Deep Texture Recognition via Exploiting Cross-Layer Statistical Self-Similarity*†‡

Zhile Chen[1,†], Feng Li[1,†], Yuhui Quan[1,*], Yong Xu[1] and Hui Ji[2]
[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
[2]Department of Mathematics, National University of Singapore, 119076, Singapore
cszhilechen@gmail.com, csfengli@mail.scut.edu.cn, csyhquan@scut.edu.cn, yxu@scut.edu.cn and matjh@nus.edu.sg

Abstract

In recent years, convolutional neural networks (CNNs) have become a prominent tool for texture recognition. The key of existing CNN-based approaches is aggregating the convolutional features into a robust yet discriminative description. This paper presents a novel feature aggregation module called CLASS (Cross-Layer Aggregation of Statistical Self-similarity) for texture recognition. We model the CNN feature maps across different layers, as a dynamic process which carries the statistical self-similarity (SSS), one well-known property of texture, from input image along the network depth dimension. The CLASS module characterizes the cross-layer SSS using a soft histogram of local differential box-counting dimensions of cross-layer features. The resulting descriptor encodes both cross-layer dynamics and local SSS of input image, providing additional discrimination over the often-used global average pooling. Integrating CLASS into a ResNet backbone, we develop CLASSNet, an effective deep model for texture recognition, which shows state-of-the-art performance in the experiments.

1. Introduction

Texture recognition is an important yet challenging problem in computer vision, with a broad spectrum of applications such as material classification [3, 6, 34], terrain recognition [42] and microscopic image analysis [23]. Its importance comes from the ubiquitousness of texture in our visual world as well as from the primal visual cue provided by texture. One main challenge in texture recognition arises from the various yet contradicting characteristics of textures [12], *e.g.*, uniformity/deformability and regularity/randomness. Such variable internal properties, together with the external distortions from environments, lead to large variability in texture images which is difficult to resolve.

In recent years, deep learning with convolutional neural networks (CNNs) has emerged as a universal approach for texture recognition; see *e.g.* [8, 15, 23, 28, 42, 44–46]. These approaches address the local variability and distortion of texture by leveraging CNNs for learning effective image features. However, as demonstrated in [8, 46], typical CNNs with fully-connected (FC) layers are not a good choice for texture recognition. The reason is, convolutional feature maps are spatially indexed and the FC layer acting like a spatial transform does not remove the correlation to spatial coordinates from its output. As a result, the output of FC layers may be sensitive to the the transforms in spatial domain, one main source of variability of textures.

In other words, CNN-based texture recognition requires a feature aggregation module that can generate a distinct description from convolutional features which is robust to spatial transforms. While the robustness to spatial arrangement can be easily achieved by simply accumulating spatial features, *e.g.* global average pooling (GAP) [1, 10, 45], the question is how to ensure and improve the discrimination during aggregation. Recently, several feature aggregation schemes for this purpose have been proposed; *e.g.* Fisher vector [8] and feature encoding [42, 46]. They aggregate features based on certain statics of a feature tensor.

In this work, we propose a novel yet effective aggregation module, named *CLASS (Cross-Layer Aggregation of Statistical Self-similarity)*, for CNN-based texture recognition. It differs from existing work in two aspects: utilization of cross-layer statistics and explicit exploitation of *statistical self-similarity (SSS)*.

1.1. Motivations and Main Idea

Cross-layer statistics A CNN builds up a hierarchical representation of an image based on a series of convolutional layers. The feature maps from one layer to the next encode texture structures from a smaller to a larger scale. If we treat the generation of feature maps of a texture image along CNN layers as a dynamic evolution process, its characteris-

*Corresponding author: Yuhui Quan.
†Z. Chen and F. Li carried out the experiments and contributed equally.
‡This work was supported in part by National Nature Science Foundation of China under Grants 61872151 and 62072188, in part by CCF-Tencent Open Fund 2020, in part by Science and Technology Program of Guangdong Province under Grant 2019A050510010.

tics of dynamics can provide useful clues for texture recognition. It is shown in [25] that the evolution rule of texture structure across scales is useful for recognition. However, characterizing cross-layer dynamics of texture features is non-trivial, *e.g*. which statistical quantities to use is a question, and there is little related work on it. This inspired us to investigate the exploitation of cross-layer statistics for CNN-based texture recognition.

SSS in texture While texture contains different yet contradicting properties, one consensus has been reached that texture can be well modeled by a stochastic process with statistical stationarity, and a texture image is a realization of such a process with external distortions. The statistical stationarity implies that each region on a texture image has similar values in terms of certain statistics. This property relates directly to SSS [26]: the patterns at different scales, although not identical, are represented by the same statistics. In the past, SSS has demonstrated its effectiveness in characterizing textures, with applications to texture recognition, analysis and synthesis; see *e.g*. [2,30,36,38,40,41,47]. Despite its importance, SSS has not been explicitly utilized in existing CNN-based texture recognition approaches.

Cross-layer SSS Wavelet transforms are a prominent tool for exploiting the SSS of images. Many studies showed that the SSS of an image is carried along and well expressed in the wavelet representation [17, 37, 39], which exists not only over space but also across scales. Indeed, CNNs have deep relationships to wavelet transforms. For instance, the hierarchical convolutional feature maps in a CNN can be viewed as a generalization of the multi-scale representation in the wavelet domain [10]. Bruna and Mallat [4] showed that the wavelet scattering transform equals to an un-trained CNN. Also, the max pooling is similar to taking local maximums in wavelet leader representation [39]. Therefore, if we treat a CNN as a counterpart of wavelet transform, SSS is likely to be carried from image domain along the feature maps in the CNN. In other words, SSS occurs both spatially and across layers in the CNN.

Inspired by above, we model the feature maps of a well-learned CNN to have cross-layer SSS and construct the CLASS module to exploit it for aggregation. Our basic idea is illustrated in Fig. 1. The feature maps selected from different CNN layers are stacked as a feature tensor in order. The cross-layer SSS is actually the one in the tensor along the channel dimension. To exploit it locally, a sliding window is used to sample spatially-local and through-channel blocks from the tensor. On each sampled block, we calculate the so-called differential box-counting (DBC) dimension [33], a well-established quantity in fractal geometry for characterizing SSS. The DBC views a feature tensor as a hyper-surface and examines the number of boxes required to cover the surface over different box scales. Then, the histogram of the DBC dimensions on all blocks is used as the descriptor. See Sec. 4.2 for details.

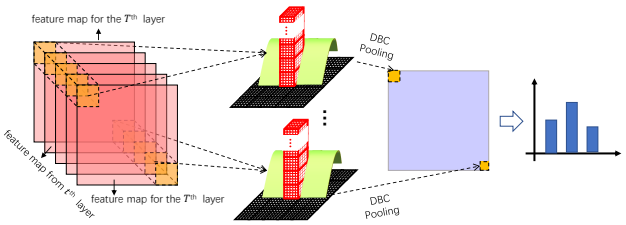


Figure 1. Illustration of basic idea of CLASS.

