Locally-Transferred Fisher Vectors for Texture Classification

Yang Song[1], Fan Zhang[2], Qing Li[1], Heng Huang[3], Lauren J. O'Donnell[2], Weidong Cai[1]
[1]School of Information Technologies, University of Sydney, Australia
[2]Brigham and Women's Hospital, Harvard Medical School, USA
[3]Electrical and Computer Engineering Department, University of Pittsburgh, USA

Abstract

Texture classification has been extensively studied in computer vision. Recent research shows that the combination of Fisher vector (FV) encoding and convolutional neural network (CNN) provides significant improvement in texture classification over the previous feature representation methods. However, by truncating the CNN model at the last convolutional layer, the CNN-based FV descriptors would not incorporate the full capability of neural networks in feature learning. In this study, we propose that we can further transform the CNN-based FV descriptors in a neural network model to obtain more discriminative feature representations. In particular, we design a locally-transferred Fisher vector (LFV) method, which involves a multi-layer neural network model containing locally connected layers to transform the input FV descriptors with filters of locally shared weights. The network is optimized based on the hinge loss of classification, and transferred FV descriptors are then used for image classification. Our results on three challenging texture image datasets show improved performance over the state-of-the-art approaches.

1. Introduction

Texture is a fundamental component in visual recognition. The study of texture, especially feature representation of textures, has evolved over the years from basic statistical features, to the most recent methods based on deep learning. Among the numerous representation methods, we are particularly interested in the feature encoding aspect. While the earlier studies have mainly used the bag-of-words (BOW) model and its variations [14, 11, 34, 13, 30, 19, 18], encoding via Fisher vectors (FV) has become the dominant approach in texture classification [21, 6, 25, 7].

Similar to BOW, FV encoding aggregates the local-level features into the image-level representation. The main uniqueness of FV encoding is the soft assignment of Gaussian components and the computation of first and second order difference vectors. In addition, while typically the dense scale-invariant feature transform (DSIFT) features are the local features used with FV encoding [17, 6, 25], the recent approach has shown that the local features from a convolutional neural network (CNN) model could produce more discriminative FV descriptors [7]. In particular, this study proposes a FV-CNN descriptor, which is computed by FV encoding of the local features extracted from the last convolutional layer of the VGG-VD model (very deep CNN model with 19 layers) [24] pretrained on ImageNet. This FV-CNN descriptor shows large improvement over the more standard FV-DSIFT descriptor [7, 26].

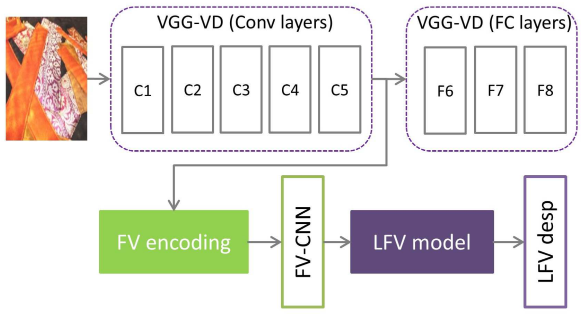

Figure 1. With the VGG-VD model, FV-CNN descriptor is computed by FV encoding of the local features from the last convolutional layer. We design the LFV model, in a multi-layer neural network construct, to further transform the FV-CNN descriptor to a more discriminative LFV descriptor.

Also, for texture classification, this FV-CNN descriptor shows higher classification performance than FC-CNN, which is the descriptor obtained from the penultimate fully connected layer of the CNN [7]. Moreover, we find that even if the pretrained VGG-VD model is fine-tuned on the texture image dataset, the fine-tuned FC-CNN descriptors are still less discriminative than the FV-CNN descriptors. These observations indicate that FV encoding is more effective than the encoding by the fully connected layers in the CNN pipeline. We suggest that the main reason of this advantage is that the GMM model used in FV encoding provides an explicit feature space modeling and this has a higher generalization capability to better capture the complex feature space.

However, with FV-CNN, the benefit of CNN is not fully utilized since it is truncated at the last convolutional layer. To better incorporate the learning capability of a CNN model, there is a trend to create end-to-end learning by mimicking the handcrafted encoding in a CNN model. For example, in the NetVLAD model [1], customized layers are inserted in place of the fully connected layers to generate a descriptor similar to the VLAD encoding. However, our experiments show that this NetVLAD model is less effective than FV-CNN descriptors in the texture classification problem. We find that besides the reason that VLAD encodes only first order differences, the classification performance of NetVLAD is also limited by the design of the fully connected layer connecting the high-dimensional VLAD descriptor with the softmax loss layer.

In this work, we consider that since the multi-layer neural network model (with fully connected layers) is very different from the GMM construct, both algorithms (FV encoding and neural network) could discover complementary information to represent the images effectively. Therefore, it could be helpful to integrate the FV encoding with a neural network model, rather than using a single model in place of the other, so that the advantages of both algorithms would be incorporated. We expect that the integrated model would generate descriptors with higher discriminative power.

