FASON: First and Second Order Information Fusion Network for Texture Recognition

Xiyang Dai Joe Yue-Hei Ng Larry S. Davis
Institution for Advanced Computer Studies
University of Maryland, College Park
{xdai, yhng, lsd}@umiacs.umd.edu

Abstract

Deep networks have shown impressive performance on many computer vision tasks. Recently, deep convolutional neural networks (CNNs) have been used to learn discriminative texture representations. One of the most successful approaches is Bilinear CNN model that explicitly captures the second order statistics within deep features. However, these networks cut off the first order information flow in the deep network and make gradient back-propagation difficult. We propose an effective fusion architecture - FASON that combines second order information flow and first order information flow. Our method allows gradients to back-propagate through both flows freely and can be trained effectively. We then build a multi-level deep architecture to exploit the first and second order information within different convolutional layers. Experiments show that our method achieves improvements over state-of-the-art methods on several benchmark datasets.

1. Introduction

Features from pre-trained deep models have been successfully utilized for texture recognition [5, 3, 4, 9]. By combining off-the-shelf deep features with second order encoding methods such as Locally Aggregated Descriptors (VLAD) [14] and Fisher Vectors [29], such approaches significantly improve texture recognition performance. However, these approaches require multiple stages of processing including feature extraction, encoding and SVM training, which do not take advantage of end-to-end optimization in deep learning.

Recently, researchers have designed network architectures specifically for texture recognition. One of the most successful approaches is the deep bilinear model proposed by Lin *et al*. [31, 20]. They model the second order statistics of convolutional features within a deep network with a bilinear pooling layer that enables end-to-end training and achieved state-of-the-art performance on benchmark datasets. However, it discards the first order information within the convolutional features, which is known to be useful to capture spatial characteristic of texture [12] and illumination [27]. The first order information is also known to be essential for back-propagation based training [22]. Hence, previous deep bilinear models neglect the potential of first order statistics in convolutional features and make training difficult.

We propose a novel deep network architecture that fuses first order and second order information. We first extend the bilinear network to combine first order information into the learning process by designing a leaking shortcut which enables the first order statistics to pass through and combine with bilinear features. Our architecture enables more effective learning by end-to-end training compared to the original deep bilinear model. This allows us to extend our fusion architecture to combine features from multiple convolution layers, which captures different style and content information. This multiple level fusion architecture further improves recognition performance. Our experiments show the proposed fusion architecture achieves consistent improvements over state-of-the-art methods on several benchmark datasets across different tasks such as texture recognition, indoor scene recognition and fine-grained object classification.

The contribution of our work is two fold:

- We design a deep fusion architecture that effectively combines second order information (calculated from a bilinear model) and first order information (preserved through our leaking shortcut) in an end-to-end deep network. To the best of our knowledge, our architecture is the first proposed method to fuse such information directly in a deep network.
- We extend our fusion architecture to take advantage of the multiple features from different convolution layers.

This paper is organized as following: In section 2, we overview previous deep network based texture representation and elaborate on the difference between our proposed architecture and previous methods. In section 3, we present the design of our fusion architecture, give details and explanations of each building block. In section 4, we describe experiments to evaluate the effectiveness of our architecture, compared with other state-of-the-art methods on standard benchmark datasets and later visualize the improvements. Finally, we conclude the paper in section 5.

2. Background and Related Work

Texture descriptors have been well studied for decades. Classic methods include region co-variance [32], local binary patterns (LBP) [23] and other hand-craft descriptors. Robust texture representations, such as VLAD [14] and Fisher vector [29] combining with SIFT features [21] have been utilized to further improve performance on texture recognition tasks.

Deep texture representations. Donahue *et al*. showed that deep networks can learn generic features from images that can be usefully applied to many computer vision tasks including texture recognition [5]. Zhou *et al*. fine-tuned such pre-trained models on specific texture and scene datasets end-to-end to achieve better performance [35]. Later, Cimpoi *et al*. combined improved Fisher vector encoding [25] with deep features, which further improves performance on various texture recognition datasets [3]. They further collected a large dataset of texture images, which is now considered as the state-of-the-art benchmark for texture recognition. Most recently, Lin *et al*. modeled second order pooling in a deep learning framework and proposed the bilinear network [31]. Later, they applied their framework to texture recognition [20]. Gao *et al*. proposed a compact bilinear network that utilizes random Maclaurin and tensor sketching to reduce the dimensionality of bilinear representations but preserves the discriminative power at the same time [7].