1.2. Contributions

Integrating the CLASS module into a ResNet backbone, we propose *CLASSNet*, an effective deep network for texture recognition. Its effectiveness is demonstrated by extensive experiments. To summarize, our main contributions in this work are as follows.

Exploiting cross-layer statistics in feature aggregation Different from most existing approaches (*e.g*. [1, 10, 15]) which aggregate convolutional features inside individual layers, we propose to perform feature aggregation using a cross-layer manner. This allows exploiting additional information ignored by inner-layer feature aggregation. There are some approaches (*e.g*. [44,45]) that merge feature maps from different layers into a new one, on which feature aggregation is performed. Different from these approaches, the CLASS module directly calculates statistical quantities across layers. Our work thus can inspire further studies on cross-layer analysis for other image classification tasks.

Incorporating SSS and deep representations We exploit SSS for feature aggregation in CNN-based texture recognition. While SSS is undoubtedly an essential property for texture, it has not been explicitly exploited in existing deep-learning-based approaches. In this paper, we show that the measurement on SSS provides an effective tool for improving feature aggregation. This can inspire future studies on combining SSS and deep CNNs for texture-related tasks.

Friendly SSS-based pooling The calculation of DBC dimensions and related computations in CLASS involve several complicated operations. We provide an efficient implementation for it, which enables the module to be painlessly ported onto and jointly trained with the backbone CNN in an end-to-end manner (unlike [8]). This allows pre-trained backbones to be transferred conveniently and effectively via fine-tuning. In addition, like some recent pooling modules, CLASS outputs a fixed-size descriptor for arbitrary-size input, allowing CLASSNet to handle varying image sizes.

State-of-the-art (SOTA) performance Benefiting from the effectiveness of the CLASS module, our CLASSNet achieved SOTA results on several benchmark datasets.

2. Related Work

Texture recognition is a long-standing problem and there have been numerous studies on it. Interested readers are referred to [23] for a comprehensive survey. In the following, we selectively review the works related to ours.

Handcrafted features for texture recognition In last two decades, a number of traditional approaches model texture by the global distribution of local primitives. Histogram of textons (*e.g*. [6, 11, 20, 32, 48, 49]) and BoVW of texture (*e.g*. [18, 19]) are two representative frameworks along this line. Most of these approaches are concentrated on designing local feature descriptors with robustness for local primitive extraction, *e.g*. LBP [11, 24, 27], RIFT [18, 19], etc. Then the global distribution is mainly encoded by the count of primitive occurrences, such as histogram.

There are some studies exploring other global descriptors. The VLAD [16] aggregates 1st-order statistics on the accumulated differences between a local descriptor and its correspondences. The FV [29] models the distribution of local descriptors by mixture of Gaussian and encodes both 1st- and 2nd-order statistics on it. The MFS [38] employs statistical quantities from fractal geometry.

CNN-based texture recognition There is an increasing number of approaches leveraging CNNs as a powerful feature extractor for texture recognition. One pioneering work can be traced back to Bruna and Mallat [4]. They proposed the scattering transform implemented by a CNN for robust texture classification. While enjoying invariance to certain spatial transforms, their CNN is not learned but with fixed weights, which cannot leverage the power of deep learning.

The seminal work using a learned CNN can be traced back to Cimpoi *et al*. [8]. They demonstrated that a vanilla CNN architecture with FC layers is ineffective for texture recognition, as the CNN's output is highly correlated to the spatial order of pixels. Therefore, they applied FV [29] for encoding the convolutional features on a pre-trained CNN. Song *et al*. [35] attached learnable locally-connected layers to the output of FV-CNN for feature refinement. Owing to the complexity of FV, the weights of the pre-trained backbone CNN in these two approaches cannot be fine-tuned on texture data for improvement.

To enable fine-tuning, Andrearczyk *et al*. [1] employed a simple GAP layer which averages the feature map of each channel to obtain a spatially-orderless description. The GAP is also used by Fujieda *et al*. [10]. They proposed to improve the model expressibility by generalizing intermediate CNN layers to perform wavelet-like spectral analysis. The GAP may discard significant details as it simply accumulates spatial elements. To address this, Lin *et al*. [21] applied the bi-linear pooling [22] which captures the 2nd-order relationship among channels. It calculates the Kronecker product between a feature tensor and itself and averages the results over spatial locations. Dai *et al*. [9] proposed to combine bi-linear pooling with GAP via concatenation for improving discrimination.

The aforementioned pooling modules are non-learnable. The DeepTEN proposed by Zhang *et al*. [46] ports the dictionary learning and residual encoding pipeline on top of convolutional layers, which allows learning inherent visual vocabularies together with the CNN for adaptive pooling. Bu *et al*. [5] proposed a locality-aware coding layer which performs dictionary learning and feature encoding on convolutional features with considerations on their locality constraints. Xue *et al*. [42] combined DeepTEN and GAP, by which local appearance and global context are simultaneously captured. The combination is done by applying bi-linear pooling [22] to the features pooled from DeepTEN and GAP. In their another work [43], differential angular images are taken into account as additional input. Instead of encoding a single convolutional layer, Hu *et al*. [15] proposed to perform feature aggregation on different convolutional blocks individually and fuse the results by an FC layer. Their aggregation module is similar to Xue *et al*. [42].

Zhai *et al*. [45] proposed to learn visual attributes for texture recognition. They constructed a model called MAP-Net which uses a multi-branch architecture to progressively learn visual texture attributes in a mutually reinforced manner. A spatially-adaptive GAP is applied on each branch for feature aggregation. In their later work [44], they proposed a model called DSRNet with a dependency learning module, which exploits the spatial dependency among texture primitives for capturing structural information of texture.

SSS-based texture recognition There are many approaches recognizing textures based on SSS. Most of them apply fractal analysis to characterizing SSS on input images. Xu *et al*. [40] calculated multi-fractal spectra on image intensities and gradients. Varma *et al*. [36] estimated fractal dimensions on image patches for local description. Wendt *et al*. [38] applied multi-fractal analysis to wavelet leader representations. Quan *et al*. [31] applied lacunarity analysis on LBPs. Badri *et al*. [2] combined multi-fractal spectrum and scattering transform [4]. All these approaches are not based on deep learning.

3. DBC for SSS Characterization

The SSS implies that the patterns at different scales can be represented by the same statistics. The DBC dimension [33] in fractal geometry is originally proposed for characterizing the SSS in a 2D gray-scale image. Its basic idea is viewing a gray-scale image as a 3D plane and examining the number of 3D boxes required to cover the surface over different box scales. See Fig. 2 for an illustration.

