Encoding Spatial Distribution of Convolutional Features for Texture Representation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Existing convolutional neural networks (CNNs) often use global averaging pooling (GAP) to aggregate feature maps into a single representation. However, GAP cannot well characterize complex distributive patterns of spatial features while such patterns play an important role in texture-oriented applications, *e.g.* material recognition and ground terrain classification. In the context of texture representation, this paper addressed the issue by proposing *Fractal Encoding (FE)*, a feature encoding module grounded by multi-fractal geometry. Considering a CNN feature map as a union of level sets of points lying in the 2D space, FE characterizes their spatial layout via a local-global hierarchical fractal analysis which examines the multi-scale power behavior on each level set. This enables a CNN to encode the regularity on the spatial arrangement of image features, leading to a robust yet discriminative spectrum descriptor. In addition, FE has trainable parameters for data adaptivity and can be easily incorporated into existing CNNs for end-to-end training. We applied FE to ResNet-based texture classification and retrieval, and demonstrated its effectiveness on several benchmark datasets.

1 Introduction

Ideal image descriptors for classification are capable of characterizing both local features and global patterns of an image. Local features are about the local singularities or fine details of the image, such as edges and corners. Global patterns are the distribution and spatial layout of local image features. While local features are critical for visual recognition, in many scenarios such as material recognition and ground terrain classification [1], global image patterns also play an important role.

Typical convolutional neural networks (CNNs) for image classification connect convolutional layers with fully-connected (FC) layers. The FC layers encode the spatial layouts of feature maps and act as a classifier. Such a structure fails to produce a robust global representation, as FC layers capture spatial layouts based on "absolute" locations and their output are not invariant to spatial transforms such as translation, rotation and scaling, that often occur in real scenarios.

To achieve the robustness to spatial transforms in CNN-based representation, many existing studies (*e.g.* [2, 3, 4]) apply global average pooling (GAP) to feature maps, which averages all entries in a feature channel so as to eliminate the dependency on spatial locations. However, the naive averaging operation totally disregards the spatial layout of a feature map. As a result, when dealing with images of complex distributive patterns, GAP cannot encode sufficiently discriminative information.

A robust yet discriminative global representation is particularly important for analyzing images of texture patterns [5], since for such images, spatial transforms are one main source of variation and distributive characteristics are the key clues during analysis. In the context of texture analysis and related applications, there have been some efforts [1, 6, 7, 8, 9, 10] on designing a more sophisticated

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

global pooling module. Most of these works are about encoding the numerical distribution of features into a single representation. Nonetheless, they may still lose the details on the spatially distributive characteristics of features which are useful for texture analysis.

Aiming at characterizing the spatially distributive patterns in feature maps to form a robust yet discriminative texture representation, this paper proposed a statistical global feature encoding module called FE (Fractal Encoding), which is grounded by fractal geometry [11, 12].

Motivations Fractal geometry has been widely used for texture analysis and synthesis; see *e.g.* [13, 14, 15]. Its central concept is the fractal dimension that measures the distributive irregularity of a spatial point set. Many studies have shown that natural textures carry fractal dimension information [13, 14, 16]. Since a CNN is a spatial feature extractor, its generated feature maps may also encode fractal dimension information that reflects the regularity of texture patterns. Inspired by this, FE describes the spatial layout of features in terms of spatially distributive regularity using fractal-dimension-based statistics in a hierarchical manner.

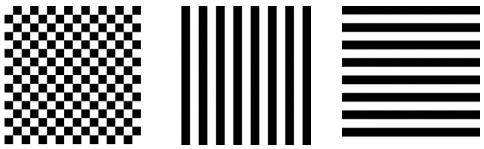
Fractal dimension is established based on the measurement at scale δ. For each δ, an object is measured in a way that ignores the irregularity of size less than δ. If the measurement is proportional to $\delta^{-\beta}$ for some β and β is almost the same for small scales δ, we can compute its limit, which is called the fractal dimension. Let the n-dimensional Euclidean space $\mathbb{R}^n$ be covered by a mesh of n-dimensional hypercubes with diameter d. Given a point set $\mathbb{E}\subset\mathbb{R}^n$, its fractal dimension denoted by $\beta(\mathbb{E})$ is defined by [12]:

$$\beta(\mathbb{E}) = \lim_{d\to 0} \frac{\log \mathcal{N}(\mathbb{E}, d)}{-\log d}, \tag{1}$$

where $\mathcal{N}(\mathbb{E}, d)$ is the number of mesh hypercubes that intersect $\mathbb{E}$.

It can be seen fractal dimension is about cross-scale examination in terms of power law. This makes it possible to encode additional discriminative information on spatial layout that the simple number counting operations in a histogram ignore. For simplicity, consider the fractal dimensions on binary images. The white/black pixels on a binary image can be viewed as a set of points on a 2D grid. See Figure 1 for some numerical examples, where the fractal dimensions are different between the grid and two bars, while the histograms of all the sample images in terms of black/white pixels are the same. If we view those sample images as specific feature patches, the fractal-dimension-based statistics from FE can well distinguish the spatial organization of feature maps for the two types of textures.

