P (Ionescu, Vantzos, and Sminchisescu 2015)	2.6×10^5	52.4M	66.1	72.4	69.3	-
CBP (Gao et al. 2016)	8192	1.64M	67.7	76.8	73.3	84.0
MPN (Li et al. 2017a)	32896	13.1M	68.0	76.5	76.2	84.1
LRBP (Kong and Fowlkes 2017)	100	0.21M	65.8	73.6	69.0	84.2
FBN (Li et al. 2017b)	-	0.39M	67.8	-	-	82.9
SMSO (Yu and Salzmann 2018)	2048	1.46M	69.3	79.5	78.0	**85.0**
FBC	512	0.63M	70.3	79.6	79.7	83.6
FBC	2048	2.51M	**71.7**	79.8	79.6	83.6
FBC	8192	10.0M	71.5	**79.9**	**80.2**	84.3

(Indoor) (Quattoni and Torralba 2009), and Caltech-UCSD Bird (CUB) (Xie et al. 2013), which are the texture dataset, the material dataset, the indoor scene dataset, and the fine-grained dataset, respectively. Following the work in (Yu and Salzmann 2018), the size of input images in DTD, Indoor, and CUB is 448×448, and the size of input images in MINC is 224×224. We use the VGG-16 network as the backbone, and layers after the *conv5-3* layer are removed. Our FBC module is on the top of the *conv5-3* layer, and the obtained representation $\boldsymbol{z}$ is followed by an FC layer and a softmax layer. The accuracies are shown in Table 1, where '**param**' denotes the number of parameters after the last convolutional layer. We set the rank of $\widetilde{\boldsymbol{U}}$ and $\widetilde{\boldsymbol{V}}$ as 1, the number of atoms as 512, 2048, and 8192, and λ as 0.001.

We can find that BiP scheme improves a significant margin than the VGG-16 model, as BiP can capture more richer information. Compared with B-CNN, our method further improves the accuracy. FBC can achieve 71.7% on the DTD dataset, 79.9% on the Indoor dataset, 80.2% on the MINC dataset, and 84.3% on the CUB dataset. When the number of atoms is 512, FBC requires only 0.63M parameters, and its performance can still exceed several BiP methods. Compared with CBP, LRBP, FBN, and SMSO which aim to generate compact representations, FBC using comparable parameters achieves competitively and even higher accuracies. Especially on the MINC dataset, our accuracy is about 2% higher than these methods. Note that, FBN is also a factorization based method, while FBC is 2.5% and 0.7% higher than it on Indoor and CUB datasets. MPN applies normalization to improve the discriminative power of BiP representations. Compared with it, FBC has more than 3% improvements on DTD, Indoor, and MINC datasets, and requires fewer parameters. Experiments show FBC can generate a not only compact but also discriminative representation.

Evaluation on the VQA Task

The VQA system is able to answer questions about images, where combining textual features and visual features is crucial. We use the VQA 1.0 (Agrawal et al. 2017) and VQA 2.0 (Goyal et al. 2017) datasets. For the VQA 1.0 dataset, we extract image features from ResNet-152. To obtain the question features, we use a Glove word embedding module after an RNN (LSTM for VQA1.0 and GRU for VQA2.0 following (Anderson et al. 2018)). For the VQA 2.0 dataset, we utilize bottom-up features (Anderson et al. 2018) to describe images.

Table 2: Accuracies on VQA 1.0 and VQA 2.0 val split.