We thus design a locally-transferred Fisher vector (LFV) method to further transform the FV-CNN descriptor in a neural network model (as shown in Figure 1). Briefly, we design a multi-layer neural network model, with the FV-CNN descriptors as the input layer and a final layer representing the hinge loss of classification. The intermediate layers comprise a locally connected layer, with local filters that transform the input data into a lower dimension. The filter weights are shared locally so that the data transform is performed differently on the sub-regions of the FV-CNN descriptor. Compared to FV-CNN, this LFV method helps to integrate the benefit of discriminative neural network in feature learning. Also when compared to end-to-end learning, the capability of FV encoding in representing the complex feature space is retained by keeping the FV-CNN component. Therefore, instead of attempting to use a single CNN model to encompass the benefits of both FV encoding and neural network, it becomes a simpler problem to design the additional neural network model on top of FV-CNN descriptors.

We performed experiments on three texture image datasets, including the KTH-TIPS2 dataset [4], the Flickr Material Dataset (FMD) [20], and the Describable Texture Datasets (DTD) [6]. We demonstrate improved performance over the recent approaches [7, 12].

1.1. Related work

The current state-of-the-art approaches for texture classification include the one with FV-CNN descriptors [7] and the bilinear CNN (B-CNN) model [12]. Both approaches use the pretrained VGG-VD model as the base network, but with different encoding techniques, i.e. FV versus bilinear encoding. The two encoding techniques provide similar classification performance with FV-CNN having a smaller feature dimension.

When applying the pretrained VGG-VD model to the texture image datasets, it could be intuitive to consider fine-tuning the model first on the specific dataset [3, 15, 16]. For FV-CNN and B-CNN models, the fine-tuning needs to be conducted down to the convolutional layers to take effect. However, it is reported in [12] that fine-tuning the VGG-VD model on the texture image datasets leads to negligible performance difference. This could be due to the small number of images available for training in the texture datasets. The B-CNN model also has the advantage of an end-to-end learning capability with its neural network construct. However, such learning requires a large image dataset and has only been performed on ImageNet [12].

A particularly interesting end-to-end learning model is the NetVLAD [1]. In this model, the outputs from the last convolutional layer are used as input to the VLAD layer, which contains learnable parameters and can be computed with convolution and softmax operations. The model is however designed for place recognition. When applied to texture images, we find that the classification performance is lower than FV-CNN, partly due to the formulation of only first order differences. Another study proposes a FisherNet model, which adds layers with similar effects to FV encoding, incorporating both first and second order differences [28]. However, this model is quite complicated requiring an explicit patch generation layer, rather than using the local features from the convolutional layers. Another model, namely HistNet, is recently proposed to simulate the histogram / BOW encoding in the CNN model [33]. However, without the first and second order difference information, such a network might not be suitable for texture classification problems.

There are also other ways to improve the FV descriptors. For example, dimensionality reduction with a large margin construct is designed and shows improvement in face recognition over the high-dimensional FV descriptor [23]. Also, with deep Fisher networks [22], multiple Fisher layers are stacked and combined with a global layer to produce the final descriptor, and discriminative dimensionality reduction is learned in each layer. In another study [27], the Gaus-

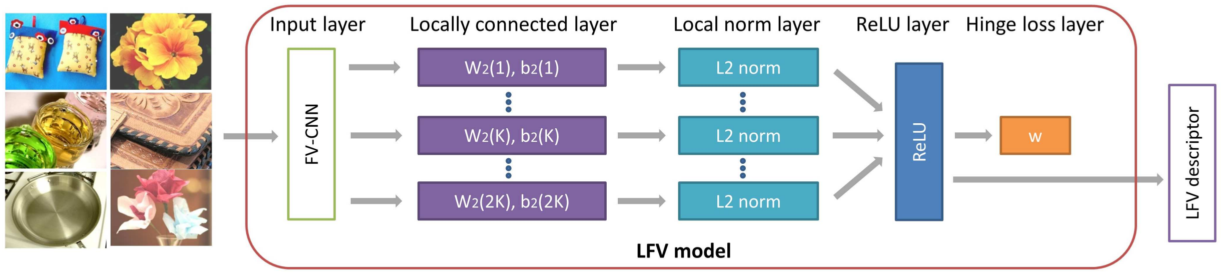

Figure 2. Our LFV model comprises the input layer, locally connected layer, local normalization layer, ReLU layer, and the hinge loss layer. The input layer is the FV-CNN descriptor. The output at the ReLU layer is the LFV descriptor generated.

sian parameters are integrated into the SVM learning objective to achieve end-to-end learning of both FV encoding and SVM classification. In addition, approximate Fisher kernel [8] is designed to incorporate latent variable modeling into Fisher encoding, so that local features need not be identically and independently distributed (iid). An intra-normalization technique [2], which is originally proposed for the VLAD descriptor, has also been applied to FV descriptors recently [10]. With this technique, each feature block is individually normalized to reduce the bursty effect in the descriptor encoding. These approaches are however less coupled with the CNN model and not designed for texture image classification.

2. Preliminary

FV encoding computes an image-level descriptor by aggregating the local patch-level features. The key step in FV encoding is to generate a Gaussian mixture model (GMM) with K components from the local features of the training set. To obtain the FV descriptor of an image, the local features in this image are soft assigned to each Gaussian component. Then based on the soft assignment, the average first and second order differences between the local features and K Gaussian components are computed and concatenated to produce the FV descriptor.

In this study, we focus on the FV-CNN descriptor. Given an image I and the VGG-VD model pretrained on ImageNet, the 512-dimensional local features are derived from the last convolutional layer of the VGG-VD model. These local features of training images are then pooled together to generate the GMM model, and encoded accordingly to produce the FV-CNN descriptor. The dimension of the FV-CNN descriptor h is $2KD$, with $D = 512$ and K is set to 64 following the approach used in [7].