In addition to discriminability, fractal dimension is theoretically invariant to spatial bi-Lipschitz transform [16, 15], a general transform that covers a wide range of often-seen image transformations. Note that the influence of the spatial transforms of an image are likely to propagate to its feature maps. Then the FE driven by fractal dimensions can enjoy the robustness in this case.



1.80 1.62 1.62

Figure 1: Fractal dimensions (numbers in the second row) calculated by the scheme of [17] on a grid image and two bar images. The white pixels are viewed as a set of spatial points on a 2D grid. If we view the binary images as some texture features (*e.g.* feature patches from a CNN), then fractal dimensions can well distinguish the spatial distribution of the two types of texture features.

Main idea The key part in FE is the fractal analysis pooling (FAP). For higher discriminability, FAP leverages multi-fractal geometry [18] to form multiple fractal-dimension-based statistical quantities. This leads to a local-global hierarchical fractal analysis process. Concretely, FE looks at a feature map as an union of point sets. The point sets are the level sets under a local point-wise categorization function $D(\cdot)$. The FE calculates a single (global) fractal-dimension-based statistical quantity on every level set:

$$f(\widehat{D}) = \beta(\mathbb{E}_{\widehat{D}}), \text{ where } \mathbb{E}_{\widehat{D}} = \{\boldsymbol{x} \in \mathbb{E} : D(\boldsymbol{x}) = \widehat{D}\}. \tag{2}$$

Encouraged by the robustness of fractal dimension to spatial transforms and scaling of feature values, we design the function D based on local fractal dimension [14] (also called Hölder exponent [12])

defined as the power $\mathcal{D}(\cdot)$ of $\mathcal{U}\left(\mathcal{B}\left(\boldsymbol{x}, r\right)\right) \propto r^{\mathcal{D}(\boldsymbol{x})}$ for $\boldsymbol{x} \in \mathbb{E}$, where $\mathcal{B}\left(\boldsymbol{x}, r\right)$ is a hypercube with center $\boldsymbol{x}$ and diameter $2r$, and $\mathcal{U}$ is some measurement. It can be seen that the local fractal dimension has a similar form as the global one in (1):

$$\mathcal{D}(\boldsymbol{x}) = \lim_{r \to 0} \frac{\log \mathcal{U}(B(\boldsymbol{x}, r))}{-\log r}. \tag{3}$$

In other words, $\mathcal{D}(\boldsymbol{x})$ characterizes the local power law around $\boldsymbol{x}$ under measurement $\mathcal{U}$.

The above local-global hierarchical fractal analysis process involves some discrete operations (*e.g.* box counting) that are non-differentiable, which creates obstacles for back propagation and model training. By appropriate design with soft relaxations, we implement FE using common operations of CNNs, so that it can be end-to-end trained on top of any standard CNN and deal with images of varying sizes. In addition, FE is designed with trainable parameters which can be jointly learned with the backbone network for better adaptivity to data, as well as for automatically setting some hyper-parameters which may be difficult to determine but critical to the performance in practice.

Since FAP and GAP capture distributive features from different aspects respectively, FE combines them with the bi-linear pooling (BLP) [19] to exploit their complementary capabilities. The resulting module is deployed to the ResNet for texture classification and retrieval.

Contributions This work contributes to deep learning and texture analysis in the following aspects. Firstly, we propose a novel fractal-analysis-driven global feature encoding module which is capable of encoding the spatially distributive regularity of features on a feature map. Secondly, we build multi-fractal analysis into a standard CNN for boosting the power of deep-learning-based texture representation, which leads to SoTA results in classification and retrieval tasks.

2 Related Work

Global pooling for deep texture representation The GAP is a prominent choice for global feature aggregation; see *e.g.* [2, 3] for its applications in texture representation. Squeezing a feature map by averaging, GAP is very lossy regarding spatial information. To address this, most existing studies drew inspiration from traditional global descriptors. Cimpoi *et al.* [20] and Song *et al.* [21] used fisher vector encoding [22]. but the resulting network is difficult for end-to-end training. Similar to kernel-based descriptors [23], the BLP proposed by Lin and Maji [19] is based on the outer product of each pair of feature point, which encodes more information than GAP in term of dependencies among channels. Dai *et al.* [24] combined BLP and GAP. However, both BLP and GAP are not trainable. Following bag-of-features [25], Zhang *et al.* [6] proposed a trainable texture encoding (TE) layer which captures the numerical distribution of features based on dictionary learning. Xue *et al.* [7] combined TE and GAP via BLP. A similar module is used in Hu *et al.* [8], but it is placed on top of multiple convolutional layers besides the last one, with an FC layer for result ensemble. Motivated by locality-constrained coding [26], Bu *et al.* [9] proposed a locality-aware coding layer which encodes features with considerations on their locality constraints. To make histograms applicable to CNNs, Peeples [10] proposed a histogram layer, which is used together with GAP for feature encoding.

The proposed FE in this work distinguishes itself from the aforementioned methods by encoding the regularity on the spatial distribution of convolutional features, providing additional discrimination.

Deep CNNs for texture classification In addition to global pooling, there are also some studies focusing on other aspects of deep-learning-based texture classification. To utilize the inherent correlations among the spatial contexts corresponding to visual attributes of texture images, Zhai *et al.* [27] proposed a multiple-attribute-perceived network with a multi-branch architecture. To efficiently describe the spatial organization of complex texture, Zhai *et al.* [28] proposed to learn the dependencies among texture features with a self-attention-like module, with very impressive results achieved.