Method	param	VQA1.0	VQA2.0
Concatenation	0.77M	54.4	53.2
Element-wise sum	0.38M	52.3	51.6
Hadamard product	0.38M	54.9	55.8
Bilinear	49.15M	55.8	56.5
Bilinear-VLAD	983.4M	58.9	60.0
Bilinear-similarity	19.6M	58.9	59.8
Bilinear-BoW	19.6M	56.3	56.9
MCB	49.15M	57.4	-
MLB	8.94M	57.9	-
MUTAN	6.07M	58.2	58.2
MFB	18.36M	58.3	58.4
BAN	46.85M	-	65.6
FBC	18.8M	**61.6**	**62.0**
FBC+Att	35.62M	-	**65.7**

Comparisons between Coding Schemes We assess the performance of our FBC algorithm against various coding techniques including the similarity-based coding, Bag of Words (BoW), and VLAD. In addition, we contrast FBC against five state-of-the-art BiP methods: MCB (Fukui et al. 2016), MLB (Kim et al. 2016), MUTAN (Ben-Younes et al. 2017), MFB (Yu et al. 2017), and BAN (Kim, Jun, and Zhang 2018), where BAN is equipped with an advanced residual attention mechanism, and the others are not.

In FBC, we set the rank of $\widetilde{\boldsymbol{U}}$ and $\widetilde{\boldsymbol{V}}$ as 5 and the number of atoms as 1024. λ is set as 0.01. Due to the memory limitation, for the three coding techniques, we add a convolutional layer with the 1×1 kernel on the two modality features to change the dimensionality to 128, thereby the dimensionality of their bilinear features is 128×128. We use 1024 centers for BoW and similarity-based coding, and 80 centers for VLAD. We choose the final dimensionality of 16000 for MCB, 1200 for MLB, 1000 for MFB, and 1280 for BAN. For a fair comparison, we replace our max pooling in Eq. (25) with the advanced attention mechanism (Kim, Jun, and Zhang 2018) of 2 glimpses, denoted by 'FBC+Att'. Results can be found in Table 2. All models are trained on the training split and evaluated on the validation split.

Coding techniques on bilinear features lead to a clear boost in the accuracy. In comparison to these coding tech-

Table 3: VQA 2.0 Test-dev and Test-standard results evaluated on the VQA Challenge 2019 platform.

Method	Att	Test-dev				Test-standard			
		All	Y/N	Num	Other	All	Y/N	Num	Other
BottomUp (Anderson et al. 2018)	✓	65.3	81.8	44.2	56.1	65.7	82.2	43.9	56.3
MCB (Fukui et al. 2016)	✓	-	-	-	-	62.3	78.8	38.3	53.4
MUTAN (Ben-Younes et al. 2017)	✓	66.0	82.9	44.5	56.5	66.4	-	-	-
MLB (Kim et al. 2016)	✓	66.3	83.6	44.9	56.3	66.6	-	-	-
MFB (Yu et al. 2017)	✓	65.0	-	-	-	-	-	-	-
MFH (Yu et al. 2018)	✓	68.8	84.3	49.6	59.9	-	-	-	-
BLOCK (Ben-Younes et al. 2019)	✓	67.6	83.6	47.3	58.5	67.9	84.0	46.8	58.8
MuRel (Cadene et al. 2019)	✓	68.0	84.8	49.8	57.9	68.4	-	-	-
BAN (Kim, Jun, and Zhang 2018)	✓	69.5	85.3	50.9	60.3	69.9	85.6	50.5	60.7
BAN + Counter (Kim, Jun, and Zhang 2018)	✓	70.0	85.4	54.4	60.5	70.4	85.8	53.7	60.7
FBC	×	65.9	83.0	43.5	57.0	-	-	-	-
FBC+Att	✓	69.7	85.3	50.7	60.6	70.1	85.7	50.4	61.0
FBC+Att+Counter	✓	70.0	85.5	53.2	60.6	70.3	85.6	53.4	61.0

niques, FBC is a head and shoulders above (*e.g.,* 2.0% higher than VLAD which is the best coding technique on VQA2.0). FBC also comfortably outperforms state-of-the-art BiP methods. For example, the performance of MFB (Yu et al. 2017) reads as 58.3% and 58.4% on the VQA 1.0 and VQA 2.0 datasets, which is 3.3% and 3.6% less than that of FBC. BAN (Kim, Jun, and Zhang 2018) utilizes an advanced attention mechanism and achieves the expressive performance. Compared with BAN, 'FBC+Att' has the same configuration, while our method obtains a better result, 65.7% on VQA 2.0, and requires fewer parameters.