Fusion with first order statistics. Researchers have realized the importance of fusing both first order and second order statistics in feature learning. Hong *et al*. proposed a second order statistics based region descriptor, named "sigma set" [13]. It was first constructed through Cholesky decomposition on the covariance matrix and then fused with the first order mean vector. Later, Doretto *et al*. fused central moments, central moment invariants, radial moments and region covariance together in a compact representation [6]. This representation is invariant to scale, translation, rotation and illumination changes. Recently, Li *et al*. proposed a feature encoding method called locality-constrained affine subspace coding (LASC) that captures both first order and second order information [19]. They showed improvement when fusing first order information into their framework. Similar to this work, we also fuse in the feature learning stage, but we fuse first order and second order information in a deep network trained in an end-to-end manner.

Combining multiple layers of CNNs. Hariharan *et al*. discussed the importance of utilizing different layers from deep networks [10]. They defined a hypercolumn representation, where the descriptor of each pixel was constructed using activations of multiple CNN units above that pixel. Cimpoi *et al*. combined multiple layers of CNN features after training to further improve performance on texture recognition [4]. We also utilize features from multiple convolutional layers to capture different style and content information. However, our approach improves over previous methods by enabling such information to be learned and combined efficiently through our end-to-end architecture.

3. FASON

In this section, we describe our proposed framework *FASON* (First And Second Order information fusion Network). We first introduce the basic components of our first order and second order fusion building blocks. Then, we describe our final deep architecture with multiple level feature fusion.

3.1. Deep Bilinear model

Deep bilinear models have shown promising results on several computer vision tasks including fine-grained image classification and texture classification [31, 7, 20]. Although bilinear models have only recently been integrated in a deep network with end-to-end training, the basic formulation of bilinear models has a long history in texture description. Given an input image I, we extract its deep convolutional features $F \in \mathbb{R}^{w\times h\times ch}$ from a CNN and calculate the bilinear feature $B \in \mathbb{R}^{ch\times ch}$ as:

$$B(F) = \sum_{i=1}^{w}\sum_{j=1}^{h} F_{i,j}F_{i,j}^{T} \tag{1}$$

This formulation is related to orderless texture descriptors such as VLAD , Fisher Vectors and region covariance [31]. It has been shown to be effective to capture the second order statistics of texture features. The diagonal entries of the output bilinear matrix represent the variances within each feature channel, while the off-diagonal entries represent the correlations between different feature channels. The descriptor lies in a Riemannian Manifold which makes quantization difficult and distance measurement between feature vectors non-trivial [24]. Therefore, the bilinear feature is usually passed through a mapping function with signed square root and l_2 normalization that projects it to Euclidean space [25]:

$$\phi(x) = \frac{sign(x)\sqrt{|x|}}{\|sign(x)\sqrt{|x|}\|_2} \tag{2}$$

3.2. Dimension reduction

The high dimensionality of bilinear features makes end-to-end training difficult in a neural network. Following [7], we use a technique called tensor sketching to reduce the $d = ch \times ch$ bilinear output to a feature in a lower dimension c. Tensor sketching [2, 26], which is known to preserve pairwise inner products, estimates the frequency of all elements in a vector. It is a random projection technique to reduce feature dimensionality using multiple random mapping vectors defined by simple independent hash functions.

Given two randomly sampled mapping vectors $h \in \mathbb{N}^d$ where each entry is uniformly drawn from $\{1, 2, \cdots, c\}$, and $s \in \{+1, -1\}^d$ where each entry is filled with either $+1$ or -1 with equal probability, the sketch function is defined as:

$$\Psi(x, s, h) = [C_1, C_2, \cdots, C_c] \tag{3}$$

where

$$C_j = \sum_{i:h(i)=j} s(i) \cdot x(i) \tag{4}$$

To reduce the dimensionality of bilinear features, the $ch \times ch$ size bilinear feature is first vectorized to $x \in \mathbb{R}^d$ where $d = ch \times ch$ and further projected to a lower c-dimensional vector $\mathcal{E} \in \mathbb{R}^c$ by:

$$\mathcal{E}(x) = \mathcal{F}^{-1}(\mathcal{F}(\Psi(x, s, h)) \circ \mathcal{F}(\Psi(x, s', h'))) \tag{5}$$

where s' and h' are drawn similarly to s and h, $\circ$ operator represents element-wise multiplication, and $\mathcal{F}$ represents the Fast Fourier Transformation. We reduce the bilinear representation to $c = 4096$ dimensions in all experiments.

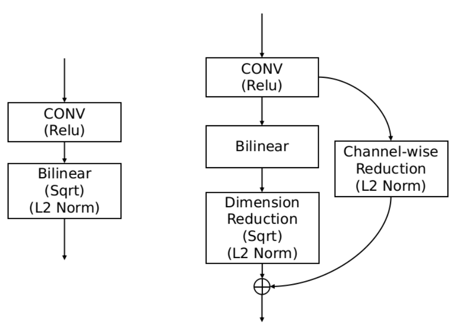

Figure 1: The core building block of our first and second order information fusion architecture compared to original bilinear model.

