This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2019.2903300, IEEE Transactions on Image Processing

Texture Classification in Extreme Scale Variations using GANet

Li Liu , Jie Chen, Guoying Zhao, Paul Fieguth, Xilin Chen, Matti Pietikäinen

***Abstract*—Research in texture recognition often concentrates on recognizing textures with intraclass variations such as illumination, rotation, viewpoint and small scale changes. In contrast, in real-world applications a change in scale can have a dramatic impact on texture appearance, to the point of changing completely from one texture category to another. As a result, texture variations due to changes in scale are amongst the hardest to handle. In this work we conduct the first study of classifying textures with extreme variations in scale. To address this issue, we first propose and then reduce scale proposals on the basis of dominant texture patterns. Motivated by the challenges posed by this problem, we propose a new GANet network where we use a Genetic Algorithm to change the filters in the hidden layers during network training, in order to promote the learning of more informative semantic texture patterns. Finally, we adopt a FV-CNN (Fisher Vector pooling of a Convolutional Neural Network filter bank) feature encoder for global texture representation.**

Because extreme scale variations are not necessarily present in most standard texture databases, to support the proposed extreme-scale aspects of texture understanding we are developing a new dataset, the *Extreme Scale Variation Textures (ESVaT)*, to test the performance of our framework. It is demonstrated that the proposed framework significantly outperforms gold-standard texture features by more than 10% on ESVaT. We also test the performance of our proposed approach on the KTHTIPS2b and OS datasets and a further dataset synthetically derived from Forrest, showing superior performance compared to the state of the art.

***Index Terms*—Texture descriptors, rotation invariance, local binary pattern (LBP), feature extraction, texture analysis**

I. Introduction

Texture analysis [1] plays a key role in computer vision, supporting a great many applications: image and scene classification, object detection and recognition, medical image analysis, robot vision and autonomous navigation for unmanned aerial vehicles. Research in texture recognition [1]–[3] often concentrates on recognizing textures with intraclass variations such as illumination, rotation, viewpoint, and small scale changes. On the other hand, in many real world applications the significant variations or changes of scale may have a dramatic impact on the appearance of an underlying texture, as resolved in some image. For example, as illustrated in Fig. 1, as the scale becomes increasingly and substantially coarser, from left to right, the corresponding texture category also changes; for example, the top row (grasses) changes from grass to lawn to a nearly featureless field. Particularly in applications such as robot vision or remote surveillance, extreme scale changes can occur quite routinely (Fig. 1 (d)), when a single image contains both near and far distances, meaning that in contexts involving autonomous or machine vision it becomes crucial to investigate texture analysis under extreme scale variations. To the best of our knowledge no existing texture classification methods can handle scale changes of such magnitude. Our aim is to fill this gap and to develop effective methods for classifying textures under the sorts of scale changes illustrated in Fig. 1.

Fig. 1. A changing scale can have a dramatic impact on the appearance of textures: (*a*) grasses, (*b*) wheat, and (*c*) trees, with the images from significantly different viewpoints, essentially sampling the textures from a broad extent of underlying continuous scale. Even more challenging (*d*) are cases of continuous scale variations within a single image. Note that for the last column of (a) (b) and (c), the images are basically textureless and only show a region of a certain color. For the second last column of (a) (b) and (c), the images show small scales and has quite different texture appearances as those of the first, second and third column. For (d), each column has a different texture, and shows continuous but extreme scale changes, containing textures in both near and far distances. All images are collected from the Internet.

Much effort has been devoted to exploring and developing texture features that are robust to a variety of imaging changes, particularly to illumination, rotation, viewpoint and noise [4]–

Corresponding author: li.liu@oulu.fi

Li Liu is with the College of System Engineering, National University of Defense Technology, Changsha, China, and she is also with the Center for Machine Vision and Signal analysis, University of Oulu, Finland (email: li.liu@oulu.fi).