3. Locally transferring Fisher vectors

We design the LFV method in a multi-layer neural network model. Figure 2 gives an overview of our model, which comprises five layers. The first input layer is simply the FV-CNN descriptor. In a CNN sense, this input layer has a size of $1 \times 1 \times (2KD) \times N$, with N as the batch size during training. We denote the nth input vector in the batch as $h(n)$.

The second layer is a locally connected layer. It consists of $2K$ filters, with each filter of D_1 neurons. Each filter is fully connected to a section of D inputs in the input layer, and produces D_1 outputs. Formally, the output $f_2(n,i) \in \mathbb{R}^{D_1}$ corresponding to the input $h(n)$ from the ith filter is computed as:

$$f_2(n,i) = W_2(i)h(n,i) + b_2(i) \tag{1}$$

where $h(n,i) \in \mathbb{R}^D$ is the ith section in the input vector $h(n)$, $W_2(i) \in \mathbb{R}^{D_1 \times D}$ is the weight matrix of the ith filter, and $b_2(i) \in \mathbb{R}^{D_1}$ is the bias vector. Also, to reduce the number of parameters, we choose to have every four consecutive filters share the same weights, hence there are a total of $2K/4$ unique filters in this layer. The total output dimension of the second layer is $1 \times 1 \times (2KD_1) \times N$. Note that with D_1 set to 64, this layer effectively condenses the FV descriptor to a lower dimension.

The third layer is a local normalization layer. Each output $f_2(n,i)$ from the second layer is L2 normalized so that the various sections have the same importance in the transferred descriptor. The fourth layer is a ReLU layer, with ReLU activation applied to the $1 \times 1 \times (2KD_1) \times N$ dimensional output of the previous layer. We denote the output of the input $h(n)$ at the fourth layer as $f_4(n)$, which can be summarized as:

$$f_4(n) = ReLU(\{\|f_2(n,1)\|_2, \ldots, \|f_2(n,2K)\|_2\}). \tag{2}$$

This $f_4(n)$ is then the transferred FV descriptor LFV from our model.

The last layer is the loss layer, which gives a loss value of classification based on the output f_4 from the previous layer. We define this layer with the hinge loss. Specifically, assume that the dataset contains L image classes. A one-versus-all multi-class linear-kernel classification model is formulated, with one weight vector $w_l \in \mathbb{R}^{2KD_1}$ for each class $l \in \{1, \ldots, L\}$. The loss value ε is computed as:

$$\frac{1}{2}\sum_{l=1}^{L} w_l^T w_l + C \sum_{l=1}^{L}\sum_{n=1}^{N} \max(1 - w_l^T f_4(n)\lambda(n,l), 0) \quad (3)$$

where $\lambda(n,l) = 1$ if the nth input vector $h(n)$ belongs to class l and $\lambda(n,l) = -1$ otherwise. Minimizing this loss value at the last layer is thus analogous to minimizing the margin in an SVM classifier.

3.1. Design explanation

In our network design, we consider the second layer conceptually similar to a fully connected layer in the VGG-VD model, which is useful for transforming the input data to a lower dimension. However, we choose to use the locally connected structure rather than fully connected, since we consider that it would be difficult to have a single filter that would effectively transform the long FV descriptors. By using the local filters, varying feature patterns could potentially be explored in different sub-regions of the FV descriptors, and the collective results from local filters could improve the overall results. Also, we set the section size as 512, which is the dimension of the local feature. Each filter thus corresponds to the mean or variance vector of one Gaussian component in the GMM model. Furthermore, although we could have one filter for each 512-dimensional section, the amount of learnable parameters would be huge and overfitting would be a problem for the small size of dataset. We thus experimented with a number of strategies to merge filters with weight sharing. We found that the simple technique of having a common filter for every four consecutive sections could provide good performance.

For the loss layer, we suggest that since LFV descriptors will be finally classified using linear-kernel SVM, the commonly used softmax loss function is not well aligned with the SVM classification objective. We thus choose to use an SVM formulation in this loss layer based on the standard hinge loss. This design is similar to the method in [29], but we explicitly define the multi-class classification loss. In addition, while it is reported in [29] that the L2-SVM formulation (squared sum of losses) provides better performance than L1-SVM (linear sum of losses), we found that L1-SVM is more effective in the texture image classification problem.

Overall, by transferring the FV descriptor using the proposed model, the benefits of FV encoding and discriminative learning of neural network are integrated in a simple manner. We also keep the network design simple with minimal layers to reduce the network complexity and the risk of overfitting. We do however suggest that it could be possible to further enhance the network with varying configurations (e.g. more locally connected layers and a different D_1), especially if the method is applied to a different dataset.