Fractal geometry for texture representation Fractal analysis has a long history for texture representation. Previous methods mostly use fractal dimensions with various calculation schemes as local appearance descriptors [14] or global distribution descriptors [13, 16]. These fractal-based descriptors are handcrafted and not adaptive to data. In comparison, our work forms a deep-learning-based fractal analysis model which can be end-to-end trained from data for better adaptivity.

3 Fractal Encoding: Pooling with Hierarchical Fractal Geometry

The proposed FE module processes a feature tensor from the CNN backbone with two paths, which perform GAP and FAP receptively. An FC layer is attached behind GAP to align GAP to FAP in terms of output size, magnitude and semantic. An upsampling layer is applied before FAP for higher spatial resolution of feature maps that makes FAP more stable. The results from the two paths are integrated by BLP followed by an FC layer. See Figure 2 for the pipeline of the FE.

The FAP first compresses the input feature tensor to C_0 channels by a $\text{Conv}_{1\times1}\rightarrow\text{BN}\rightarrow\text{ReLU}$ block. On the new feature tensor, FAP processes it slice by slice and the concatenates the results over all slices. By viewing the entries on a feature slice as a set of points in $\mathbb{R}^2$, FAP runs a hierarchical fractal dimension analysis on it, *i.e.*, FAP groups the points and calculates a fractal-dimension-based statistical quantity on each group. Accordingly, there are three main blocks in FAP, as described in the next subsections.

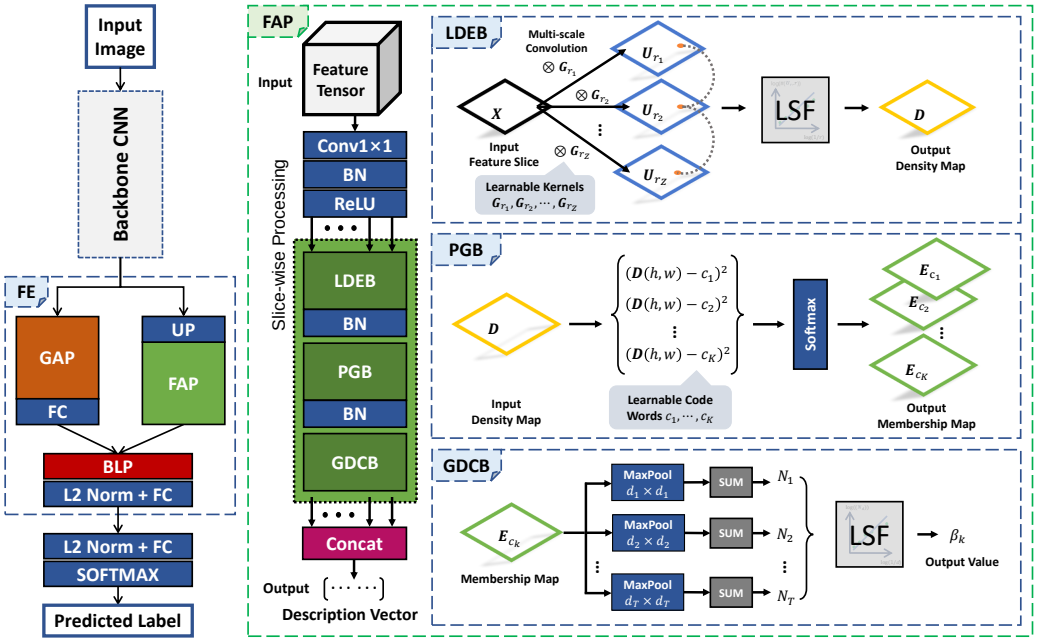


Figure 2: Diagram of proposed fractal encoding (FE) module.

3.1 Local Dimension Estimation Block (LDEB)

The LDEB calculates a point-wise local fractal dimension map $\boldsymbol{D} \in \mathbb{R}^{H\times W}$ from the input feature slice $\boldsymbol{X} \in \mathbb{R}^{H\times W}$:

$$\text{LDEB} : \boldsymbol{X} \in \mathbb{R}^{H\times W} \rightarrow \boldsymbol{D} \in \mathbb{R}^{H\times W}. \tag{4}$$

The calculation is based on (3), where one key is computing the measurement $\mathcal{U}$, *i.e.*, forming a series of measurement maps: $\boldsymbol{U}_r \in \mathbb{R}^{H\times W}$, $r = 1, 2, \cdots$.

The measurement $\mathcal{U}$ in (3) calculates a value on local block $\mathcal{B}(\boldsymbol{x}, r)$ for every $\boldsymbol{x}$. For a feature slice, a local block can be viewed as a local patch of size $r \times r$ in the slice, and $\mathcal{U}$ is essentially calculated on all the patches sampled with a sliding window on $\boldsymbol{X}$, which can be modeled by a convolution process. We define the measurement to be learnable for better data adaptivity. Since the measurement is defined on multiple scales, we simulate this by the multi-scale convolution:

$$\boldsymbol{U}_r = \boldsymbol{G}_r \otimes \boldsymbol{X}, \quad r = 1, 2, \cdots, \tag{5}$$

where $\otimes$ denotes convolution, and $\boldsymbol{G}_r \in \mathbb{R}^{r\times r}$ is an $r \times r$ kernel learned together with the model.