Guoying Zhao and Matti Pietikäinen are with the Center for Machine Vision and Signal Analysis, University of Oulu, 90014 Oulu, Finland. email: {guoying.zhao, matti.pietikainen}@oulu.fi

Jie Chen is with the Center for Machine Vision and Signal analysis, University of Oulu, Finland and he is also with Peng Cheng Laboratory, China.

P. Fieguth is with the Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada. email: pfieguth@uwaterloo.ca

X. Chen is with the Key Lab of Intelligent Information Processing of CAS, Institute of Computing Technology, CAS, Beijing, 100190, China, and he is also with Peng Cheng Laboratory, China. email:xlchen@ict.ac.cn

[10]. In terms of change-invariant features, scale variations are amongst the hardest to handle and only modest progress has been made in finding features invariant to even small scale changes [2], [11], [12]. These methods have demonstrated good performance on benchmark datasets such as Brodatz [13], CUReT [14], UIUC [2] and KTHTIPS [15], and more recently the OpenSurfaces (OS) dataset [16]; however all of these datasets exhibit rather modest scale variations, not necessarily representative of the significantly harder problem of texture recognition in the presence of the extreme scale variations of interest here.

Existing successful texture representation paradigms seek to represent textured images statistically as histograms over texels or textons [2], [12], [17], [18]. The fundamental question is the *scale* of this image, in that texels may compactly represented when computed at some certain scale, but not at larger and/or smaller scales. The *right* scale is thus part of the definition of the texture and plays an important role, recognized early in the pioneering work of Julesz [19].

Motivated by the above, our method first searches scale proposals and reduces the number of proposals by finding the basic dominant texels that occur most frequently in the image. Based on the challenges posed by this problem, we then propose a new network GANet and adopt a FV-CNN feature encoder for global texture representation. The main contributions of this work are summarized as follows:

- To the best of our knowledge, we conduct pioneering investigation towards the problem of recognizing textures exhibiting extreme scale variations, that is, with scale variations of two or more orders of magnitude.
- When the scale of a texture changes, the category of the resulting texture image also changes at some boundary, boundaries which are important to identify in order to label two images of the same physical material as different texture classes. This paper offers the first investigation of such scale boundaries.
- We propose a new network, which we refer to as GANet, which can learn more informative texture patterns by using a genetic algorithm to change the filters in the hidden layers during network training.
- We contribute a large texture dataset consisting of 15,747 texture images having substantial scale variations, in an effort to support the study of texture and scale.

II. Related Work

Texture can be characterized by statistical distributions of texels or textons, which are defined as repetitive local features that are responsible for the preattentive discrimination of textures [19]. The recent literature on texture analysis is vast, and recent surveys can be found in [7], [18], [20], [21].

The approach in this work is related to the texel size or texture scale. Lindeberg [22] investigated scale for texture description, suggesting that texture characteristics strongly depend upon it. Mirmehdi and Petrou [23] discussed scale variations in real scenes and used them for the segmentation of color textures. There is recent work focusing on the estimation of the local or global scale of textured images without explicitly extracting texture texels [2], [11]. To search the scale proposals of a given texture image, we adopt the binarized normed gradients (BING) algorithm [24], which has been shown to be very efficient and powerful in proposing local salient regions.

Recently, deep Convolutional Neural Networks (CNN) [25]–[27] [28], [29] have demonstrated excellent results in many domains of computer vision, including texture recognition [6]–[8], [12], [30], [31]. For example, Zhang et al. proposed a Deep Texture Encoding Network (DeepTEN) with a novel Encoding Layer integrated on top of convolutional layers, which ports the entire dictionary learning and encoding pipeline into a single model. Different from other methods build from distinct components, such as SIFT descriptors or pre-trained CNN features for material recognition. DeepTEN provides an end-to-end learning framework, where the inherent visual vocabularies are learned directly from the loss function. The features, dictionaries, encoding representation and the classifier are all learned simultaneously. The representation is orderless and therefore is useful for material and texture recognition. Dai et al. proposed an effective fusion architecture - FASON that combines second order information flow and first order information flow. FASON allows gradients to back-propagate through both flows freely and can be trained effectively. They build a multi-level deep architecture to exploit the first and second order information within different convolutional layers. Zhang Detect 2018 et al. proposed an effective and scalable method for learning feature detectors for textures, which combines an existing "ranking" loss with an efficient fully-convolutional architecture as well as a new training-loss term that maximizes the "peakedness" of the response map. They demonstrated that their detector is more repeatable than existing methods, leading to improvements in a real-world texture-based localization application.