3.3. First order information fusion by gradient leaking

Now we introduce our core building block of the first order and second order information fusion. Although bilinear models exploit second order information from deep features well, they often suffer from the problem of vanishing gradients when gradient flow back-propagates through them, which makes it difficult to learn such models in an end-to-end training process. Therefore, recent work usually places the bilinear layer after the last convolutional layer to minimize this problem.

Inspired by the recent success of deep residual networks [11], we design a shortcut connection that passes through the first order information and combines with the second order information generated from the bilinear layer, as shown in Figure 1. Assuming we generate the deep feature F from the previous convolutional layer, instead of using the bilinear feature $B(F)$ directly, we combine it with a leaking function $M(F)$ that encodes first order information. Since the bilinear layer essentially captures the covariance between each feature channel, we define our leaking function as:

$$\mathcal{M}(F) = \frac{1}{wh} \sum_{i=1}^{w} \sum_{j=1}^{h} F_{i,j} \tag{6}$$

to provide the mean of each feature channel. This is analogous to global average pooling for a convolutional feature map.

The first order information is then combined with the second order information as follows:

$$\hat{\mathcal{B}}(F) = \mathcal{E}(\mathcal{B}(F)) \oplus \mathcal{M}(F) \tag{7}$$

where $\oplus$ represents the vector concatenation operation.

With the proposed formulation, the first order information can be exploited for classification, and the training can be stabilized as the architecture provides a direct pathway for the gradients to lower layers from the leak.

3.4. First and second order fusion with multiple levels of convolutional features

One benefit of our fusion framework is that we can fuse more convolutional features into bilinear layers and conduct an effective end-to-end training as shown in Figure 2. Given arbitrary convolutional feature maps $F_1, F_2, \cdots F_i$, we can fuse them together simply by:

$$\hat{\mathcal{B}}(F) = \biguplus_i \mathcal{E}(\mathcal{B}(F_i)) \oplus \mathcal{M}(F_i) \tag{8}$$

where $\biguplus_i$ indicates concatenating features generated from different convolutional layers. In this way, we force the network to utilize the first and second order information across

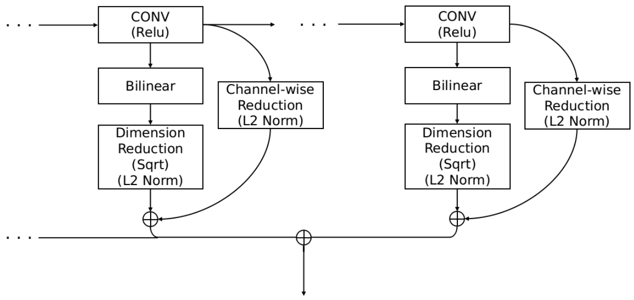

Figure 2: The illustration of how we accumulate first and second order information from multiple convolutional layers.

multiple convolutional feature maps, which generally captures different style and content information.

We investigate two major network architectures: a single fusion at `conv5` level (equivalent to features generated from `conv5_4` of VGG-19 network) and a multiple fusion at `conv4`, `conv5` layers (equivalent to features generated from `conv4_4` and `conv5_4` of VGG-19 network). For fair comparison, we also conduct experiments using typical bilinear networks without fusion on these same two setups. The detailed configurations of our architectures are shown in Figure 3.