A gray-scale image $\boldsymbol{I} \in \mathbb{R}^{M \times M}$ is represented as a 3D plane $z = I(x, y)$. The xy-plane is partitioned into non-overlapping grids of size of $s \times s$ pixels, where $s \in \mathbb{Z}_+$

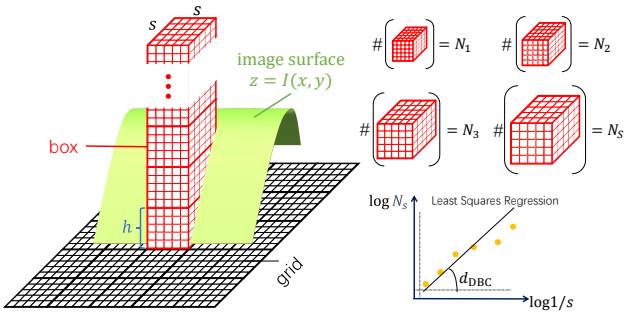


Figure 2. Illustration of DBC method.

varies from 1 to S (zero padding on $\boldsymbol{I}$ is used to ensure $M/s \in \mathbb{Z}$). Let $g_{\max}(i,j)$ and $g_{\min}(i,j)$ denote the maximum and minimum gray levels on the (i,j)th grid respectively. Let $n_s(i,j)$ be the number of boxes of size $s \times s \times h_s$ required to be placed one after another on the (i,j)th grid to fill the image surface (*i.e.* gray-level variations) of that grid, where $h_s = sI_{\max}/M$ is the box height and $I_{\max} = \max_{x,y} \boldsymbol{I}(x,y)$. Let N_s denote the number of boxes required to fill the whole image surface at the box scale s. Then n_s and N_s can be directly calculated by

$$n_s(i,j) = \lceil g_{\max}(i,j)/h_s \rceil - \lceil g_{\min}(i,j)/h_s \rceil + 1, \quad (1)$$

$$N_s = \sum_{i,j} n_s(i,j). \quad (2)$$

Note that $\lceil g_{\max}/h_s \rceil$ and $\lceil g_{\min}/h_s \rceil$ are the indices of the boxes that contain the maximum and minimum gray levels on the corresponding grid respectively. With estimated $N_s, \forall s$, DBC applies linear least-squares regression to fit the points $\{(\log 1/s, \log N_s)\}_s$ into a line. The slope of that line is defined as the DBC dimension of $\boldsymbol{I}$ and calculated by

$$\frac{\sum_{s=1}^{S}(S\log\ N_s - \sum_{s'=1}^{S}\log\ N_{s'})(\sum_{s'=1}^{S}\log s' - S\log s)}{\sum_{s=1}^{S}(S\log s - \sum_{s'=1}^{S}\log s')^2}. \quad (3)$$

DBC for 3D tensors In this work, we adapt DBC to process 3D feature tensors instead of 2D images. The adaption is straightforward. Consider a feature tensor $\mathcal{P}(x,y,z) \in \mathbb{R}^{M\times N\times T}$ of spatial size $M \times N$ and T channels, which is represented by a 4D plane in terms of magnitude. Now the partition is performed on the xyz-plane which results in 3D grids of size $s \times s \times \bar{s}$. Accordingly, the boxes are of size $s \times s \times \bar{s} \times h_s$, and $g_{\max}(x,y,z)$ and $g_{\min}(x,y,z)$ denote the maximum and minimum magnitudes on the (x,y,z)th grid respectively. Then we have

$$n_s(i,j,k) = \lceil \frac{g_{\max}(i,j,k)}{h_s} \rceil - \lceil \frac{g_{\min}(i,j,k)}{h_s} \rceil + 1, \quad (4)$$

$$N_s = \sum_{i,j,k} n_s(i,j,k). \quad (5)$$

Then the DBC dimension is calculated based on (3),(4),(5).

4. CLASSNet

4.1. Architecture

The architecture of CLASSNet is illustrated in Fig. 3, where we use ResNet [14] as the backbone. Briefly, ResNet sequentially connects a series of residual blocks (RBs), a GAP layer and an FC layer. Each RB is mainly composed of several convolutional (Conv) layers and a skip connection. For convenience, we also refer to the 1st Conv layer as the 1st RB. The CLASS module is placed on top of all RBs and connected to the FC layer. It collects the feature tensor output by each RB as input, and aggregates them into a single description. The descriptions generated from CLASS and GAP are then concatenated as the texture representation which is passed to the FC layer for classification. The details on the CLASS module are given in the next.

4.2. The CLASS Module

There are four stages in CLASS: size normalization, cross-layer grouping, DBC pooling, and aggregation,

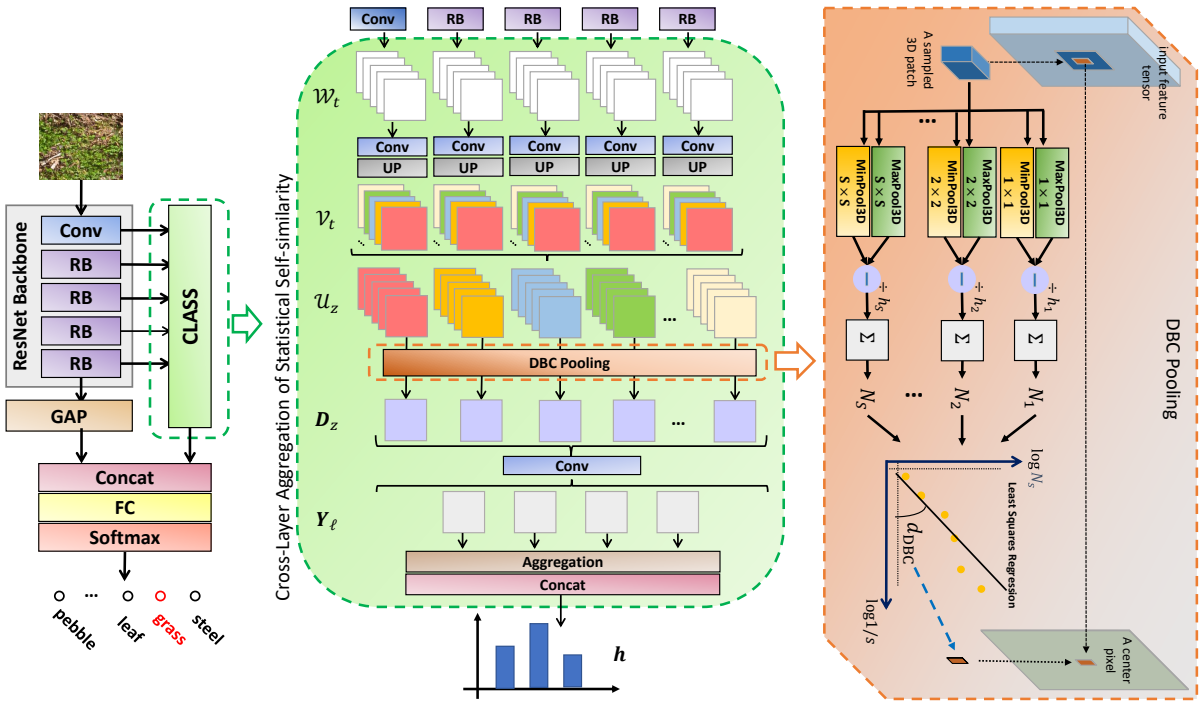
Size normalization Suppose the total number of RBs is T. Let $\mathcal{W}_t, t = 1,\cdots,T \in \mathbb{R}^{M_t\times N_t\times Z_t}$ denote the feature tensor output by the tth RB, with varying spatial size $M_t \times N_t$ and varying channel number Z_t over t. To facilitate subsequent processing, CLASS first normalizes $\{\mathcal{W}_t\}_t$ to the same size. This is done by applying 1×1 convolutions that transform each tensor $\mathcal{W}_t$ to a Z-channel one and then upsampling it to a fixed spatial size $M \times N$ using bilinear interpolation, where $M = \max_t M_t, N = \max_t N_t$. The resulting equal-size feature tensors are denoted by $\mathcal{V}_t \in \mathbb{R}^{M\times N\times Z}, t = 1,\cdots T$.