Xian TextureGAN et al. investigated deep image synthesis guided by sketch, color, and texture. They allowed a user to place a texture patch on a sketch at arbitrary locations and scales to control the desired output texture. Their generative network learns to synthesize objects consistent with these texture suggestions. To achieve this, they develop a local texture loss in addition to adversarial and content loss to train the generative network, TextureGAN.

However, it is a common belief that existing CNN architectures are not robust to appearance variations such as rotation, scale and noise [7] (see Section V for more details.), and the texture recognition work on CNN mainly focuses on domain transferability [6], [12].

In additon, there are other approaches proposed for texture analysis. Mehta and Egiazarian presented a rotation invariant and computationally efficient texture descriptor called Dominant Rotated Local Binary Pattern (DRLBP). A rotation invariance is achieved by computing the descriptor with respect to a reference in a local neighborhood. A reference is fast to compute maintaining the computational simplicity of the Local Binary Patterns(LBP). DRLBP not only retains the com-plete structural information extracted by LBP, but it also captures the complimentary information by utilizing the magnitude information, thereby achieving more discriminative

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2019.2903300, IEEE Transactions on Image Processing

power. Depeursinge et al. presented texture operators called SWM, which encoding class-specific local organizations of image directions (LOIDs) in a rotation invariant fashion. The LOIDs are key for visual understanding, and are at the origin of the success of the popular approaches. SWM learns data-specific representations of the LOIDs in a rotation-invariant fashion. The image operators are based on steerable circular harmonic wavelets (CHWs), offering a rich and yet compact initial representation for characterizing natural textures.

Motivated by the challenges posed by recognizing textures exhibiting extreme scale variations, we propose a new network, GANet, where we use a genetic algorithm (GA) to change the filters in the hidden layers during network training in order to promote the learning of more informative semantic texture patterns and to suppress the number of nonsemantic ones. Here, semantic pattern for textures means a pattern include a texel/texton or a combination of texels/textons. Non-semantic pattern for textures means a pattern does not include any texel/texton. There certainly has already been work applying GA to deep learning [32]–[36]: The work in [35] aims at learning the architectures of modern CNNs by employing an encoding method to represent each network architecture in a fixed length binary string; in [33], a GA was used to train networks with a large number of layers, each of which was trained independently to reduce the computational burden; in [32], a GA was used to improve the performance of a deep autoencoder and to produce a sparser neural network; and in [34] a GA was used to train a network when annotated training data were not available.

Our method is quite different from previous work [32]–[36], in that our work aims at developing CNNs specifically to learn more semantic patterns, a focus which is has not been studied.

III. Methodology

Our work builds on the extensive literature on CNNs in texture recognition [12], but further motivated by the work of Zhou *et al.* [31], which demonstrate a relationship between semantic filters and recognition success. In particular, those CNN convolutional filters showing semantic patterns are regarded as effective at visual recognition, while those showing non-semantic patterns are regarded as being incompletely learned, on which basis the authors claimed that increasing the number of semantic filters improves the recognition performance. As a result, we are inspired to adopt a genetic algorithm (GA) to promote the learning of networks in a global and optimized way, by which we aim to reduce the number of non-semantic filters and to increase the number of semantic ones.

GAs are effective at searching large and complex spaces in an intelligent way to approximately solve global optimization problems. Furthermore, from weak learning theory in pattern recognition [37], using an ensemble of models boosts classification performance, since multiple models capture richer semantic filters than a single model does, thus we propose to adopt three CNN models in our GANet. We will define genetic operations of mutation and crossover, so that we can traverse the search space efficiently, seeking to maximize the number of semantic patterns, thereby successively eliminating non-semantic ones. By using different training sets to train the three networks, we realize further improvements in filter diversities of the hidden layers, increasing the string diversities used for crossover and mutation, which we expect should lead to improved performance.