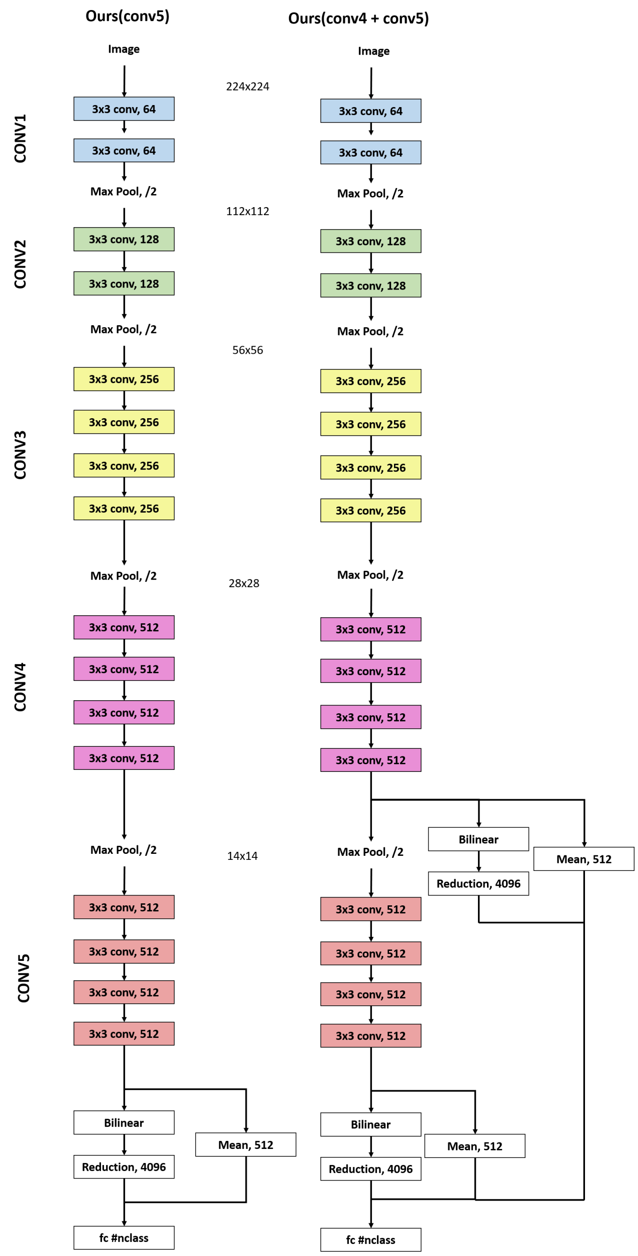

Figure 3: The detailed configurations of our first and second order information fusion architectures. Colored boxes represent convolutional layers with ”3×3” filter size and number of output channels. Notice that we group convolution layers by their spatial dimensions, such as `conv4` and `conv5`. This allows us to maintain the same notation across different network architectures. For all convolutional layers, we use padding size 1 and use ReLU as activation layer. For all the max pooling layers, we use kernel size 2 and stride size 2 without padding.

4. Experiments

We evaluate the effectiveness and performance of our architecture and compare with state-of-the-art methods on several datasets. We also adopt the artistic style transfer technique of [8] to visualize the qualitative improvements of our architecture.

4.1. Datasets and implementations

Datasets. We evaluate our architecture on four benchmark datasets: DTD (Describable Texture) dataset [3], KTH-T2b (KTH-TISP2-b) dataset [1], MIT-Indoor (MIT indoor scene) dataset [28] and Stanford Car196 [18] dataset.

The DTD dataset is considered the most widely used benchmark for texture recognition. It contains 47 texture categories with a total of 5640 images. All images are ”in the wild”, from the web image rather than collected in a controlled setting. This dataset is challenging due to its realistic nature and large intra-category variation. We report the 10-fold average accuracy as in [20].

The KTH-T2b dataset contains 11 material categories with 4752 images total. Images in each category are captured from 4 physical, planar samples under controlled scale, pose and illumination. In our experiments, we follow the evaluation setup in [20] and report the 4-fold average accuracy.

The MIT-Indoor dataset contains 67 indoor scene categories with at least 100 images per category and 15620 images total. The images can be considered as weakly structured and orderless textures, which provide a reasonable evaluation of the generalization power of our texture models. We use the training and testing split provided with the dataset, for a total of 5360 images for training and 1340 images for testing.

The Stanford Car196 dataset contains 196 different fine-grained car classes at the level of make, model and year with 16185 images total. The images are further split into a 50-50 split with 8144 training images and 8041 testing images. This dataset is considered as one the most challenging fine-grained classification dataset. We evaluate on this dataset to further test the generalization of our model.

Implementation details. We implement our architecture in a customized version of Caffe [15]. We adopt a two-stage training process to speed up training. We first fix all layers except the last fully connected layer for classification in the network to form a convex learning problem, and then relax the network to fine-tune all the layers with a constant small learning rate and high momentum. In detail, in the first training step, we use a fixed learning policy with learning rate starting at 1, weight decay at 5×10^{-6} and run for up to 50 epochs. In the second training stage, we fine-tune all the layers using a fixed learning policy with learning rate of 0.001 and weight decay at 5×10^{-4} for another 100 epochs. We did not use data augmentation and dropout for fair comparison to previous work. Incorporating these techniques may further improve the results. For experiments conducted on DTD, KTH-T2b and Stanford Car196 datasets, we use a 224×224 input size, while for the MIT-Indoor dataset we evaluate two setups with 224×224 and 448×448 input size.