3.2. Parameter learning

The forward and backward passes of the locally connected layer can be implemented by a combination of $2K/4$ standard fully connected neural networks to learn the parameters W_2 and b_2. The input data to each network is of size $1 \times 4 \times D \times N$ and the output is of size $1 \times 4 \times D_1 \times N$. The combination of all individual outputs then gives a total dimension of $1 \times 1 \times (2KD_1) \times N$. Standard implementation is also used for the L2 normalization and ReLU layers. For the loss layer, the loss function can be differentiated with respect to $f_4(n)$ and w_l to obtain the derivatives for backpropagation. In particular, we obtain the following:

$$\frac{\partial \varepsilon}{\partial f_4(n)} = -C \sum_{l=1}^{L} \lambda(n,l) w_l \mathbf{1}(1 > w_l^T f_4(n)\lambda(n,l)) \quad (4)$$

and

$$\frac{\partial \varepsilon}{\partial w_l} = w_l - C \sum_{n=1}^{N} \lambda(n,l) f_4(n) \mathbf{1}(1 > w_l^T f_4(n)\lambda(n,l)) \quad (5)$$

where the regularization parameter C is set to 0.1.

The parameters W_2, b_2, and w_l are initialized by treating the local filters as individual networks and training them separately based on the sections of FV-CNN descriptors. In other words, we create $2K$ separate networks, with each one used to train one filter as the initial values; and we found such an initialization process to be particularly useful for the FMD dataset. This initialization process leads to considerable improvement in classification results over the random initialization. In addition, we also found that adding a dropout layer with rate 0.5 before the loss layer can further reduce the feature redundancy and improve the final classification result slightly. This is thus incorporated into the network when learning parameters.

4. Experiments

4.1. Datasets and implementation

We used three texture image datasets for experiments. The KTH-TIPS2 dataset contains 4752 images from 11 material classes, with each class of 432 images. The images in each class are divided into four samples of different scales. Following the standard protocol, one sample is used for training and three samples are used for testing during each split. The FMD dataset contains 1000 images from 10 material classes with each class of 100 images. During experiments, half of the images are randomly selected for training and the other half for testing. The DTD dataset contains 5640 images from 47 texture classes, with each class having 120 images. Unlike KTH-TIPS2 and FMD, the images in DTD have varying sizes. DTD is also considered as the most challenging dataset since it contains images in the

Table 1. The classification accuracies (%), comparing our LFV method with FV-CNN [7], FV-CNN computed with fine-tuned VGG-VD model (backpropagation to the last convolutional layer), FV descriptor generated with end-to-end CNN learning similar to the NetVLAD model (backpropagation to the FV layer), and B-CNN [12]. Linear-kernel SVM classification is performed with all compared approaches.

Dataset	Our LFV	FV-CNN	Fine-tuned FV	End-to-end FV	B-CNN
KTH-TIPS2	82.6±2.6	81.4±2.4	80.9±2.3	78.5±2.1	80.2±2.8
FMD	82.1±1.9	79.7±1.8	79.3±2.1	76.9±1.6	80.5±1.6
DTD	73.8±1.0	72.4±1.2	71.8±1.0	68.2±1.3	71.9±1.0

wild. For DTD, the training / testing splits published with the dataset are used, and within each split, 2/3 of the images are used for training and 1/3 for testing. For all datasets, four splits of training and testing are conducted, and the mean accuracy is used as the performance metric.

When generating the FV-CNN descriptors, we follow the approach in [7]. The images are scaled to multiple sizes, with scales of 2^s, $s = -3, -2.5, \dots, 1.5$, and the VGG-VD model (with 19 layers) is applied to each scale. The local features from the last convolutional layer are pooled together to generate a GMM with $K = 64$ Gaussian components. The resultant FV-CNN descriptor is then $2KD = 65536$ dimensional. This high-dimensional FV-CNN descriptor is then input to the LFV model to obtain the transferred descriptors. The learning rates of the various layers are set to 0.05 and the batch size N is set to 50. The LFV model provides a discriminative dimensionality reduction and reduces the descriptor dimension to $2KD_1 = 8192$. Linear-kernel SVM is finally used to classify the LFV descriptors. Our code was implemented based on VLFeat [31] and MatConvNet [32] libraries.

4.2. Compared approaches

For performance comparison, we evaluated the following approaches. For all approaches, VGG-VD is used as the base model, and linear-kernel SVM is used as the classifier.

Pretrained model. FV-CNN descriptors are generated with the VGG-VD model pretrained on ImageNet. This is the same approach proposed in [7], and also the input to our LFV model.

Fine-tuned model. FV-CNN descriptors are also computed by first fine-tuning the VGG-VD model on the texture image dataset. The fine-tuning is performed in a standard manner with the backpropagation stopped at various convolutional layers.

End-to-end learning of FV descriptor. We also experiment with an end-to-end CNN-based learning method to derive the FV descriptors. To do this, we modify the NetVLAD model to replace the VLAD layer with an FV layer while keeping all the other layers unchanged. Also, a fully connected layer of L neurons (L being the number of image classes) and a softmax loss layer are appended at the end of the NetVLAD model for parameter learning. The FV layer is constructed following the design in [28]. Briefly, in the FV layer, a weight vector w_k and bias vector b_k are defined corresponding to each Gaussian component k. The first and second order difference vectors are computed using element-wise product and sum operations between the weight vector, local feature, and bias vector. This layer is differentiable and hence can be embedded into the CNN model. Note that the model is initialized using the pretrained VGG-VD model, and the resultant FV descriptor is also $2KD$ dimensional.