A. Proposed GANet Network

Our proposed GANet is shown in Fig. 2, highlighting the genetic operations of mutation and crossover.

Mutation is a genetic operator used to maintain genetic diversity from one generation to the next of a population of chromosomes, analogous to biological mutation. The mutation process of an individual involves flipping each bit independently with some probability q. In practice, q is often small (0.05), since setting q too high causes the search to turn into a primitive random search. A modest q allows the good properties of a survived individual to more likely be preserved, while still providing opportunities to explore.

In contrast, *Crossover* is a genetic operator to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, involving a swapping, with probability p, between two individuals, allowing the diversities of the filters in the hidden layers to be improved.

The number of filters used for crossover or mutation is not a trivial question, normally found empirically. Based on our experiments we have chosen 10% of the filters for exchange during crossover and 5% of the filters for mutation. Here, we improve the classical/traditional genetic algorithm (GA) according to our case for crossover and mutation. Specifically, we already have a good initial population used for GA operations since the initial population is encoded from the filters in an off-the-shelf CNN model (VGGNet [24]) pre-trained on a large-scale dataset like ImageNet [26]. We thus decrease the probability for crossover but increase probability for mutation. On the one hand, the probability for crossover in our case is 10% vs. typical value 60% of the classical/traditional GA. In other words, we reduce the probability for crossover because of the good initial population. On the other hand, the probability for mutation in our case is 5% vs. typical value 0.8% of the classical/traditional GA. In the traditional GA, the population size typically contains several hundreds or thousands of possible solutions. For each generation, therefore, one would have several individuals to perform the mutation. In our case, the population size for first layer has only 64 filters, which means we have only 3 individuals used for mutation. In other words, we increased the probability for mutation because of the small population.

To undertake the GA operations, all of the filters in a convolutional/pooling layer are concatenated into one string. For crossover we choose two filter strings U_i and U_j from two nets at random, and then exchange some elements between these two strings. For example, as shown in Fig. 2,

$$\{u_{i1}, u_{i2}, ..., u_{it}\} \text{ from } U_i \text{ are exchanged with } \{u_{j1}, u_{j2}, ..., u_{jt}\} \text{ from } U_j. \quad (1)$$

For mutation we similarly choose two filter strings from two nets, however unlike exchange, mutation uses some elements in one string to replace some in the other. For example, again as shown in Fig. 2, we use elements $\{u_{b1}, u_{b2}, ..., u_{bt}\}$ from U_b to replace certain elements in U_a, but keep U_b unchanged. Note that in all cases the two filter strings for crossover and mutation are from the same layer. In our case, on the one hand, we flip the bit in the string i.e., from semantic elements to be non-semantic elements. Specifically, we encode the non-semantic elements in one string as 0s and semantic elements as 1s. We used 0s in one string to replace 1s in the other string. It is a kind of flipping bits. On the other hand, we did not use 1s to replace 0s because our motivation is to simulate the strategy of dropout [34] in deep network learning, intentionally designed to slow down the learning process, such that the occasional removal of semantic filters leads to more effective overall learning. In addition, we also use the probability to control the bit flipping by randomly selecting 5% bit to perform mutation.

The key question is the selection of filters for crossover and mutation. We have developed the following rules:

- *Crossover* elements are chosen at random from the two networks.
- *Mutation* elements are chosen such that, paradoxically, *non*-semantic filters replace semantic ones. Although the targeted removal of semantic filters feels perverse, the approach is, in fact, analogous to the strategy of dropout [38] in deep network learning, intentionally designed to slow down the learning process, such that the occasional removal of semantic filters leads to more effective overall learning.