4.2. Effectiveness of fusion

We first evaluate the effectiveness of our fusion architecture by comparing two networks with and without first order information fusion on single (conv5) and multiple

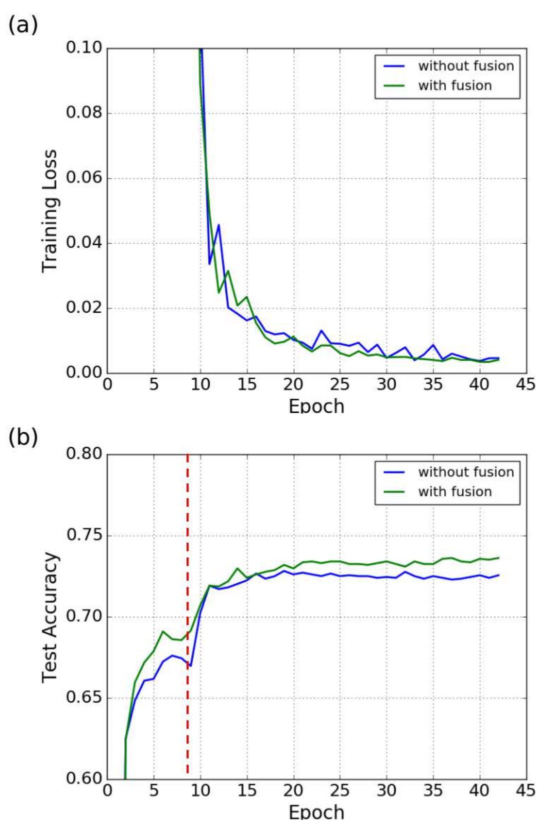

Figure 4: Comparison of learning curves on bilinear model using single level of convolutional feature (conv5) with and without our first order information fusion on DTD dataset.

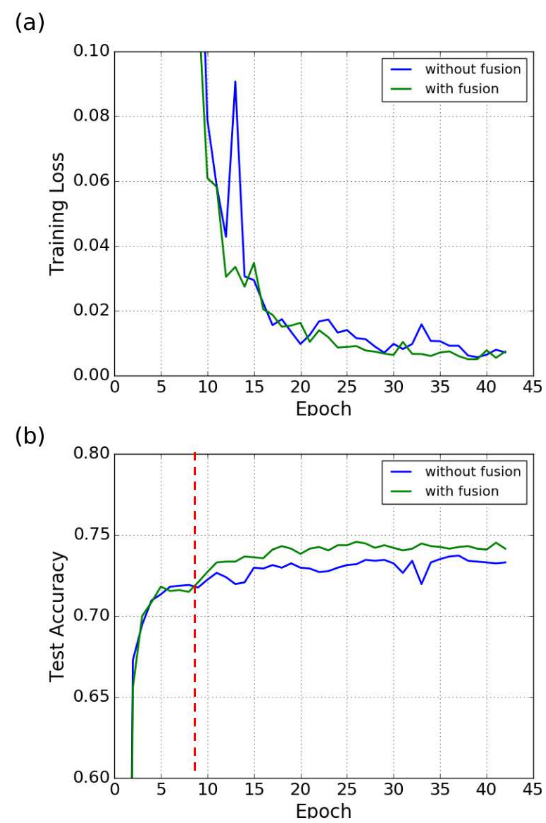

Figure 5: Comparison of learning curves on bilinear model using two levels of convolutional features (conv4+conv5) with and without our first order information fusion on DTD dataset.

(conv4+conv5) convolutional layers. For fair comparison, we use the same learning hyperparameters and training/testing split on the DTD dataset. For the first 8 epochs, we only train the last layer with a fixed learning rate 1. We then fine-tune all the layers with a fixed learning of 0.001. Figure 4 and 5 show both the training loss and testing accuracy. The red vertical lines highlight the point at which we switch from learning only the last layer to learning all layers. As shown by the plots of testing accuracy (Figure 4b and Figure5b), our architecture with first order information fusion clearly outperforms the bilinear network without fusion in both training stages. The training loss plots (Figure 4a and Figure5a) show that our architecture can be trained more smoothly. This is more observable in the multiple convolutional layer setup (Figure5a). In both setups, our experiments demonstrate the effectiveness of our approach.

We also evaluate the effectiveness of our fusion architecture on different deep networks such as VGG-16 and VGG-19. Table 1 shows the performance of our architecture applied to different models on the DTD dataset. Our fusion architectures gives consistent improvements over a baseline bilinear network. The performance further boosts with our multiple layer fusion when two level of convolutional layers conv4 and conv5 are combined. Further combining lower layers in the network did not improve performance. Combining the improvements from first and second order information fusion with multi-layer feature fusion, we obtain a 2% improvement from a strong bilinear CNN baseline for both VGG-16 and VGG-19.