Include FC-CNN. As reported in [7], the FC-CNN descriptor provided much lower results than FV-CNN, but can be concatenated with FV-CNN to obtain a more discriminative feature representation. We also evaluated the classification performance by concatenating FC-CNN with our LFV descriptor. For this concatenation, the 4096-dimensional FC-CNN descriptor obtained from the penultimate layer of VGG-VD is transformed using a model similar to LFV, but with FC-CNN as the input, and the section size D is set to 64 (a convenient number). We found that this transformed FC-CNN descriptor gives better classification results than simply concatenating the original FC-CNN descriptor.

B-CNN. The B-CNN encoding is also used to obtain the image descriptors. Similar to FV-CNN, the images are scaled to multiple scales and the features from different scales are pooled together.

Dimension reduced descriptor. Since our IFV descriptor effectively reduces the feature dimension of the original FV-CNN descriptor, we also compare with the other dimensionality reduction algorithms, including principal component analysis (PCA), linear discriminant analysis (LDA), the compact bilinear pooling designed to reduce the B-CNN descriptor [9], and a simple fully connected layer in place of the locally connected layer in our LFV model.

4.3. Results

Table 1 lists the classification results using (i) original FV-CNN obtained using VGG-VD pretrained on ImageNet, (ii) FV-CNN from fine-tuned VGG-VD model, (iii) FV descriptor with end-to-end learning, (iv) B-CNN, and (v) our LFV model. The results show that our LFV method

Table 2. The classification accuracies (%) of LFV and LFV combined with FC-CNN.

Dataset	LFV	LFV + FC-CNN
KTH-TIPS2	82.6±2.6	83.1±1.8
FMD	82.1±1.9	83.5±1.6
DTD	73.8±1.0	75.2±1.2

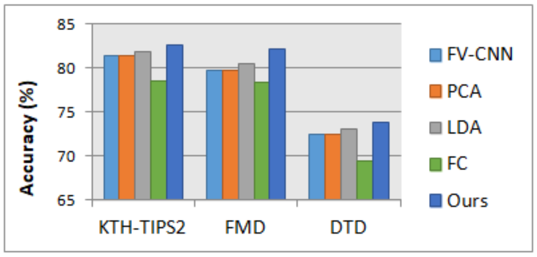

Figure 3. Classification accuracies (%) comparing our LFV method with SVM classification on original FV-CNN descriptor, and the dimension reduced FV-CNN descriptor using PCA, LDA, and the neural network model with a fully connected (FC) layer in place of the locally connected layer.

achieved the highest classification performance on all three datasets. Compared to the current state of the art (FV-CNN and B-CNN), our method provides the larger improvement on the FMD dataset than the KTH-TIPS2 and DTD datasets. We suggest that this difference in improvements could be partly affected by the number of image classes. The hinge loss function would normally better model the differentiation when the number of image classes is small (e.g. 10 classes in FMD).

Also, it is interesting to see that the fine-tuned FV-CNN actually gives lower accuracy than the original FV-CNN. This undesirable effect of fine-tuning could be due to the small number of images available for training. Note that the results given in the table are from backpropagation only to the last convolutional layer. If lower convolutional layers are also fine-tuned, similar or worse results are obtained. In addition, the end-to-end learning of FV descriptors results in the lowest performance. This indicates that when the training data is limited, the generalization capability of GMM is more effective than the supervised learning in CNN in representing the complex feature space. We do however suggest that it might be possible to further enhance the result with the end-to-end learning approach, with more thorough experiments on the design of the training method with data augmentation or multi-scale handling. This is however beyond the scope of this study.

When the FC-CNN descriptor is concatenated with the LFV descriptor, the classification performance is further improved on all three datasets, as shown in Table 2. Recall that this FC-CNN descriptor is the transformed descriptor based on the same LFV model (with different parameters). This result also indicates that our LFV model is not limited to transforming FV descriptors but can be extended to apply to different high-dimensional feature vectors. In addition, our LFV model has a similar number of parameters to the VGG-F model [5]. However, the ImageNet pretrained and fine-tuned VGG-F model provided less than 50% accuracy on texture classification, hence further demonstrating the advantages of using FV-CNN and our LFV descriptors.

Figure 3 shows the various results comparing our LFV method with the other dimensionality reduction techniques. For PCA and LDA, the feature dimension is reduced to the maximum possible dimension when using such techniques. For FC, to restrict the network size, we set the fully con-

Table 3. The classification accuracies (%) of LFV and the compact bilinear pooling (CBP) [9]. The results of CBP are taken from [9], based on two algorithms (RM & TS). Since the CBP method was evaluated using 1/3 of images for training and 2/3 for testing, for fair comparison, we also use this setup here to evaluate LFV. Note that both LFV and CBP have the same feature dimension of 8192.

Dataset	LFV	CBP-RM	CBP-TS
DTD	68.6±1.0	63.2	67.8

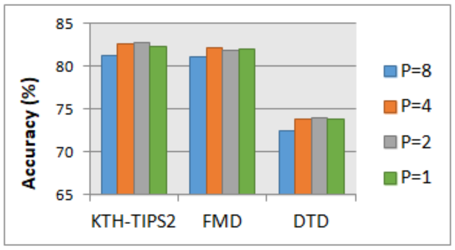

Figure 4. Classification accuracies (%) of our LFV method when different numbers of local filters have shared weights. For example, $P = 4$ is the default setting, meaning every four consecutive filters have the same weights.