The numerical implementation of (3) is done by the least-squares fitting (LSF) on $\boldsymbol{U}_r$ versus r in the log-log coordinate system. A series of measurement maps $\{\boldsymbol{U}_r,\ r = r_1, r_2, \cdots, r_Z\}$ are first

calculated, where $r_1 < \cdots < r_Z$ are a finite sequence of predefined ordered positive radii. Then the estimation of local fractal dimension map in (3) becomes an LSF problem:

$$\min_{\boldsymbol{D},\boldsymbol{P}} \sum_{z=1}^{Z} (\boldsymbol{D}(w,h)\log r_z + \boldsymbol{P}(w,h) - \log\ \boldsymbol{U}_{r_z}(w,h))^2, \tag{6}$$

where $\boldsymbol{P}(w,h)$ denotes bias term in the fitting. Setting the derivative of (6) *w.r.t.* $\boldsymbol{D}$ and $\boldsymbol{P}$ to zero respectively, we have

$$\begin{cases} \sum_{z=1}^{Z} \left(\boldsymbol{D}(w,h)\log^2 r_z + \boldsymbol{P}(w,h)\log r_z - \log r_z \log\ \boldsymbol{U}_{r_z}(w,h)\right) = 0 \\ \sum_{z=1}^{Z} (\boldsymbol{D}(w,h)\log r_z + \boldsymbol{P}(w,h) - \log\ \boldsymbol{U}_{r_z}(w,h)) = 0 \end{cases} \tag{7}$$

Then, the local fractal dimension map $\boldsymbol{D} \in \mathbb{R}^{W\times H}$ is computed as the solution to (7):

$$\boldsymbol{D}(w,h) \ = \ \frac{Z\sum_{z=1}^{Z}\log r_z \log\ \boldsymbol{U}_{r_z}(w,h) - \left(\sum_{z=1}^{Z}\log r_z\right)\left(\sum_{z=1}^{Z}\log\ \boldsymbol{U}_{r_z}(w,h)\right)}{Z\sum_{z=1}^{Z}\log^2 r_z - \left(\sum_{z=1}^{Z}\log r_z\right)^2}. \tag{8}$$

3.2 Point Grouping Block (PGB)

The PGB categorizes the entries on the feature slice based on the map $\boldsymbol{D}$ output by the LDEB. Let

$$\mathbb{E} = \{1,\cdots,W\} \times \{1,\cdots,H\} \subset \mathbb{R}^2, \tag{9}$$

$$\mathbb{E}_{\widehat{D}_k} = \{(h,w) \in \mathbb{E} : \widehat{D}_k \le \boldsymbol{D}(h,w) < \widehat{D}_{k+1}\}, \tag{10}$$

for $k = 1,\cdots,K$, given the intervals $[\widehat{D}_1,\widehat{D}_2],[\widehat{D}_2,\widehat{D}_3],\cdots,[\widehat{D}_K,\widehat{D}_{K+1}]$ where $\widehat{D}_k \in \mathbb{R}$ for all k. The set $\mathbb{E}_{\widehat{D}_k}$ can be represented by a binary map $\boldsymbol{E}_{\widehat{D}_k}$ of the same size as $\boldsymbol{D}$, where $\boldsymbol{E}_{\widehat{D}_k}(h,w) = 1$ if $\widehat{D}_k \le \boldsymbol{D}(h,w) < \widehat{D}_{k+1}$ and 0 otherwise.

The setting of intervals $[\widehat{D}_k,\widehat{D}_{k+1}], k = 1,\cdots,K$ can have a noticeable impact on the accuracy of multi-fractal analysis [18]. However, they are non-trivial to determine, as the range of $\boldsymbol{D}$ is uncertain and uniform intervals are often not optimal. In addition, grouping with hard assignment may cause noticeable quantization errors [29], as well as optimization difficulty in back propagation during model training. To address these issues, we implement the grouping function using a soft assignment scheme [29] with trainable quantization intervals. Instead of using binary values, the soft assignment uses real values in $[0,1]$ for defining $\boldsymbol{E}_{c_k}(h,w)$ to indicate the degree of membership. Let $\{c_k \in \mathbb{R},\ k = 1,...,K\}$ denote a set of trainable interval anchors. For each c_k, we define its corresponding membership map $\boldsymbol{E}_{c_k} \in [0,1]^{W\times H}$ by

$$\boldsymbol{E}_{c_k}(w,h) = \frac{\exp(-s_k(\boldsymbol{D}(w,h) - c_k)^2)}{\sum_{k=1}^{K}\exp(-s_k(\boldsymbol{D}(w,h) - c_k)^2)}, \tag{11}$$

where $\{s_k \in \mathbb{R}\}_{k=1}^{K}$ is a set of learnable weights. It can be implemented via Softmax. To summarize, the PGB takes a local dimension map $\boldsymbol{D}$ as input, and generates a series of soft membership maps:

$$\text{PGB} : \boldsymbol{D} \in \mathbb{R}^{H\times W} \to \{\boldsymbol{E}_{c_k} \in [0,1]^{H\times W}\}_{k=1}^{K}. \tag{12}$$