	VGG-16	VGG-19
conv5	72.45	72.82
conv5+fusion	73.09	73.62
Improvements	**+0.64**	**+0.80**
conv5+conv4	72.87	73.31
conv5+conv4+fusion	74.47	74.57
Improvements	**+1.60**	**+1.26**

Table 1: Effectiveness comparison across different network architecture on one training and testing split on DTD dataset. Our models gives consistent improvements from standard Bilinear CNN.

4.3. Comparison with state-of-the-arts

Texture recognition. We compare the performance of our fusion architecture with several state-of-the-art methods such as [3, 4, 20] on the DTD dataset. All the results are reported based on an input size 224×224 for fair comparisons. The methods annotated with * in the table indicates that multiple scales of inputs are used instead of a single input size. Table 2 shows that our method achieves the best

Method	Accuracy
DeCAF + IFV[3]	66.7 ± 0.9
FV-CNN [20]	67.8 ± 0.9
B-CNN [20]	69.6 ± 0.7
FASON (conv5)	$\mathbf{72.3 \pm 0.6}$
FASON (conv4+conv5)	$\mathbf{72.9 \pm 0.7}$
FC-VGG* [4]	62.9 ± 0.8
FV-VGG* [4]	72.3 ± 1.0
FC+FV-VGG* [4]	74.7 ± 1.0
FC-SIFT FC+FV-VGG* [4]	75.5 ± 0.8

Table 2: Comparison with state-of-the-art methods on the DTD dataset with 224×224 input size.

Method	Accuracy
TREE [30]	66.3
DeCAF [3]	70.7 ± 1.6
DeCAF + IFV [3]	76.2 ± 3.1
FV-CNN [20]	74.8 ± 2.6
B-CNN [20]	75.1 ± 2.8
FASON (conv5)	$\mathbf{76.5 \pm 2.3}$
FASON (conv4+conv5)	$\mathbf{76.4 \pm 1.5}$
FC-VGG* [4]	75.4 ± 1.5
FV-VGG* [4]	81.8 ± 2.5
FC+FV-VGG* [4]	81.1 ± 2.4
FC-SIFT FC+FV-VGG* [4]	81.5 ± 2.0

Table 3: Comparison with state-of-the-art methods on the KTH-T2b dataset with 224×224 input size. * denotes results obtained from multiple scales.

Method	Input Size		
	224	448	ms
LASC [19]	63.4	–	–
PLACE [36]	70.8	–	–
FC-VGG [4]	–	–	67.6
FV-VGG [4]	–	–	81.0
FV-CNN [20]	70.1	78.2	78.5
B-CNN [20]	72.8	77.6	79.0
FASON (conv5)	**76.0**	**80.8**	–
FASON (conv4+conv5)	**76.8**	**81.7**	–

Table 4: Comparison with state-of-the-art methods on the MIT-Indoor dataset with different input sizes. *ms* denotes results obtained from multiple scales.

Method	Accuracy
CNN [16]	70.5
ELLF [16]	73.9
CNN Finetuned [34]	83.1
FT-HAR-CNN [34]	86.3
BoT [33]	92.5
Parts [17]	92.8
FV-CNN [31]	85.7
B-CNN [31]	90.6
B-CNN (VGG16 + VGG19) [31]	91.3
FASON (conv5)	**92.5**
FASON (conv4+conv5)	**92.8**

Table 5: Comparison with state-of-the-art methods on the Stanford Car196 dataset using 224×224 input size.

performance among all single-feature methods. In particular, our best model gives 3% improvement over the B-CNN baseline. Our method also outperforms FC-VGG and FV-VGG which use multiple input scales. We also report the fusion results from previous work that use multiple features. Our method is competitive with such methods but requires only one single network.

We also evaluate the performance of our fusion architecture on KTH-T2b dataset with several state-of-the-art methods such as [30, 3, 4, 20]. Similar to the DTD dataset, all reported results are based on input size 224×224, and * in the table also represents multiple scales are used. As shown in Table 3, our method also gains a 1.4% boost compared to the B-CNN baseline. Our best model is only behind FV-VGG which uses deep features calculated at three different scales. Our method with multiple convolutional layers FASON (conv4+conv5) performances slightly worse than a single layer model FASON (conv5) in this dataset. We believe this is caused by the smaller amount of training data provided in the KTH-T2b dataset, which leads to over-fitting. Our method is again competitive to previous approaches.