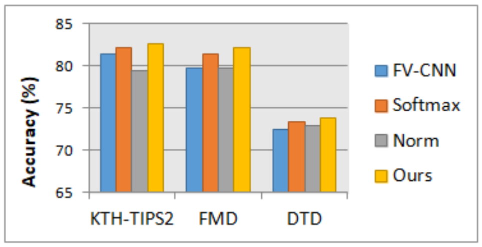

Figure 5. Classification accuracies (%) comparing our LFV method with using softmax as the loss layer, and performing intra-normalization on the FV-CNN descriptor.

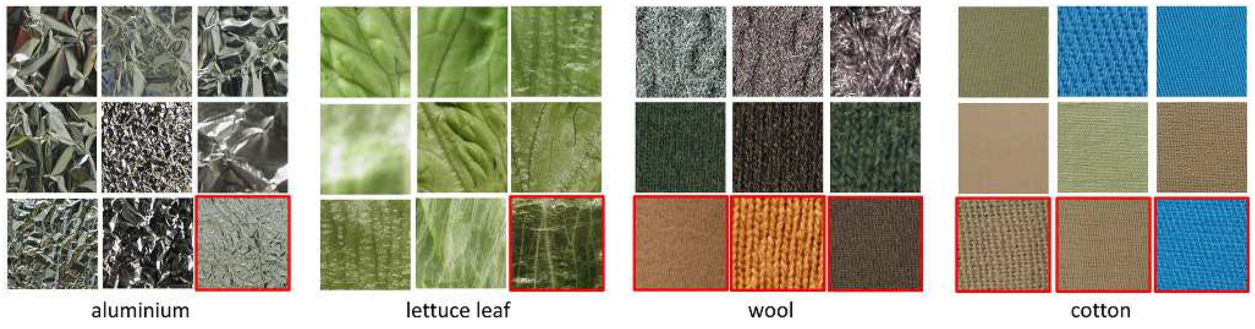

Figure 6. Example images from the KTH-TIPS2 dataset. With our LFV method, the ‘aluminium’ and ‘lettuce leaf’ image classes are the best classified classes (around 99.1% recall), while the ‘wool’ and ‘cotton’ classes are worst classified (around 25.9% and 40.1% recall, respectively). The red border indicates images that are misclassified.

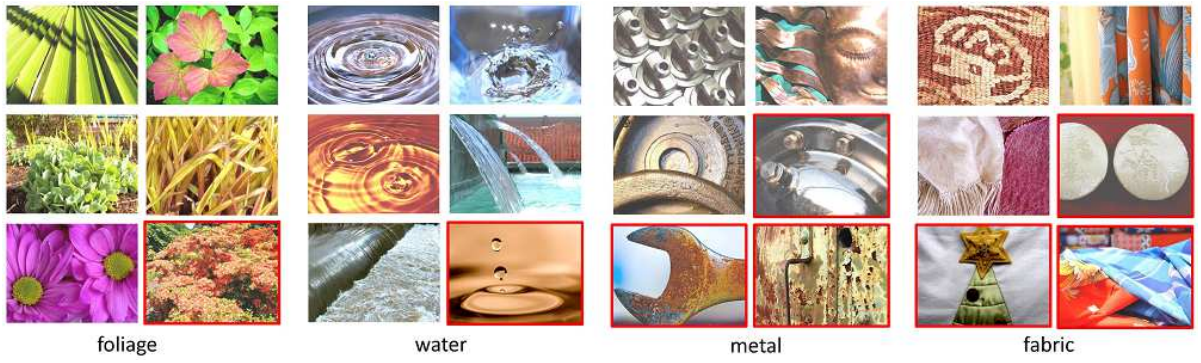

Figure 7. Example images from the FMD dataset. With our LFV method, the ‘foliage’ and ‘water’ image classes are the best classified classes (around 96% and 94% recall, respectively), while the ‘metal’ and ‘fabric’ classes are worst classified (around 64% and 72% recall, respectively). The red border indicates images that are misclassified.

nected layer to have 1024 neurons. The results show that PCA does not affect the classification performance, indicating that there is indeed a large degree of redundancy in the FV-CNN descriptor that could be effectively removed. It is interesting that LDA results in some improvement in the classification performance, hence LDA could be a better alternative than SVM for classifying the FV-CNN descriptors. The FC approach gives the lowest classification accuracy, demonstrating the necessity of using the locally connected layer instead of fully connected layer when transforming the descriptors. In addition, recently a compact bilinear pooling (CBP) method [9] was proposed to reduce the dimension of the B-CNN feature. The method includes two similar algorithms, RM and TS, and the results on the DTD dataset are reported. The two CBP algorithms and our LFV method all reduce the feature dimension to 8192. Our evaluation shows that our LFV method outperforms CBP, as shown in Table 3. These results demonstrate that our LFV method can be regarded as an effective discriminative dimensionality reduction algorithm, based on the supervised learning with a multi-layer neural network model.

We note that an important parameter in our method is the number of local filters of shared weights. We denote this number as P. By default, we specify that every four ($P = 4$) consecutive local filters have the same weights. This is mainly to reduce the network size. Figure 4 shows the effect of this P value on the classification performance. The classification result tends to increase slightly when $P = 2$ or $P = 1$ is used. However, the training complexity and time required also increase with smaller P settings. On the other hand, $P = 8$ means too many local filters have shared weights, and the classification result is reduced considerably. Overall, we suggest that $P = 4$ is a well balanced choice when designing the network model.