Comparisons with State-of-the-art Methods We compare FBC against state-of-the-art BiP methods on the VQA 2.0 test set (see Table 3). All models are trained on the training and validation splits, and the Test-dev and Test-standard results are evaluated on the VQA Challenge 2019 platform. 'FBC+Att' uses the attention mechanism (Kim, Jun, and Zhang 2018) of 8 glimpses. We also evaluate FBC combined with the counter model (Zhang, Hare, and Prügel-Bennett 2018) following (Kim, Jun, and Zhang 2018).

FBC without any attention mechanism achieves comparable results with MUTAN, MLB, and MFB that use the attention mechanism. For example, MFB achieves 65.0% on the Test-dev split, while FBC achieves 65.9%, 0.9% higher than it. When FBC is equipped with the attention mechanism (*i.e.,*, 'FBC+Att'), it has the same configuration with MFH and BAN. They achieve 68.8% and 69.5%, while 'FBC+Att' is 0.9% and 0.2% higher than them. FBC also achieves better performance than the latest BiP methods: BLOCK and MuRel. These results show representations from FBC are more powerful, and its performance can be further enhanced equipped with advanced VQA techniques.

Analyses of Parameters

The rank r and the atom number k play important roles in FBC. We vary r from 1 to 20, and k from 128 to 8192, and measure the accuracy of FBC on the MINC dataset (see Figure 3). Due to the memory limitation, we do not evaluate the performance of $k = 8192, r = 20$.

We find that FBC is relatively stable for the atom number k. When $k = 8192$ and $r = 1$, FBC achieves the best performance 80.2%. When $k = 128$ and $r = 1$, the accuracy is about 79.3% with only 0.9% decrease. This shows that FBC does not need a large number of atoms, and it can extract essential information into a compact representation. When $k \leq 4096$, as r increases, accuracies first improve and then decrease, showing that low-rank atoms are suitable to the rank-one features, and a large r may bring an overfitting model. As bilinear features are rank-one matrices, and one row and one column can represent the whole feature, a small number of low-rank atoms can construct inputs well.

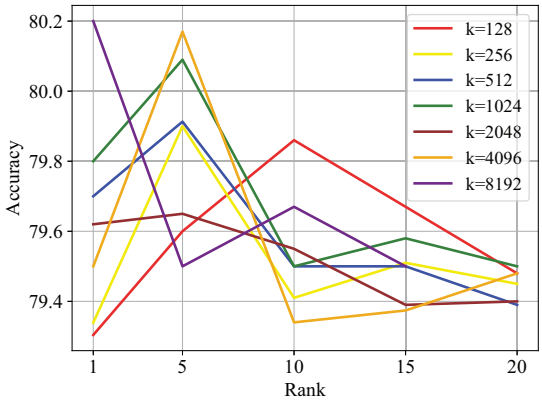


Figure 3: The accuracy on the MINC dataset.

Conclusion

In this paper, we have proved BiP is a similarity-based coding-pooling formulation and presented a factorized bilinear coding (FBC) method to fuse features from the coding perspective. FBC can address the redundancy issue of BiP and generate a compact representation. Our method avoids the explicit computation of high-dimensional bilinear features, and the spatial complexity of required parameters is reduced from $\mathcal{O}(pq)$ to $\mathcal{O}((p+q)r)$ with $r \ll p, q$. Meanwhile, FBC can overcome the burstiness issue. We show that the compact representation from FBC is more discriminative as compared to BiP. Experiments demonstrate that FBC performs competitively or even exceeding various state-of-the-

art algorithms on image classification and VQA tasks.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China (NSFC) under Grants No. 61702037 and No. 61773062, Beijing Municipal Natural Science Foundation under Grant No. L172027, and Alibaba Group through Alibaba Innovative Research Program.