Indoor scene classification. In addition to the pure texture datasets, we evaluate our models on the MIT-Indoor dataset and compare with state-of-the-art methods such as [19, 36, 4, 20]. We evaluate our model with both 224×224 input size and 448×448 input size. Table 3 shows that our method is superior to previous methods and our best performance is 0.7% better than previous state of the art method FV-CNN, which used multiple scales. Again, our method largely outperforms the B-CNN baseline with about a 4% improvement consistently over input size 224 and 448. Meanwhile, our models also take advantage of larger input size, which is consistent with previous results. Our method with multiple convolutional layers FASON(conv4+conv5) further improves performance over a single convolution layer FASON(conv5).

Fine-grained classification. We further evaluate our models on the Stanford Car196 dataset and compare with popular state-of-the-art methods. Following standard evaluation protocol, we use the provided bounding box during training and testing. All images are cropped around the bounding boxes and then resized to 224×224. We compare with state-of-the-art methods [16, 34, 33, 17] with the same input size. As shown in Table 5, our models improve significantly over the bilinear model [31] when using the same VGG-19 architecture. Meanwhile, our models result in a 1.2% and 1.5% improvement over the best bilinear models mixing different architectures reported in [31]. Our best model is comparable to [17], which utilizes part information of cars to boost performance. Overall, our fusion models have shown promising generalization ability in fine-grained classification task and are competitive to state-of-the-art methods on the Stanford Car dataset.

4.4. Artistic Style Transfer

Artistic style transfer is a popular technique introduced in [8] that transfers the style from one artistic image to another image. Because this technique utilizes a bilinear representation to compute the style loss, performing style transfer provides an intuitive way to visualize and understand what is learned in the networks. To generate visually plausible style transfer results, the networks need to learn a good representation for both content and style. Since the style of an image is closely related to its texture, we apply our learned networks, which have learned good texture representations, to the task of artistic style transfer.

We follow the suggested settings described in [8] and use `conv4_2` layer for content loss with weight 1 and use `conv1_1`, `conv2_1`, `conv3_1`, `conv4_1` and `conv5_1` layers for style loss with weight 0.2. We run L-BFGS for 512 iterations in all experiments.

We compare our fusion architectures with the standard VGG network (using the weights learned form the DTD dataset for classification task) on different combinations of content and style images in Figure 6. The red boxes highlights the major differences in style. With the side-by-side comparison, our model shows richer styles (the cloud and wall appear more stylish in the images) and more accurate content (the contours of building and tower appear to be better preserved) in the generated images. These qualitative results suggest that with our architecture can preserve style and content information more effectively. Furthermore, our multi-layer fusion architecture is even better than our single-layer fusion architecture.