Cross-layer grouping For the analysis along network layers, we construct Z feature tensors denoted by $\mathcal{U}_z, z = 1,\cdots,Z$, by reorganizing $\mathcal{V}_t$s as follows. Let $\boldsymbol{V}_t^{z'} \in \mathbb{R}^{M\times N}$ represent $\mathcal{V}_t(:,:,z')$ which is the feature map of the z'th channel on the tth RB. The feature tensor $\mathcal{U}_z$ is then defined by

$$\mathcal{U}_z = [\boldsymbol{V}_1^z; \boldsymbol{V}_2^z; \cdots; \boldsymbol{V}_T^z] \in \mathbb{R}^{M\times N\times T},\ \forall z, \quad (6)$$

where $[\cdot;\cdot]$ denotes the concatenation along the 3rd dimension. In other words, we pick up one feature map from each $\mathcal{V}_t$ in order and stack them into a new feature tensor.

DBC pooling Suppose SSS is carried from input image to the intermediate feature maps across different RBs. Then $\mathcal{U}_z$ is modeled as a volumetric representation with SSS, where the SSS presents both spatially and across its channel dimension. Considering the spatial homogeneity of texture, we characterize the cross-layer statistics on spatially-local and through-channel blocks. For this purpose, we resort to the DBC method mentioned in Sec. 3, one effective tool for measuring SSS in image data. Concretely, a sliding window is applied to sampling 3D local patches of size $7\times7\times T$. On



(a) Architecture of CLASS-Net (b) Flowchart of CLASS (c) Details of DBC Pooling

Figure 3. Illustration of architecture of CLASSNet.

each sampled patch we apply the DBC analysis, by which a DBC value is obtained on a local patch. With such operations, the 3D tensor $\mathcal{U}_z$ is transformed to a 2D map denoted by $\boldsymbol{D}_z \in \mathbb{R}^{M\times N}$. Then we have a feature tensor $\mathcal{D} = [\boldsymbol{D}_1; \cdots; \boldsymbol{D}_Z] \in \mathbb{R}^{M\times N\times Z}$ of DBC values.

Concretely, given a sampled 3D patch $\mathcal{P}$, we calculate its DBC dimension using (3),(4),(5). To capture SSS across all RBs, we fix the 3rd dimension of the boxes to $\bar{s} = T$, and vary its 1st and 2nd dimensions: $s \in [1, 4]$. The box height is set to $h_s = s\frac{\max_{x,y,z}\mathcal{P}(x,y,z)}{\min(M,N)}$.

Aggregation Before aggregation, a series of 1×1 convolutions are applied to $\mathcal{D}$ for feature refinement, which generates L refined feature maps: $\boldsymbol{Y}_\ell \in \mathbb{R}^{M\times N}, \ell = 1\cdots, L$ ($L = 1, 4$ for ResNet-18/50). On each $\boldsymbol{Y}_\ell$, we calculate a soft histogram from it as follows. Let $\{\beta_k \in \mathbb{R},\ k = 1, ..., K\}$ denote a set of learnable bin centers ($K = 8$ in our implementation). The soft histogram is calculated via applying softmax to the residual error vector $[(\boldsymbol{Y}(x,y) - \beta_1)^2, \cdots, (\boldsymbol{Y}(x,y) - \beta_K)^2]$:

$$\boldsymbol{h}_0^{(\ell)}(k) = \sum_{i,j}^{M,N} \frac{\exp(-s_k^2(\boldsymbol{Y}(x,y) - \beta_k)^2)}{\sum_{k'=1}^{K}\exp(-s_{k'}^2(\boldsymbol{Y}(x,y) - \beta_{k'})^2)}, \quad (7)$$

for all k, where $\{s_k \in \mathbb{R}\}_{k=1}^K$ is a set of scaling factors learned together with $\{\beta_k\}_k$. It can be seen that the soft histogram counts the contribution of $\boldsymbol{Y}(x,y)$ to the bin of β_k according to their ℓ_2 residual error in a soft manner. For improvement, we also calculate soft histograms $\boldsymbol{h}_1^{(\ell)}, \cdots, \boldsymbol{h}_4^{(\ell)}$ on 4 uniformly-divided regions of $\boldsymbol{Y}_\ell$ and concatenate them as $\boldsymbol{h}^{(\ell)} = [\boldsymbol{h}_0^{(\ell)}, \cdots, \boldsymbol{h}_4^{(\ell)}]$. The bin centers are shared in $\boldsymbol{h}^{(\ell)}$ but individual across different ℓ. Finally the CLASS outputs $\boldsymbol{h} = [\boldsymbol{h}^{(1)}, \cdots, \boldsymbol{h}^{(L)}]$.

4.3. Implementation and Training

At first glance, the CLASS module involves some seemingly complicated operations. However, it can be effectively implemented by the basic operations in deep networks supported by existing deep learning platform, so that the associated CLASSNet can be effectively trained. Recall that the main computations involved in CLASS include (3)∼(7). The implementation of (3),(5),(6) is straightforward. The computation of (4) involves $g_{\max}, g_{\min}$, *i.e.* taking maximum/minimum values across local blocks with different block sizes, which can be implemented by 3D max/min pooling with varying pooling sizes. The computation of (7) can be done using softmax activation.

There are not many additional trainable parameters introduced by CLASS, which just include the weights of the convolutional layers for size normalization and that for generating $\boldsymbol{Y}_\ell$ from $\boldsymbol{D}_z$s in DBC pooling, as well as the learnable bin centers and scaling factors in soft histogram aggregation. These parameters are jointly trained with those of the ResNet backbone, using the KL-divergence loss with a fixed number of epochs.

Figure 4. Samples images from six datasets.

5. Experiments

5.1. Datasets

Six benchmark datasets are used for evaluation, whose characteristics are summarized as follows. *(a)* Ground Terrain in Outdoor Scenes (GTOS) [43] is a dataset of outdoor ground materials with 40 categories, with a training/testing split given. *(b)* GTOS-Mobile [42] is a dataset collected from GTOS via mobile phone, which consists of 100011 material samples from 31 categories. *(c)* Materials in Context 2500 (MINC-2500) [3] is a dataset of 23 material categories, each of which contains 2500 images. It provides five training/test splits. *(d)* KTH-TIPS2b [6] is a dataset composed of 4752 images from 11 material categories. *(e)* Describable Texture Dataset (DTD) [7] contains 47 categories of wild textures, with 120 images per category. It provides 10 preset splits into equally-size training, validation and test sets. *(f)* Flickr Material Dataset (FMD) [34] is composed of 10 different material categories, with 100 images each category. These datasets cover a broad spectrum of textures. See Fig. 4 for some examples.