We also evaluated using the standard softmax function for the loss layer instead of our SVM loss, with an additional fully-connected layer ahead of the softmax layer. As shown in Figure 5, the softmax loss provides on average 0.5% lower accuracy than the SVM loss, indicating the benefit of using an SVM loss function. In addition, we consider that our local transformation of the FV-CNN descriptor is conceptually related to the intra-normalization technique on VLAD [2], since in both approaches the transformation / normalization is performed on individual sections of the

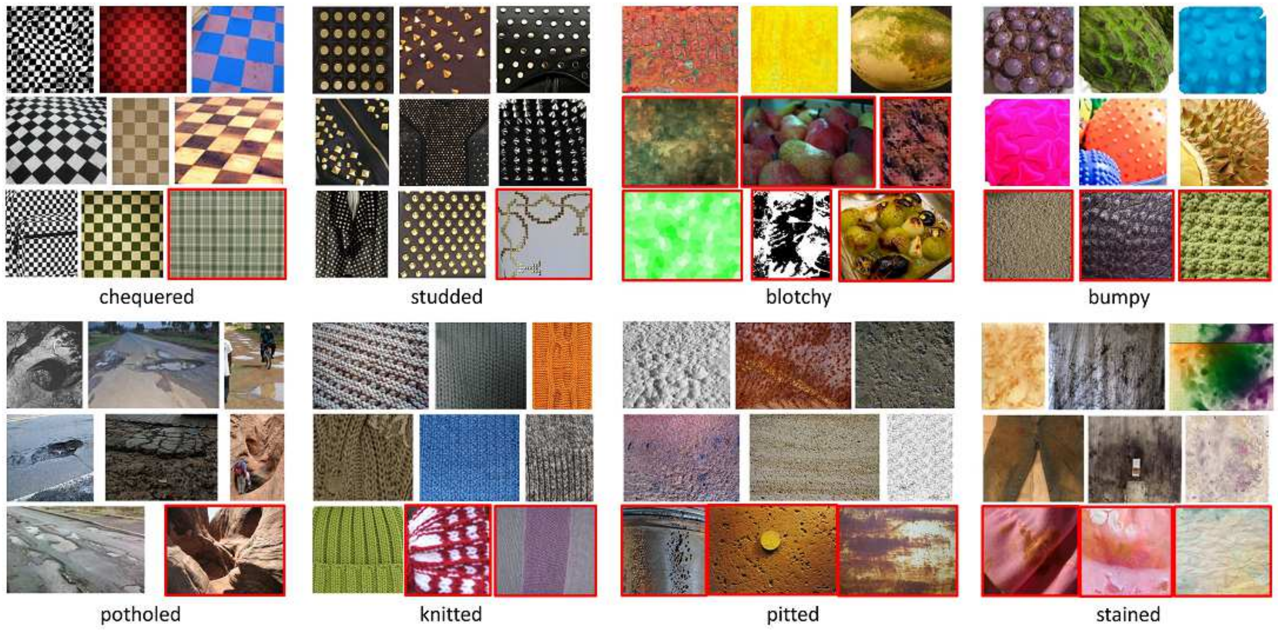

Figure 8. Example images from the DTD dataset. With our LFV method, the 'chequered', 'studded', 'potholed', and 'knitted' image classes are the best classified classes (around 97.5%, 97.5%, 95.0%, and 92.5% recall, respectively), while the 'blotchy', 'bumpy', 'pitted', and 'stained' classes are worst classified (around 35.0%, 47.5%, 50.0%, and 50.0% recall, respectively). The red border indicates images that are misclassified.

descriptor. Therefore, we also evaluated our LFV method against the intra-normalization technique. As shown in Figure 5, compared to the original FV-CNN descriptor, the intra-normalization technique decreases the classification accuracy on the KTH-TIPS2 dataset by about 2% and provides a small improvement on the DTD dataset only, while our LFV method achieves consistent enhancement over FV-CNN on all three datasets. This demonstrates the advantage of having a supervised learning-based transformation rather than a predefined normalization.

Figures 6, 7, and 8 show example images of the classification results. Take the KTH-TIPS2 dataset for example. The aluminium and lettuce leaf classes are visually distinctive from the other classes and hence exhibit excellent classification performance. The lowest classification accuracy was obtained for the wool class, which is often misclassified as cotton or linen classes due to the similar visual characteristics among these fabric classes. For the FMD dataset, it can be seen that although the images in the foliage class also exhibit large visual variation, our method could effectively identify the distinguishing pattern of the leaves and the classification performance for this class is high.

The main computational expensive process is the application of the CNN model to compute the local features at multiple scales, requiring about 2 seconds per image. After the CNN local features are computed, the encoding of Fisher vectors need less than 1 minute for each dataset. Therefore, for a test image at run time, there is little additional cost to compute the FV-CNN descriptor compared to obtain a CNN feature at the last fully connected layer. The training of local filters in LFV needs about 100 epochs on each dataset, and the training time varies depending on the size of the data. For example, on the largest DTD dataset, the training takes about 70 minutes with CPU Core i7 and GPU GeForce GTX 745.