3.3 Global Dimension Calculation Block (GDCB)

The GDCB accepts $\{\boldsymbol{E}_{c_k}\}_{k=1}^{K}$ from the PGB as input, calculates a fractal-dimension-based quantity β on each $\boldsymbol{E}_{c_k}$ individually, and finally concatenates β_ks as output:

$$\text{GDCB} : \{\boldsymbol{E}_{c_k} \in [0,1]^{H\times W}\}_{k=1}^{K} \to \boldsymbol{\beta} = (\beta_1,\beta_2,\cdots,\beta_K) \in \mathbb{R}^K. \tag{13}$$

The quantity β_k is calculated based on (1). Similar to the computation on $\boldsymbol{D}$, Equation (1) can be numerically implemented by using bi-log LSF. However, the box counting operation in (1) is about counting the patches with non-zero entries, which is non-differentiable and thus makes training infeasible. Also recall that $\boldsymbol{E}_{c_k}$ is a soft membership map with values in $[0,1]$ instead of binary ones. It is very likely that $\boldsymbol{E}_{c_k}$ contains no zeros and the box counting is inapplicable in this case.

Based on above considerations, we replace the box counting $\mathcal{N}$ in (1) by the sum of the maximal values from all patches of size $d \times d$ in $\boldsymbol{E}_{c_k}$. When $\boldsymbol{E}_{c_k}$ is binary, it equals to box counting. The new scheme can be efficiently implemented by stride-1 max pooling with window size $d \times d$ and summation. By varying $d = d_1, \cdots, d_T$, we have the sums denoted by $N_1, \cdots, N_T$. Similar to (7), β_k is then calculated as the LSF solution to the modified (1):

$$\beta_k = \frac{T \sum_{t=1}^{T} \log d_t \log N_t - \left(\sum_{t=1}^{T} \log d_t\right)\left(\sum_{t=1}^{T} \log N_t\right)}{T \sum_{t=1}^{T} \log^2 d_t - \left(\sum_{t=1}^{T} \log d_t\right)^2}. \tag{14}$$

4 Experimental Evaluation

4.1 Deployment of FE module and setting of experiments

The FE module can be easily incorporated into a CNN by porting it on top of some convolutional layers. To evaluate the effectiveness of FE, we insert it next to the last convolutional layer of the ResNet [30], an often-used backbone (*e.g.* [6, 7]). The resulting CNN is referred to as FENet. Inheriting the advantages from FE, FENet can deal with flexible image size and be end-to-end trained.

We apply FENet to texture classification and retrieval on six benchmark datasets, including GTOS [1], GTOS-M [7], KTH [31], MINC [32], DTD [33] and FMD [34]. Same as [28, 6], we use the provided split schemes on GTOS, MINC and DTD, and random 10 splits on KTH-TIPS2b and FMD with recommended split sizes. The mean and standard deviation of classification accuracies over all splits are calculated. The results are reported using two runs on GTOS-M and five-time statistics on other datasets.

We set $C_0 = 3$ in GAP, $r = 1, 2, ..., 6$ in LDEB, $K = 16$ in PGB, and $d_t = 2, 3, \cdots, 6$ in GDCB. The output sizes of the two paths in FE are both set to 48. Following [28, 6], we use ResNet-18 and ResNet-50 as the backbone respectively. On all the datasets, FENet is trained with the cross-entropy loss via the momentum SGD optimizer with default settings and 30 epochs. The batch size is set to 16 on FMD, 32 on KTH, and 64 on the other four datasets. The learning rate is initialized to $1e^{-3}$ on FMD, $5e^{-3}$ on MINC, GTOS and GTOS-M, and $1e^{-2}$ on KTH and DTD datasets, with cosine decay every 10 epochs. The ResNet backbones are initialized with the pre-trained models on Imagenet. The $\{\boldsymbol{G}_r\}_r$ in (5) are initialized as Gaussian kernels with bandwidth 1. Other parameters of FENet are initialized by Xavier [35]. Data augmentation via horizontal flipping and random cropping to 224×224 is applied. Our FENet is implemented with PyTorch 1.7, and the code will be released upon paper's acceptance. All the experiments were run on a single Titan XP GPU.

4.2 Results in Texture Classification

Our FENet is compared with several recent deep models for texture classification, which include DeepTEN [6], DEPNet [7], LSCNet [9], MAPNet [27], DSRNet [28] and HistNet [10]. The results on the six datasets are listed in Table 1. The results of the compared methods are directly quoted from existing literature whenever possible, or left blank otherwise.

It can be seen from Table 1 that FENet performs the best on KTH, FMD, MINC and GTOS in terms of mean accuracy, no matter using ResNet-18 backbone or ResNet-50 backbone. On GTOS-M, the FENet is the best performer with ResNet-18 backbone.

On DTD, the FENet performs better than others except DSRNet and MAPNet. The reason why FENet shows unsatisfactory performance on DTD is probably that, unlike most classic texture recognition datasets, the creation and categorization of DTD focus more on higher-level semantics such as perceptual properties and visual impressions; see supplementary materials for some examples. The regularity of spatial feature distribution of the images in DTD is insufficient to distinguish such high-level semantics. As a result, the capability of the FE module for characterizing spatial distribution regularity is as helpful as on other traditional texture datasets.