Our experimental evaluation mainly follows [44,46]. We use the provided splits on GTOS and MINC-2500, and the splits as [44] for DTD. As for KTH-TIPS2b and FMD, each dataset is randomly divided into 10 splits with recommended split size, and the mean accuracy across splits are recorded. We report the result in the form of "$mean \pm s.t.d.\%$". The results on DTD, FMD, KTH-TIPS2b, MINC-2500 and GTOS are based on 5-time statistics, and the results on GTOS-Mobile are averaged over 2 runs.

5.2. Implementation Details

Our model is implemented with PyTorch and run on a single RTX Titan GPU. On GTOS-Mobile, it takes around 0.51 hours per epoch during training and 0.01 seconds to test an image. Following existing work, the ResNet18 and ResNet50 [14] are used as the backbone network respectively. The SGD optimizer with momentum of 0.9 is used for training, and the batch size is set to 32. The learning rate with cosine decay is initialized to 2×10^{-3} for GTOS-Mobile and 1×10^{-3} for other datasets. The training is finished after 30 epochs. The ResNet backbone is initialized with pre-trained models. Other network parameters are initialized by the default Kaiming [13]. Following existing literature (*e.g.* [42, 46]), all images in training and test are resized to 256×256 and then cropped to 224×224. Horizontal flipping with probability 0.5 is applied to input images for data augmentation in training.

5.3. Comparison against State-of-the-arts

We select 11 CNN-based texture recognition approaches in recent years for performance comparison, including FC-CNN [8], FV-CNN [8], BP-CNN [21], LFV [35], FASON [9], DeepTEN [46], DEPNet [42], LSCNet [5], MAPNet [45], DSR [44] and HistNet [28]. The classification results of these approaches on all six datasets are summarized in Tab. 1. The results are quoted from existing literature (mainly from [44]) whenever possible, or left blank otherwise. The best results using ResNet18 and ResNet50 are distinguished by different colors.

Results using ResNet18 backbone Using ResNet18, our CLASSNet performs the best on all the benchmark datasets in terms of mean classification accuracy, while the s.t.d. results are comparable to other models. Particularly, compared to the second best performer, it shows noticeable improvement of more than 3% accuracy on KTH and GTOS, and of around 1.6% accuracy on GTOS-Mobile. Such results have demonstrated the power of CLASSNet.

Results using ResNet50 backbone Overall, the competitiveness of CLASSNet using ResNet50 is not as high as that using ResNet18, but CLASSNet is still very competitive. As the best performer on KTH, FMD, MINC and GTOS, CLASSNet shows noticeable advantages on KTH and MINC and is slightly better than DSR on FMD and GTOS. It performs worse than DSRNet and MAPNet on DTD and GTOS-Mobile. Recall that the feature maps along network depth are converted into a new feature tensor before DBC pooling in our method. The number of

Table 1. Performance comparison of different methods in terms of classification accuracy (%). The best result on each dataset is marked in **red**/**blue** for ResNet18 and ResNet50 respectively.

Method	Source	Backbone	DTD		KTH		FMD		MINC		GTOS		GTOS-Mobile	
			mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
FC-CNN	CVPR15	VGGVD	62.9	0.8	81.8	2.5	77.4	1.8	-	-	-	-	-	-
FV-CNN	CVPR15	VGGVD	72.3	1.0	75.4	1.5	79.8	1.8	-	-	77.1	-	-	-
BP-CNN	CVPR16	VGGVD	69.6	0.7	75.1	2.8	77.8	1.9	-	-	-	-	-	-
BP-CNN	CVPR16	ResNet18	-	-	-	-	-	-	-	-	-	-	75.43	-
LFV	ICCV17	VGGVD	73.8	1.0	82.6	2.6	82.1	1.9	-	-	-	-	-	-
FASON	CVPR17	VGGVD	72.3	0.6	76.5	2.3	-	-	-	-	-	-	-	-
DeepTEN	CVPR17	ResNet18	-	-	-	-	-	-	-	-	-	-	76.12	-
DeepTEN	CVPR17	ResNet50	69.6	-	82.0	3.3	80.2	0.9	81.3	-	84.5	2.9	-	-
DEPNet	CVPR18	ResNet18	-	-	-	-	-	-		-	-	-	82.18	-
DEPNet	CVPR18	ResNet50	73.2	-	-	-	-	-	82.0	-	-	-	-	-
LSCNet	PR19	VGG16	71.1	-	76.9	-	82.4	-	-	-	-	-	-	-
LSCNet	PR19	ResNet18	-	-	-	-	76.3	-	-	-	-	-	-	-
LSCNet	PR19	ResNet50	-	-	-	-	81.2	-	-	-	-	-	-	-
MAPNet	ICCV19	VGGVD	74.1	0.6	82.7	1.5	82.9	0.9	-	-	80.8	2.5	82.00	1.6
MAPNet	ICCV19	ResNet18	69.5	0.8	80.9	1.8	80.8	1.0	-	-	80.3	2.6	82.98	1.6
MAPNet	ICCV19	ResNet50	76.1	0.6	84.5	1.3	85.2	0.7	-	-	84.7	2.2	86.64	1.5
DSRNet	CVPR20	VGGVD	74.9	0.7	83.5	1.5	84.0	0.8	-	-	81.8	2.2	82.94	1.6
DSRNet	CVPR20	ResNet18	71.2	0.7	81.8	1.6	81.3	0.8	-	-	81.0	2.1	83.65	1.5
DSRNet	CVPR20	ResNet50	**77.6**	0.6	85.9	1.3	86.0	0.8	-	-	85.3	2.0	**87.03**	1.5
HistNet	ArXiv20	ResNet18	-	-	-	-	-	-	-	-	-	-	79.75	0.8
HistNet	ArXiv20	ResNet50	72.0	1.2	-	-	-	-	82.4	0.3	-	-	-	-
CLASSNet (Ours)		ResNet18	**71.5**	0.4	**85.4**	1.1	**82.5**	0.7	**80.5**	0.6	**84.3**	2.2	**85.25**	1.3
CLASSNet (Ours)		ResNet50	74.0	0.5	**87.7**	1.3	**86.2**	0.9	**84.0**	0.6	**85.6**	2.2	85.69	1.4

channels in the new tensor is the same for the ResNet18 and ResNet50 backbones. Such a fixed channel number makes our model hard to utilize all benefits brought by the deeper structure of ResNet50 over ResNet18. Thus, the performance improvement on ResNet50 is not as high as ResNet18, but the result remains SOTA.