5. Conclusions

We present a texture image classification method in this paper. Our method, called the locally-transferred Fisher vector (LFV), transforms the FV-CNN descriptor in a multi-layer neural network model to obtain a more discriminative feature representation. The LFV model comprises a locally connected layer with filters of locally shared weights and a hinge loss layer representing the SVM classification objective. With the LFV model, the benefits of FV encoding and neural network are integrated in a simple and effective manner, and the resultant LFV descriptor has a lower dimension than the FV-CNN descriptor. Our method is evaluated on three texture image datasets including KTH-TIPS2, FMD, and DTD. The results show that our LFV descriptors provide higher classification performance than the state-of-the-art approaches based on FV-CNN and B-CNN descriptors. We also demonstrate that LFV is more effective than fine-tuning or end-to-end learning of FV-CNN descriptors.

References

[1] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. *CVPR*, pages 5297–5307, 2016. 2

[2] R. Arandjelovic and A. Zisserman. All about VLAD. *CVPR*, pages 1578–1585, 2013. 3, 7

[3] Y. Aytar and A. Zisserman. Tabula rasa: model transfer for object category detection. *ICCV*, pages 2252–2259, 2011. 2

[4] B. Caputo, E. Hayman, and P. Mallikarjuna. Class-specific material categorisation. *ICCV*, pages 1597–1604, 2005. 2

[5] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: delving deep into convolutional nets. *BMVC*, pages 1–12, 2014. 6

[6] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. *CVPR*, pages 3606–3613, 2014. 1, 2

[7] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. *CVPR*, pages 3828–3836, 2015. 1, 2, 3, 5

[8] R. G. Cinbis, J. Verbeek, and C. Schmid. Approximate fisher kernels of non-iid image models for image categorization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(6):1084–1098, 2016. 3

[9] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. *CVPR*, pages 317–326, 2016. 5, 6, 7

[10] L. W. A. v. d. H. C. W. L. Liu, C. Shen. Encoding high dimensional local features by sparse coding based fisher vectors. *NIPS*, pages 1–9, 2014. 3

[11] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1265–1278, 2005. 1

[12] T. Lin and S. Maji. Visualizing and understanding deep texture representations. *CVPR*, pages 2791–2799, 2016. 2, 5

[13] L. Liu, P. Fieguth, G. Kuang, and H. Zha. Sorted random projections for robust texture classification. *ICCV*, pages 391–398, 2011. 1

[14] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *Int. J. Comput. Vis.*, 43(1):7–27, 2001. 1

[15] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. *CVPR*, pages 1–10, 2014. 2

[16] W. Ouyang, X. Wang, C. Zhang, and X. Yang. Factors in finetuning deep model for oject detection with long-tail deistribution. *CVPR*, pages 864–873, 2016. 2

[17] F. Perronnin, J. Sanchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. *ECCV*, pages 143–156, 2010. 1

[18] Y. Quan, Y. Xu, Y. Sun, and Y. Luo. Lacunarity analysis on image patterns for texture classification. *CVPR*, pages 160–167, 2014. 1

[19] L. Sharan, C. Liu, R. Rosenholtz, and E. H. Adelson. Recognizing materials using perceptually inspired features. *Int. J. Comput. Vis.*, 103(3):348–371, 2013. 1

[20] L. Sharan, R. Rosenholtz, and E. H. Adelson. Material perception: what can you see in a brief glance? *Journal of Vision*, 9(8):784, 2009. 2

[21] G. Sharma, S. ul Hussain, and F. Jurie. Local higher-order statistics (lhs) for texture categorization and facial analysis. *ECCV*, pages 1–12, 2012. 1

[22] K. Simonyan and A. V. an A. Zisserman. Deep fisher networks for large-scale image classification. *NIPS*, pages 163–171, 2013. 2

[23] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher vector faces in the wild. *In: BMVC,*, pages 1–12, 2013. 2

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR, arXiv:1409.1556*, 2015. 1

[25] Y. Song, W. Cai, Q. Li, F. Zhang, D. Feng, and H. Huang. Fusing subcategory probabilities for texture classification. *CVPR*, pages 4409–4417, 2015. 1

[26] Y. Song, Q. Li, D. Feng, J. Zou, and W. Cai. Texture image classification with discriminative neural networks. *Computational Visual Media*, 2(4):367–377, 2016. 1

[27] V. Sydorov, M. Sakurada, and C. H. Lampert. Deep fisher kernels - end to end learning of the fisher kernel GMM parameters. *CVPR*, pages 1402–1409, 2014. 2

[28] P. Tang, X. Wang, B. Shi, X. Bai, W. Liu, and Z. Tu. Deep FisherNet for object classification. *arXiv:1608.00182*, 2016. 2, 5

[29] Y. Tang. Deep learning with linear support vector machines. *ICML Workshop*, pages 1–6, 2013. 4

[30] R. Timofte and L. J. V. Gool. A training-free classification framework for textures, writers, and materials. *BMVC*, pages 1–12, 2012. 1

[31] A. Vedaldi and B. Fulkerson. Vlfeat: an open and portable library of computer vision algorithms. *ACM MM*, pages 1469–1472, 2010. 5

[32] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. *ACM MM*, pages 589–692, 2015. 5

[33] Z. Wang, H. Li, W. Ouyang, and X. Wang. Learnable histogram: statistical context features for deep neural networks. *ECCV*, pages 246–262, 2016. 2

[34] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories. *Int. J. Comput. Vis.*, 73(2):213–238, 2007. 1