It is worth mentioning that the training data of GTOS-M is divided into three subsets with small, medium and large scales respectively. It is shown in [7] that the performance of deep CNNs such as DeepTEN using the single-resolution subset rather than the multi-resolution ones of GTOS-M will have a noticeable accuracy decrease. In comparison, our FENet has no performance decrease when

Table 1: Classification accuracies (%) in the form of "mean±s.t.d.". Best results are **boldfaced**. The precision varies as the results are quoted from different works.

	Method	DTD	KTH	FMD	MINC	GTOS	GTOS-M
ResNet-18	DeepTEN	-	-	-	-	-	76.12
	DEPNet	-	-	-	-	-	82.18
	LSCNet	-	-	76.3	-	-	-
	MAPNet	69.5 ± 0.8	80.9 ± 1.8	80.8 ± 1.0	-	80.3 ± 2.6	82.98 ± 1.6
	DSRNet	**71.2** ± 0.7	81.8 ± 1.6	81.3 ± 0.8	-	81.0 ± 2.1	83.65 ± 1.5
	HistNet	-	-	-	-	-	79.75 ± 0.8
	Ours	69.59 ± 0.04	**86.62** ± 0.08	**82.26** ± 0.29	**80.57** ± 0.10	**83.10** ± 0.23	**85.10** ± 0.36
ResNet-50	DeepTEN	69.6	82.0 ± 3.3	80.2 ± 0.9	81.3	84.5 ± 2.9	-
	DEPNet	73.2	-	-	82.0	-	-
	LSCNet	-	-	81.2	-	-	-
	MAPNet	76.1 ± 0.6	84.5 ± 1.3	85.2 ± 0.7	-	84.7 ± 2.2	86.64 ± 1.5
	DSRNet	**77.6** ± 0.6	85.9 ± 1.3	86.0 ± 0.8	-	85.3 ± 2.0	**87.03** ± 1.5
	HistNet	72.0 ± 1.2	-	-	82.4 ± 0.3	-	-
	Ours	74.20 ± 0.10	**88.24** ± 0.23	**86.74** ± 0.19	**83.98** ± 0.14	**85.71** ± 0.06	85.20 ± 0.43

trained on single-scale data, and the result of FENet on GTOS-M in Table 1 is actually based on only the medium-scale training data. See Section 4.5 for more analysis.

4.3 Results in Texture Retrieval

The globally-pooled feature fed to the classification module in FENet can be used as a texture descriptor. Its effectiveness is evaluated by the retrieval task as follows. An image sample is represented by its corresponding feature tensor output by the last pooling layer (*i.e.* after BLP). Each sample in the test dataset is used as a query to search the remaining samples, and its top-R similar samples in the dataset are collected in terms of cosine similarity. The average precision and recall over all query samples are calculated.

By varying R from 1 to the number of searched samples, we obtain the precision-recall (PR) curves and the corresponding mean average precision (mAP), which are shown in Figure 3. It can be seen that FENet achieved the best retrieval results among the compared methods on GTOS-M, KTH and FMD datasets. Such results have demonstrated the effectiveness of the texture feature representation learned in FENet.

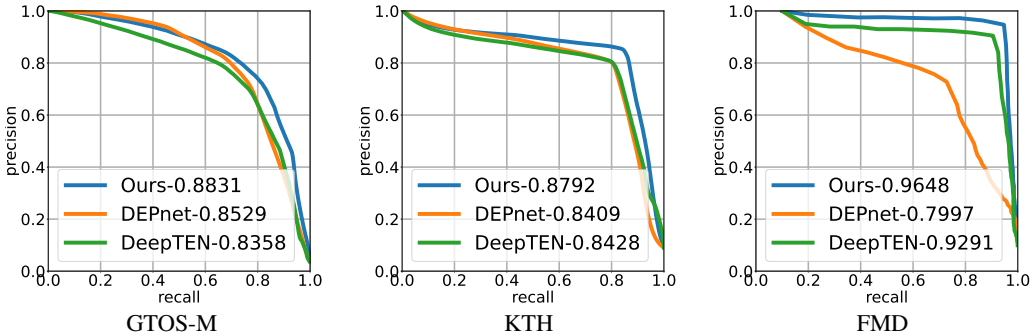


Figure 3: PR curves of several methods in texture retrieval. The values in legends are mAP.

4.4 Effectiveness of FE Module

Ablation study on FE module in FENet To demonstrate the effectiveness of FE in FENet, we form a baseline ("w/o FE Module") from FENet by removing its FE module as well as the BLP layer and adapting the dimension of the related FC layer. Similarly, we form another baseline ("w/o GAP") by removing the GAP layer in FENet. These two baselines are trained in the same way as FENet.

The results of the baselines are compared to FENet on GTOS-M and KTH datasets in Table 2. It can be seen that the performance decrease caused by the removal of FE Module is significant on GTOS-M and KTH. This suggests that the regularity of spatial distribution captured by FE Module plays a more important role. When we removed the GAP in FE, performance decreases slightly. All these results have demonstrated the effectiveness of FE Module in FENet.

Table 2: Classification accuracy (%) of FENet with ablations.