Overall comparison In comparison to DSRNet, the top performer in other methods, CLASSNet performs better across all datasets when fixing backbones to ResNet18, and it shows advantages on 4/6 datasets when using ResNet50. Such results indicate that the CLASS module works particularly well for lighter-weight backbones, and sees its values particularly in the scenarios where a lighter-weight model is preferred due to limitations of computational resources. Indeed, DSRNet and CLASSNet encode texture from different aspects: spatial dependency vs. cross-layer SSS, which can be combined. Three closely-related methods to ours are DeepTEN, DEPNet and HistNet. They all exploit histograms of local features but without considering cross-layer statistics. The superior performance of CLASSNet to them indicates the benefits of CLASS.

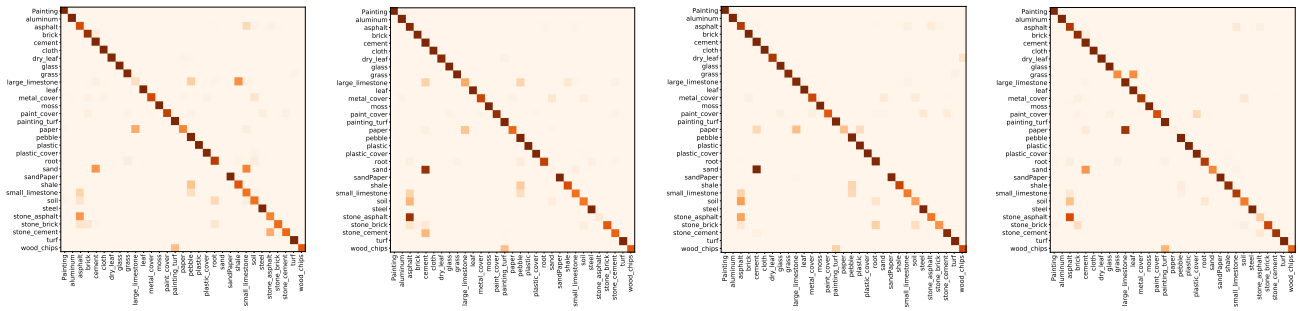
Model complexity comparison In addition to classification accuracy, we also compare the methods in terms of *(a)* model size measured by the number of model parameters; and *(b)* efficiency measured by the floating-point operations per second of the model. See Tab. 2 for the comparison with DeepTEN and DEPNet. The complexity of our model is comparable to others in terms of both criteria.

Table 2. Complexity comparison of different models in terms of number of model parameters (#Params) and floating-point operations per second (FLOPs).

Model	#Params (≈, M)		FLOPs (≈, G)	
	ResNet18	ResNet50	RestNet18	ResNet50
Backbone	11.19	23.57	1.82	4.11
DeepTEN	11.37	23.90	1.82	4.12
DEPNet	12.01	25.56	1.82	4.11
CLASSNet	11.23	23.70	1.83	4.14

5.4. Ablation Studies

We conduct the following ablation studies for analyzing the effectiveness of CLASS in CLASSNet. We generate a baseline model from CLASSNet by removing its CLASS module, which is denoted by 'w/o CLASS'. To further verify the effectiveness of the SSS-based statistics, we replace the DBC dimension calculation step on each sampled 3D local block, with a simple global averaging pooling operation. In other words, we only use mean for the cross-layer statistics, rather than the DBC-based SSS statistics. The resulting baseline model is denoted by 'DBC→Mean'. These two baseline models are trained with the same scheme as ours. Their results using ResNet18 backbone on GTOS-Mobile and MINC are listed in Tab. 3 for comparison.



(a) 'w/o CLASS' (ResNet18) (b) 'w/o CLASS' (ResNet50) (c) CLASSNet (ResNet18) (d) CLASSNet (ResNet50)

Figure 5. Confusion matrices of several models on GTOS-Mobile.

It can be seen that the decrease of performance caused by the removal of CLASS module is noticeable on both datasets. This suggests that the cross-layer SSS encoded by our CLASS module does provide additional discriminability to the CNN-based texture representation. We can also see that 'DBC→Mean' performs moderately better than 'w/o CLASS'. This result indicates that cross-layer statistics do benefit texture recognition. However, its results are still worse than original CLASSNet. The reason is probably that, the simple mean cannot capture essential properties of texture and thus the resulting texture representation is insufficiently discriminative. In comparison, CLASSNet employs DBC-based statistics to characterize SSS, one essential property of texture, leading to better results.

Table 3. Performance comparison of CLASSNet and baselines in terms of classification accuracy (%) in ablation studies. ResNet18 is used as the backbone.

Dataset	w/o CLASS	DBC →Mean	CLASSNet
GTOS-Mobile	81.91 ± 1.5	83.43 ± 2.0	85.25 ± 1.3
MINC-2500	77.98 ± 0.5	78.63 ± 0.4	80.50 ± 0.6

5.5. Confusion Analysis

To examine the behavior of CLASSNet on individual categories, we calculate its confusion matrices on GTOS-Mobie using ResNet18 and ResNet50 backbones respectively, and show them in Fig. 5. We also include the baseline 'w/o CLASS' for comparison. It can be seen that the confusion matrices of CLASSNet are more diagonally concentrated and cleaner than those of 'w/o CLASS'. Many confusing categories for 'w/o CLASS' can be well distinguished in CLASSNet. This suggests that CLASSNet can generate texture descriptions with stronger robustness and higher discrimination. Comparing the two confusion matrices of CLASSNet, we can see that CLASSNet with ResNet50 shows slight improvement on some confusing categories for CLASSNet with ResNet18, *e.g.* Sand versus Cement; and it also brings more confusion on some classes, *e.g.* Paper versus Large Limestone. Some confusing cases for CLASSNet with ResNet18 are shown in Fig. 6, where each confusing image pair has quite similar appearance and their SSS is similar as well. It is not surprising that CLASSNet is not good at handing these cases.

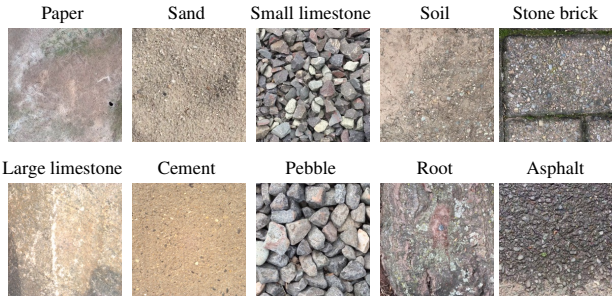


Figure 6. Confusing cases of CLASSNet on GTOS-Mobile. Top: samples from the class c^* of the worst accuracy. Bottom: samples from the class where most samples in c^* are incorrectly predicted.

5.6. More Results

See supplementary materials for additional results.

6. Summary

The special characteristics of texture differ from those of general visual data and make texture recognition become a fundamental, longstanding yet challenging problem in computer vision. This paper proposed an effective CNN-based approach that exploits cross-layer SSS for texture recognition. It exploited the cross-layer dynamics of feature maps and made use of SSS-related statistics for improving the discrimination in aggregating CNN features. The characterization of cross-layer SSS is done by the CLASS module which calculates DBC-based descriptions on cross-layer feature maps. Since cross-layer SSS helps to reveal the underlying process of texture, the texture representation generated by the CLASS module is effective for classification. Equipped with CLASS, the CLASSNet achieved SOTA results in the experiments. Cross-layer SSS may benefit other applications, and it will be studied in our future work.