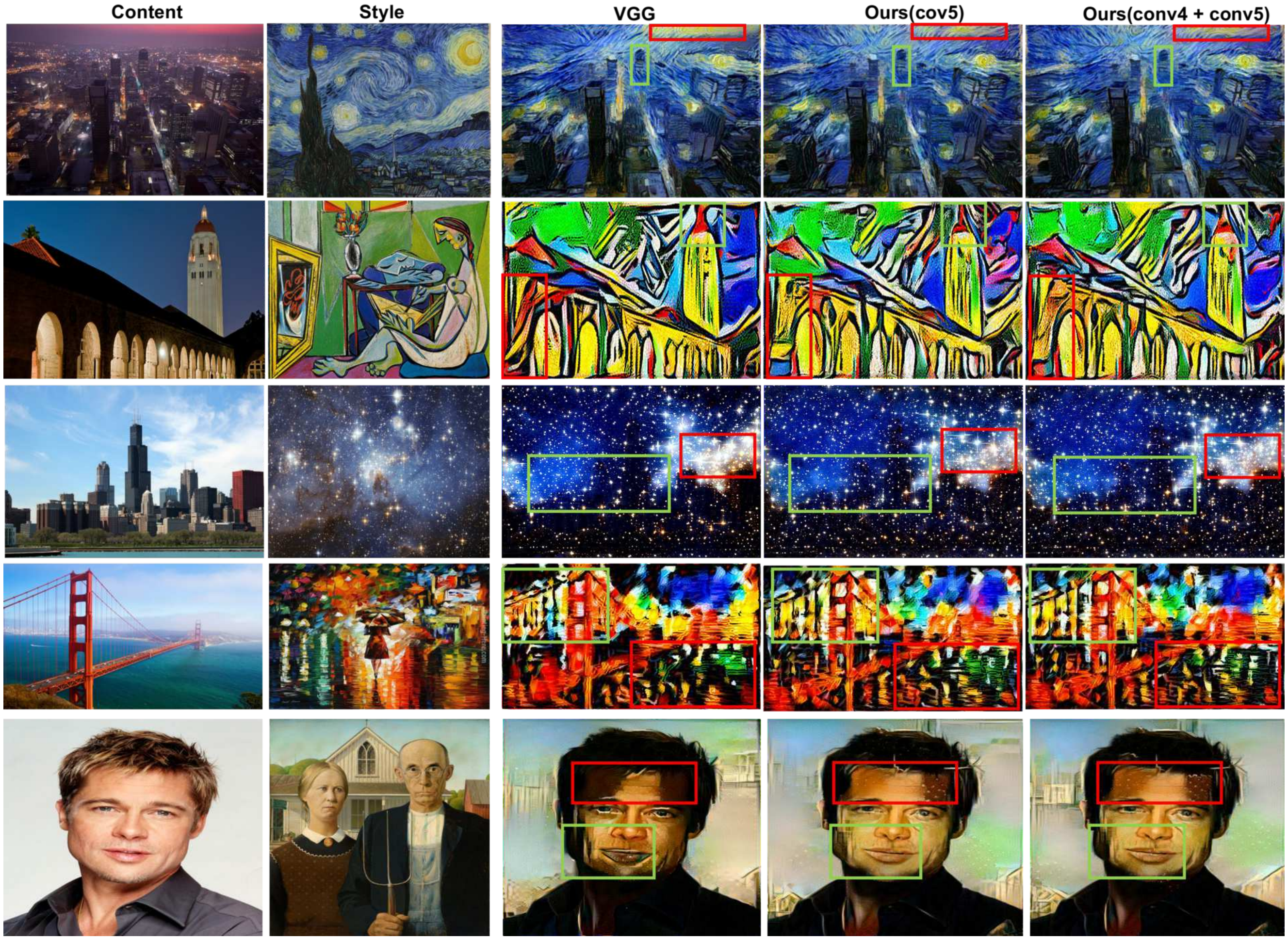

Figure 6: Comparison of style transfer results using different models. The green box highlights the content differences between images. The red box highlight the style difference between images.

5. Conclusion

We presented a novel architecture that aggregates first and second order information within a deep network. Experiments show that our fusion architecture consistently improves over the standard bilinear networks. We additionally propose an architecture combining information from the multiple levels of convolutional layers, which further improves overall performance. Our network can be trained end-to-end effectively. We achieve state-of-the-art performance for a single network on several benchmark datasets. In addition, the better learned texture representation from our network is shown qualitatively by the improved artistic style transfer results.

References

[1] B. Caputo, E. Hayman, and P. Mallikarjuna. Class-specific material categorisation. In *ICCV*, 2005.

[2] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3 – 15, 2004.

[3] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

[4] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

[6] G. Doretto and Y. Yao. Region moments: Fast invariant descriptors for detecting small image structures. In *CVPR*, 2010.

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

[8] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[9] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014.

[10] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

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

[12] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 229–238. ACM, 1995.

[13] X. Hong, H. Chang, S. Shan, X. Chen, and W. Gao. Sigma set: A small second order statistical region descriptor. In *CVPR*, 2009.

[14] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[16] J. Krause, T. Gebru, J. Deng, L.-J. Li, and L. Fei-Fei. Learning features and parts for fine-grained recognition. In *ICPR*, 2014.

[17] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In *CVPR*, 2015.

[18] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.

[19] P. Li, X. Lu, and Q. Wang. From dictionary of visual words to subspaces: Locality-constrained affine subspace coding. In *CVPR*, 2015.

[20] T.-Y. Lin and S. Maji. Visualizing and understanding deep texture representations. In *CVPR*, 2016.

[21] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[22] S. Mohamed. A statistical view of deep learning, 2015.

[23] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[24] X. Pennec, P. Fillard, and N. Ayache. A riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

[25] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[26] N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

[27] T. Pouli, E. Reinhard, and D. W. Cunningham. *Image Statistics in Visual Computing*. A. K. Peters, Ltd., Natick, MA, USA, 1st edition, 2013.

[28] A. Quattoni and A. Torralba. Recognizing indoor scenes. *CVPR Workshop*, 2009.

[29] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013.

[30] R. Timofte and L. V. Gool. A training-free classification framework for textures, writers, and materials. In *BMVC*, 2012.

[31] A. R. Tsung-Yu Lin and S. Maji. Bilinear cnns for fine-grained visual recognition. In *ICCV*, 2015.

[32] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In A. Leonardis, H. Bischof, and A. Pinz, editors, *ECCV*, 2006.

[33] Y. Wang, J. Choi, V. I. Morariu, and L. S. Davis. Mining discriminative triplets of patches for fine-grained classification. In *CVPR*, 2016.

[34] S. Xie, T. Yang, X. Wang, and Y. Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *CVPR*, 2015.

[35] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*. 2014.

[36] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS*. 2014.