Dataset	Backbone	w/ FE Module [FENet]	w/o FE Module	w/o GAP
GTOS-M	ResNet-18	85.10 ± 0.36	82.29 ± 0.30	83.86 ± 0.64
KTH	ResNet-18	86.62 ± 0.08	83.25 ± 0.76	86.43 ± 0.07
GTOS-M	ResNet-50	85.20 ± 0.43	82.57 ± 0.65	84.33 ± 0.09
KTH	ResNet-50	88.24 ± 0.23	83.82 ± 0.25	87.59 ± 0.15

Incorporating FE module into DeepTEN We also demonstrate the effectiveness of the FE module by incorporating it into DeepTEN [6] and examining the performance gain. Recall that the DeepTEN also uses a ResNet backbone, together with an encoding module (called TE) which acts as a soft histogram with learned bins. Similar to our FENet, the FE module (together with the upsampling layer in front of it) is inserted in parallel to the TE layer after the last RB of the ResNet backbone. The output of the FE and TE modules are integrated by BLP. The resulting model is trained with the same scheme as the original work.

The results on the four datasets are listed in Table 3. It can be seen that the introduction of FE module brings significant improvement to DeepTEN on GTOS-M, KTH and FMD, as well as noticeable improvement on MINC. Such an improvement again demonstrated the effectiveness of the FE module. Indeed, the TE module in DeepTEN is essentially a soft counting layer which only considers the qualitative distribution but ignores the spatial distribution of features in feature slices. The FE module provides complementary information to the TE module by characterizing the spatial organization of feature maps based on multi-fractal analysis.

Table 3: Classification accuracy (%) of DeepTEN w/o and w/ FE module.

Dataset	Backbone	DeepTEN	DeepTEN + FE Module
GTOS-M	ResNet-18	76.12	81.15 ± 0.29
KTH	ResNet-50	82.00 ± 3.30	86.93 ± 0.30
FMD	ResNet-50	80.20 ± 0.90	86.32 ± 0.43
MINC	ResNet-50	81.3	83.62 ± 0.11

4.5 More Analysis

Complexity and efficiency We compare the complexity of backbone CNN, DeepTEN, DEPNet and FENet in terms of the number of parameters and number of floating operations per second (FLOPS). We also compare their efficiency in term of inference time. See Table 4 for the comparison. Our FE module only introduces a small number of additional parameters to the backbone, and the complexity of FENet is comparable to that of other deep models under the same backbone.

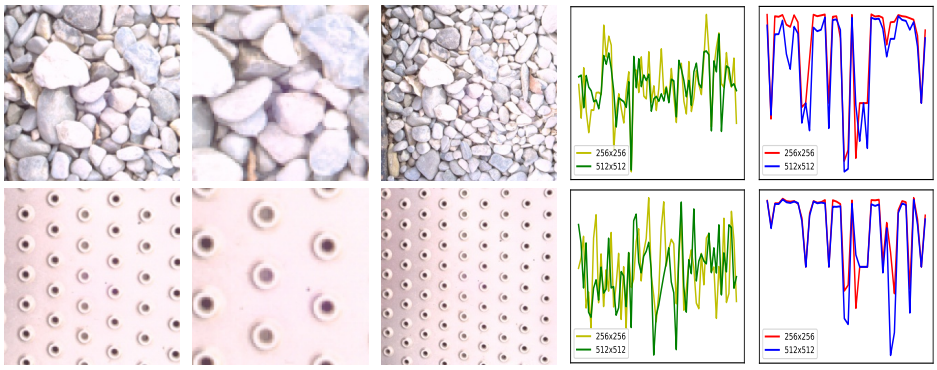
Table 4: Number of parameters, number of FLOPs and inference time on GPU of different models. The values on the left/right are for the ResNet-18/ResNet-50 backbones respectively.

Method	# Parameters (1×10^6)	# FLOPS (1×10^9)	GPU Time (s/100pics)
Backbone CNN	11.1924 / 23.5716	1.8185 / 4.1094	0.1387 / 0.3665
DeepTEN	11.3736 / 23.9017	1.8229 / 4.1234	0.1165 / 0.3000
DEPNet	12.2679 / 25.4841	1.8333 / 4.1627	0.0751 / 0.3160
FENet	11.5071 / 23.9323	1.8205 / 4.1159	0.1393 / 0.4530

Generalization across scales Scaling is one often-seen variation of images which is challenging for texture analysis. Empirically, the FENet enjoys good generalization to scale changes. An experiment