References

[1] Vincent Andrearczyk and Paul F. Whelan. Using filter banks in convolutional neural networks for texture classification. *Pattern Recognit. LETT*, 84:63–69, 2016. 1, 2, 3

[2] Hicham Badri, Hussein Yahia, and Khalid Daoudi. Fast and accurate texture recognition with multilayer convolution and multifractal analysis. In *Proc. ECCV*, pages 505–519. Springer International Publishing, 2014. 2, 3

[3] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the materials in context database. In *Proc. CVPR*, 2015. 1, 6

[4] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE Trans. Pattern Anal. Mach. Intell*, 35:1872–1886, 08 2013. 2, 3

[5] Xingyuan Bu, Yuwei Wu, Zhi Gao, and Yunde Jia. Deep convolutional network with locality and sparsity constraints for texture classification. *Pattern Recognit.*, 91:34–46, July 2019. 3, 6

[6] Barbara Caputo, Eric Hayman, and P. Mallikarjuna. Class-specific material categorisation. In *Proc. ICCV*, volume 2, pages 1597 – 1604 Vol. 2, 2005. 1, 3, 6

[7] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proc. CVPR*, pages 3606–3613, 2014. 6

[8] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In *Proc. CVPR*, pages 3828–3836, 2015. 1, 2, 3, 6

[9] X. Dai, J. Y. Ng, and L. S. Davis. Fason: First and second order information fusion network for texture recognition. In *Proc. CVPR*, pages 6100–6108, 2017. 3, 6

[10] Shin Fujieda, Kohei Takayama, and Toshiya Hachisuka. Wavelet convolutional neural networks for texture classification. *Proc. CVPR*, Jul 2017. 1, 2, 3

[11] Zhenhua Guo, Lei Zhang, and David Zhang. A completed modeling of local binary pattern operator for texture classification. *IEEE Trans. Image Process.*, 19(6):1657–1663, 2010. 3

[12] R. M. Haralick. Statistical and structural approaches to texture. *Proc. IEEE*, 67(5):786–804, 1979. 1

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. ICCV*, pages 1026–1034, 2015. 6

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016. 4, 6

[15] Yuting Hu, Zhiling Long, and Ghassan AlRegib. Multi-level texture encoding and representation (multer) based on deep neural networks. In *Proc. ICIP*, 2019. 1, 2, 3

[16] Herve Jegou, Matthijs Douze, Cordelia Schmid, and Patrick Perez. Aggregating local descriptors into a compact image representation. In *Proc. CVPR*, 2010. 3

[17] Hui Ji, Xiong Yang, Haibin Ling, and Yong Xu. Wavelet domain multifractal analysis for static and dynamic texture classification. *IEEE Trans. Image Process.*, 22, Aug 2012. 2

[18] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *IEEE Trans. Pattern Anal. Mach. Intell*, 27(8):1265–1278, 2005. 3

[19] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A discriminative framework for texture and object recognition using local image features. In *Toward Category-Level Object Recognition*, pages 423–442. Springer, 2006. 3

[20] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Comput. Vision*, 43:29–44, 06 2001. 3

[21] Tsung-Yu Lin and Subhransu Maji. Visualizing and understanding deep texture representations. In *Proc. CVPR*, pages 2791–2799, 2016. 3, 6

[22] Tsung Yu Lin, Aruni Roychowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proc. ICCV*, 2015. 3

[23] Li Liu, Jie Chen, Paul Fieguth, Guoying Zhao, Rama Chellappa, and Matti Pietikäinen. From bow to cnn: Two decades of texture representation for texture classification. *Int. J. Comput. Vision*, (Nov):1–36, 2018. 1, 3

[24] Li Liu, Paul Fieguth, Yulan Guo, Xiaogang Wang, and Matti Pietikäinen. Local binary features for texture classification: Taxonomy and experimental study. *Pattern Recognit.*, 62:135–160, Feb. 2017. 3

[25] Topi Mäenpää and Matti Pietikäinen. Multi-scale binary patterns for texture analysis. In Josef Bigun and Tomas Gustavsson, editors, *Image Anal.*, pages 885–892, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 2

[26] Mandelbrot and B. How long is the coast of britain? statistical self-similarity and fractional dimension. *Science*, 156(3775):636–638, 1967. 2

[27] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognit.*, 29(1):51–59, 1996. 3

[28] Joshua Peeples, Weihuang Xu, and Alina Zare. Histogram layers for texture analysis, 2020. 1, 6

[29] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proc. CVPR*, pages 1–8, 2007. 3

[30] Yuhui Quan, Yong Xu, and Yuping Sun. A distinct and compact texture descriptor. *Image Vision Comput.*, 32, Apr 2014. 2

[31] Yuhui Quan, Xu Yong, Yuping Sun, and Luo Yu. Lacunarity analysis on image patterns for texture classification. In *Proc. CVPR*, 2014. 3

[32] Payam Saisan, Gianfranco Doretto, Ying Nian Wu, and Stefano Soatto. Dynamic texture recognition. In *Proc. CVPR*, volume 2, pages II–II. IEEE, 2001. 3

[33] N. Sarkar and B. B. Chaudhuri. An efficient differential box-counting approach to compute fractal dimension of image. *IEEE Trans.Syst. Man Cyber.*, 24(1):115–120, 1994. 2, 3

[34] Lavanya Sharan, Ruth Rosenholtz, and EH Adelson. Material perception: What can you see in a brief glance? *J. VIsion*, 9:784–784, Aug 2010. 1, 6

[35] Yang Song, Fan Zhang, Qing Li, Heng Huang, and Weidong Cai. Locally-transferred fisher vectors for texture classification. In *Proc. ICCV*, 2017. 3, 6

[36] Manik Varma and Rahul Garg. Locally invariant fractal features for statistical texture classification. In *Proc. ICCV*, 2007. 2, 3

[37] M. Vergassola and U. Frisch. Wavelet transforms of self-similar processes. *Physica D: Nonlinear Phenomena*, 54(1):58 – 64, 1991. 2

[38] Herwig Wendt, Patrice Abry, Stéphane Jaffard, Hui Ji, and Zuowei Shen. Wavelet leader multifractal analysis for texture classification. In *Proc. ICIP*, pages 3829–3832, 11 2009. 2, 3

[39] Herwig Wendt, Stéphane G. Roux, Stéphane Jaffard, and Patrice Abry. Wavelet leaders and bootstrap for multifractal analysis of images. *Signal Process.*, 89(6):1100 – 1114, 2009. 2

[40] Yong Xu, Hui Ji, and Cornelia Fermüller. Viewpoint invariant texture description using fractal analysis. *Int. J. Comput. Vision*, 83(1):85–100, Jun 2009. 2, 3

[41] Y. Xu, Y. Quan, H. Ling, and H. Ji. Dynamic texture classification using dynamic fractal analysis. In *Proc. ICCV*, pages 1219–1226, 2011. 2