References

Agrawal, A.; Lu, J.; Antol, S.; Mitchell, M.; Zitnick, C. L.; Parikh, D.; and Batra, D. 2017. Vqa: Visual question answering. *International Journal of Computer Vision* 123(1):4–31.

Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; and Zhang, L. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6077–6086.

Bell, S.; Upchurch, P.; Snavely, N.; and Bala, K. 2015. Material recognition in the wild with the materials in context database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 413–420.

Ben-Younes, H.; Cadene, R.; Cord, M.; and Thome, N. 2017. Mutan: Multimodal tucker fusion for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 3, 2612–2620.

Ben-Younes, H.; Cadene, R.; Thome, N.; and Cord, M. 2019. Block: Bilinear superdiagonal fusion for visual question answering and visual relationship detection. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Cadene, R.; Ben-younes, H.; Cord, M.; and Thome, N. 2019. Murel: Multimodal relational reasoning for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cimpoi, M.; Maji, S.; Kokkinos, I.; Mohamed, S.; and Vedaldi, A. 2014. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3606–3613.

Fukui, A.; Park, D. H.; Yang, D.; Rohrbach, A.; Darrell, T.; and Rohrbach, M. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Gao, Y.; Beijbom, O.; Zhang, N.; and Darrell, T. 2016. Compact bilinear pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 317–326.

Gong, Y.; Wang, L.; Guo, R.; and Lazebnik, S. 2014. Multi-scale orderless pooling of deep convolutional activation features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 392–407.

Goyal, Y.; Khot, T.; Summers-Stay, D.; Batra, D.; and Parikh, D. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6904–6913.

Ionescu, C.; Vantzos, O.; and Sminchisescu, C. 2015. Matrix backpropagation for deep networks with structured layers. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2965–2973.

Kim, J.-H.; On, K.-W.; Lim, W.; Kim, J.; Ha, J.-W.; and Zhang, B.-T. 2016. Hadamard product for low-rank bilinear pooling. *arXiv preprint arXiv:1610.04325*.

Kim, J.-H.; Jun, J.; and Zhang, B.-T. 2018. Bilinear attention networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1564–1574.

Kong, S., and Fowlkes, C. 2017. Low-rank bilinear pooling for fine-grained classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7025–7034.

Li, P.; Xie, J.; Wang, Q.; and Zuo, W. 2017a. Is second-order information helpful for large-scale visual recognition? In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2070–2078.

Li, Y.; Wang, N.; Liu, J.; and Hou, X. 2017b. Factorized bilinear models for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2079–2087.

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

Lin, T.-Y.; RoyChowdhury, A.; and Maji, S. 2015. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1449–1457.

Quattoni, A., and Torralba, A. 2009. Recognizing indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 413–420.

Riesenhuber, M., and Poggio, T. 1999. Hierarchical models of object recognition in cortex. *Nature neuroscience* 2(11):1019.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556, 2014*.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 267–288.

Wei, X.; Zhang, Y.; Gong, Y.; Zhang, J.; and Zheng, N. 2018. Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Xie, L.; Tian, Q.; Hong, R.; Yan, S.; and Zhang, B. 2013. Hierarchical part matching for fine-grained visual categorization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1641–1648.

Yu, K., and Salzmann, M. 2018. Statistically motivated second order pooling. *arXiv preprint arXiv:1801.07492, 2018*.

Yu, Z.; Yu, J.; Fan, J.; and Tao, D. 2017. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 3, 1821–1830.

Yu, Z.; Yu, J.; Xiang, C.; Fan, J.; and Tao, D. 2018. Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering. *IEEE Transactions on Neural Networks and Learning Systems* PP(99):1–13.

Zhang, Y.; Tang, S.; Muandet, K.; Jarvers, C.; and Neumann, H. 2019. Local temporal bilinear pooling for fine-grained action parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, Y.; Hare, J.; and Prügel-Bennett, A. 2018. Learning to count objects in natural images for visual question answering. In *Proceedings of International Conference on Learning Representations (ICLR)*.