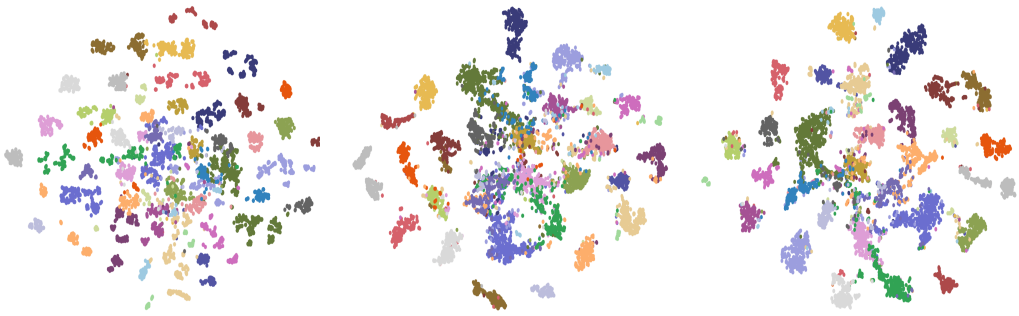
is conducted on the GTOS-M dataset whose training/test set is divided into three subsets with resolution of 256×256, 384×384 and 512×512 respectively; see Figure 4 for some examples. We train the ResNet-18 baseline, DEPNet and FENet only using the subset of resolution 384×384. Then the models are used as the global descriptors on the test set of resolution 384×384, as well as on both the training and test sets of other two resolutions. This is done by taking the features generated from the GAP layer in ResNet-18 baseline or the BLP layer in DEP and FENet. Figure 5 visualizes the descriptions via t-SNE [36]. It can be seen that the descriptions generated by FENet are more concentrated than those of the other two models.

See also Figure 4 for the visualization on the descriptions generated on specific images with different scales. The FENet outputs descriptions with higher degree of invariance to scale changes. Such a robustness is probably due to the robustness of the fractal-dimension-based features to bi-Lipschitz transform as well as the cross-scale analysis in FE.



(a) 384×384 (b) 512×512 (c) 256×256 (d) TEN Feature (e) FE Feature

Figure 4: Multi-scale sample images in GTOS-M and their features generated by TE and FE modules.



(a) ResNet-18 Baseline (b) DEPNet (c) FENet

Figure 5: The t-SNE visualization results on the scale-varying images of GTOS-M.

5 Summary

Feature aggregation is a key step in many CNN-based vision tasks, particularly for texture analysis. It aims at transforming feature maps into a single robust yet discriminative representation, by eliminating the dependency of features on spatial order while encoding the distributive characteristics of features. In this work, we fulfilled this goal by the proposed FE module, which leverages multi-fractal geometry and uses a hierarchical fractal analysis process for encoding the regularity of spatial arrangement in a feature map. The FE module can be easily incorporated into standard CNNs for end-to-end training. Building FE into ResNet, we developed the FENet for texture classification and retrieval. The experiments on six benchmark datasets have demonstrated the effectiveness of FENet.

In addition to texture representation, the FE can also be incorporated into the CNNs in other visual recognition tasks which need to utilize the spatial organization of local patterns. We leave this to our future work.

References

[1] J. Xue, H. Zhang, K. Nishino, and K. Dana. Differential viewpoints for ground terrain material recognition. *IEEE Trans. Pattern Anal. Mach. Intell*, pages 1–1, 2020.

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

[3] Shin Fujieda, Kohei Takayama, and Toshiya Hachisuka. Wavelet convolutional neural networks for texture classification. Jul 2017.

[4] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[5] 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. *International Journal on Computer Vision*, (Nov):1–36, 2018.

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

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

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

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

[10] Joshua Peeples, Weihuang Xu, and Alina Zare. Histogram layers for texture analysis. *arXiv preprint arXiv:2001.00215*, 2020.

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

[12] Kenneth J Falconer and KJ Falconer. *Techniques in fractal geometry*, volume 16. Wiley Chichester (W. Sx.), 1997.

[13] Lance M Kaplan. Extended fractal analysis for texture classification and segmentation. *IEEE Trans. Image Process.*, 8(11):1572–1585, 1999.

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

[15] Hongteng Xu, Junchi Yan, Nils Persson, Weiyao Lin, and Hongyuan Zha. Fractal dimension invariant filtering and its cnn-based implementation. In *Proc. CVPR*, 2017.

[16] Yong Xu, Hui Ji, and Cornelia Fermüller. Viewpoint invariant texture description using fractal analysis. *International Journal on Computer Vision*, 83(1):85–100, Jun 2009.

[17] Mandelbrot and Benoit B. The fractal geometry of nature. *American Journal of Physics*, 51(4), 1998.

[18] Lars Olsen. Multifractal geometry. In *Fractal geometry and stochastics II*, pages 3–37. Springer, 2000.

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

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

[21] Y. Song, F. Zhang, Q. Li, H. Huang, L. J. O'Donnell, and W. Cai. Locally-transferred fisher vectors for texture classification. In *Proc. ICCV*, pages 4922–4930, 2017.

[22] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of Computer Vision*, 105(3):222–245, 2013.

[23] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. In *Advances in Neural Information Processing Systems*, pages 244–252, 2010.

[24] 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.

[25] 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.

[26] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Proc. CVPR*, pages 3360–3367. IEEE, 2010.

[27] 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.

[28] 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.

[29] Yong Xu, Yuhui Quan, Zhuming Zhang, Haibin Ling, and Hui Ji. Classifying dynamic textures via spatiotemporal fractal analysis. *Pattern Recognition*, 48(10):3239–3248, 2015.

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

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

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

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

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

[35] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. Int. Conf. Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[36] Van Der Maaten Laurens and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2605):2579–2605, 2008.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See [Failure Cases] in Supplementary Materials.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Broader Impact
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] **However, as stated in Section 4.1, the code will be made public upon paper's acceptance.**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [N/A]

   (c) Did you include any new assets either in the supplemental material or as a URL? [No]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]