[42] Jia Xue, Hang Zhang, and Kristin Dana. Deep texture manifold for ground terrain recognition. In *Proc. CVPR*, pages 558–567, 2018. 1, 3, 6

[43] Jia Xue, Hang Zhang, Kristin Dana, and Ko Nishino. Differential angular imaging for material recognition. In *Proc. CVPR*, pages 764–773, 2017. 3, 6

[44] Wei Zhai, Yang Cao, Zheng-Jun Zha, HaiYong Xie, and Feng Wu. Deep structure-revealed network for texture recognition. In *Proc. CVPR*, pages 11007–11016. IEEE, 2020. 1, 2, 3, 6

[45] Wei Zhai, Yang Cao, Jing Zhang, and Zheng-Jun Zha. Deep multiple-attribute-perceived network for real-world texture recognition. In *Proc. ICCV*, pages 3612–3621. IEEE, 2019. 1, 2, 3, 6

[46] Hang Zhang, Jia Xue, and Kristin Dana. Deep ten: Texture encoding network. In *Proc. CVPR*, pages 2896–2905, 2017. 1, 3, 6

[47] Dongxiao Zhou. Texture analysis and synthesis using a generic markov-gibbs image model. 01 2006. 2

[48] Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *Int. J. Comput. Vision*, 27(2):107–126, 1998. 3

[49] Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural Comput.*, 9(8):1627–1660, 1997. 3

Deep Texture Recognition via Exploiting Cross-Layer Statistical Self-Similarity [Supplementary Materials]

1. Parameter influence analysis

The experiments on how the hyper-parameter setting influences the performance of the proposed method are conducted as follows. We consider two hyper-parameters. (i) The channel number of $\mathcal{V}_t$, *i.e.* Z. In our classification experiment, we set $Z = 16$. In the influence test, we try another two values: $Z = 8$ and $Z = 32$, and their corresponding results are listed in Tab. 1 for comparison. It can be seen that increasing Z will lead to improvement on FMD and KTH. (ii) The number of $\boldsymbol{Y}_\ell$s, *i.e.* L. It will affect the final descriptor length. In the paper, we set $L = 1$ for ResNet18 backbone. In the influence test, we also try other values: $L = 3, 5, 7, 9$, and their results are listed in Tab. 2 for comparison. It can be seen that enlarging L will lead to slight performance improvement. Note that when increasing Z or L, the model complexity will increase accordingly. We did not tuneup these hyper-parameters but just set them to common values which suffice to lead to good results.

Table 1. Classification accuracy (%) using different values of Z, with the ResNet18 backbone used.

Dataset	$Z = 8$	$Z = 16$	$Z = 32$
FMD	82.3 ± 0.6	82.5 ± 0.7	82.9 ± 0.7
KTH	85.4 ± 1.1	85.4 ± 1.1	86.1 ± 1.2

Table 2. Classification accuracy (%) using different values of L, with the ResNet18 backbone used.

Dataset	$L = 1$	$L = 3$	$L = 5$	$L = 7$	$L = 9$
FMD	82.5 ± 0.7	82.4 ± 0.8	82.6 ± 0.7	82.7 ± 0.7	82.7 ± 0.8
KTH	85.4 ± 1.1	85.5 ± 1.1	85.7 ± 1.3	85.8 ± 1.2	85.9 ± 1.2

2. Comparison to a deeper ResNet w/o CLASS

We construct a ResNet50 baseline with very close size as our mode, by adding a basic residual block to the front of ResNet50, which is denoted by ResNet50+. Its number of parameters is slightly larger than our CLASS-Net, *i.e.* 24.7M vs. 23.7M. See Table 3 for the results and comparison. Our CLASS-Net noticeably outperforms ResNet50+. Such results demonstrate that, the performance gain of CLASS-Net is not from the increased module size but from the mechanism of the CLASS module.

Table 3. Performance comparison of CLASS-Net and ResNet50+ in terms of classification accuracy (%).

	DTD	KTH	FMD	MINC	GTOS
CLASS-Net	74.0±0.5	87.7±1.3	86.2±0.9	84.0±0.6	85.6±2.2
Resnet50+	68.8±0.4	81.9±1.8	72.6±1.5	80.9±0.3	81.4±2.5

3. Visualizing cross-layer SSS

See Fig. 1 for an illustration on the log-log fitting done on certain feature tensors in DBC pooling on four texture images of two classes. For better illustration with more points, we set S to a larger value and retrained the model. Each red/blue square denotes the receptive region related to the feature points whose log-log behaviors are shown. As the points lie well on a line in Fig. 1, it indicates that the cross-layer SSS holds well and is captured by our model.

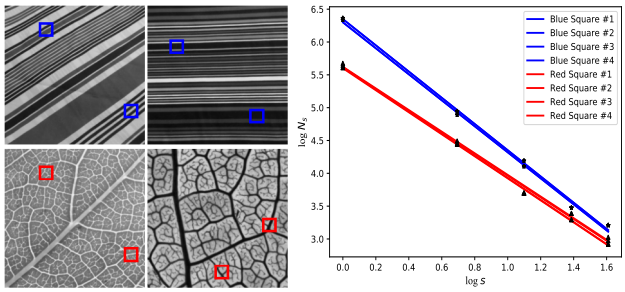


Figure 1. Illustration of log-log fitting in our DBC pooling.

4. Results of removing GAP

See Table 4 for the comparison of our method to a baseline constructed via removing the GAP while flattening the input features directly. Without GAP, there is certain performance decrease which varies on different datasets. Indeed, the CLASS module generates the description by examining the variations of feature maps, which can be roughly viewed as 'high-pass', while GAP characterizes feature maps via average, which can be viewed as 'low-pass'. These results

show that the descriptions generated by CLASS module encode aspects that are different from GAP, providing complementary information.

Table 4. Performance comparison of CLASS-Net w/ and w/o GAP on ResNet18 backbone, in terms of classification accuracy (%).

	DTD	KTH	FMD	MINC	GTOS
CLASS-Net	71.5±0.4	85.4±1.1	82.5±0.7	80.5±0.6	84.3±2.2
w/o GAP	66.0±0.5	84.8±1.2	79.3±1.0	79.5±0.8	83.4±2.0

5. Layer Contribution in DBC Pooling

Recall that our DBC pooling uses 5-layer feature maps for CLASS module with the RseNet18 backbone. It is interesting to check the performance change without one-layer feature map. As shown in Table 5, the performance w/o the k^{th} layer in DBC pooling decreases for all k. The decrease amount is similar for different k.

Table 5. Performance of CLASS-Net w/o the kth layer, in terms of classification accuracy (%), with the ResNet18 backbone used.

	w/o 1^{st}	w/o 2^{nd}	w/o 3^{rd}	w/o 4^{th}	w/o 5^{th}	Full
FMD	81.4±0.9	81.5±1.0	81.7±1.1	81.4±0.9	81.7±1.0	82.5±0.7
DTD	70.8±1.0	70.8±0.9	70.6±1.0	70.6±0.9	70.8±0.8	71.5±0.4