What remains to be determined is a criterion by which a filter is judged to be semantic or not. Motivated by the method presented in [31], we visualized the filters and found that filters at the first convolutional layer are typically responsive to simple texture patterns, such as line elements, crosses and corners, whereas deeper layers are associated with more complex patterns having higher level semantics.

Therefore at the first layer each we apply the Local Binary Pattern (LBP) operator [4] to the layer. The basic form of LBP takes as input a local neighborhood around each pixel and thresholds the neighborhood pixels at the value of the central pixel. The resulting binary-valued string is then weighted as follows:

$$LBP(g_c) = \sum_{i=0}^{P-1} 2^i s(g_c - g_i) (2)$$

where the parameter P means the number of the neighbors (e.g., P =8 with neighborhood radius R =1), and g_c is the central pixel. g_c and g_i are the gray-level values at c and i, and $s(A)$ is 1 if $A \geq 0$ and 0 otherwise.

One extension of the original LBP is the uniform patterns: an LBP is 'uniform' if it contains at most two 0-1 or 1-0 transitions when viewed it as a circular bit string (e.g., 11110011 is a uniform pattern). A uniform pattern usually conpresonds to edges, cornor and flat area in textures [4] .In our case, each location in the layer is considered semantic (systematic, non-random) if its corresponding LBP pattern is uniform (i.e., edges, cornor or flat area etc), and likewise non-semantic (irregular, random) if the corresponding LBP pattern is nonuniform [4].

At coarser layers, the distinction is a bit more subtle. We begin with a pre-trained VGGNet, from which (as in [31]) we visualize the filters of each layer and then cluster the filters into two groups (semantic and nonsemantic). The minimal activation of the semantic group is used as the semantic threshold τ_l, which will be a function of layer l. Then, for some input image I, let f_j^l be the output of the jth convolutional filter at the lth layer, that is, such that f_j^l includes the effect of the activation function (here a ReLU — Rectified Linear Unit). A given filter output at position (r, c) and layer l will be considered semantic if $f_j^l(r, c) > \tau_l$.

For each position (r, c) in feature map x_j^l (i.e., the output channel of the j-th convolutional filter at the l-th layer) , we wish to determine whether the region around (r, c) corresponds to semantic behavior. To do this, we search within a 5×5 window centered at (r, c) to count the number of semantic activations, as just defined. If we cannot find k (typically $k = 10$) strong activations, we neglect position (r, c) and move to the next position in the feature map. If we do have at least k activations, we assert filter j to be semantically meaningful, an assessment to be taken into account during mutation and in the final assessment of the resulting network. Note that our aim is to check whether a filter is semantic or not. In the activation map it is easy to find several locations whose activations are larger than the given threshold because textures consists of repeated texels. If we find such one location in the activation map generated by a filter, we regard this filter is semantic. For the other locations on the activation map, we just ignore them.

To learn our semantically driven network, we start with an off the shelf CNN model (VGGNet [26]) pretrained on a large scale dataset like ImageNet [30]. The model is then fine tuned via the genetic strategy of Fig. 2, given texture images. Based on our experiments, we found the fraction of filters showing semantic patterns increasing from 60% in the original VGGNet (fine tuned on textures, but with no GA) to around 70% in GANet (VGGNet after the genetic algorithm).

B. Scale Proposal

Intuitively, the scale change of a texture image clearly affects its appearance, however the direct estimation of global scale is very unstable. In order to achieve scale invariance in texture classification we propose to search a set of scale proposals or candidate scale levels in a given texture image. We then reduce the number of scale proposals by searching among the basic dominant texels that occur most frequently.

Scale Proposal Searching. In general, there can be not only multiple scales, but indeed *continuous* changes in scale within a single image, as was illustrated in Fig. 1 (*d*). A necessary prerequisite for texture classification with extreme scale variations to be successful is that we should find all of the existing scale proposals; to this end, we use the BING algorithm [24] to find candidate texture element (texel) windows, and then compute the scale proposals according to these windows. The rationale behind the BING searcher is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2019.2903300, IEEE Transactions on Image Processing

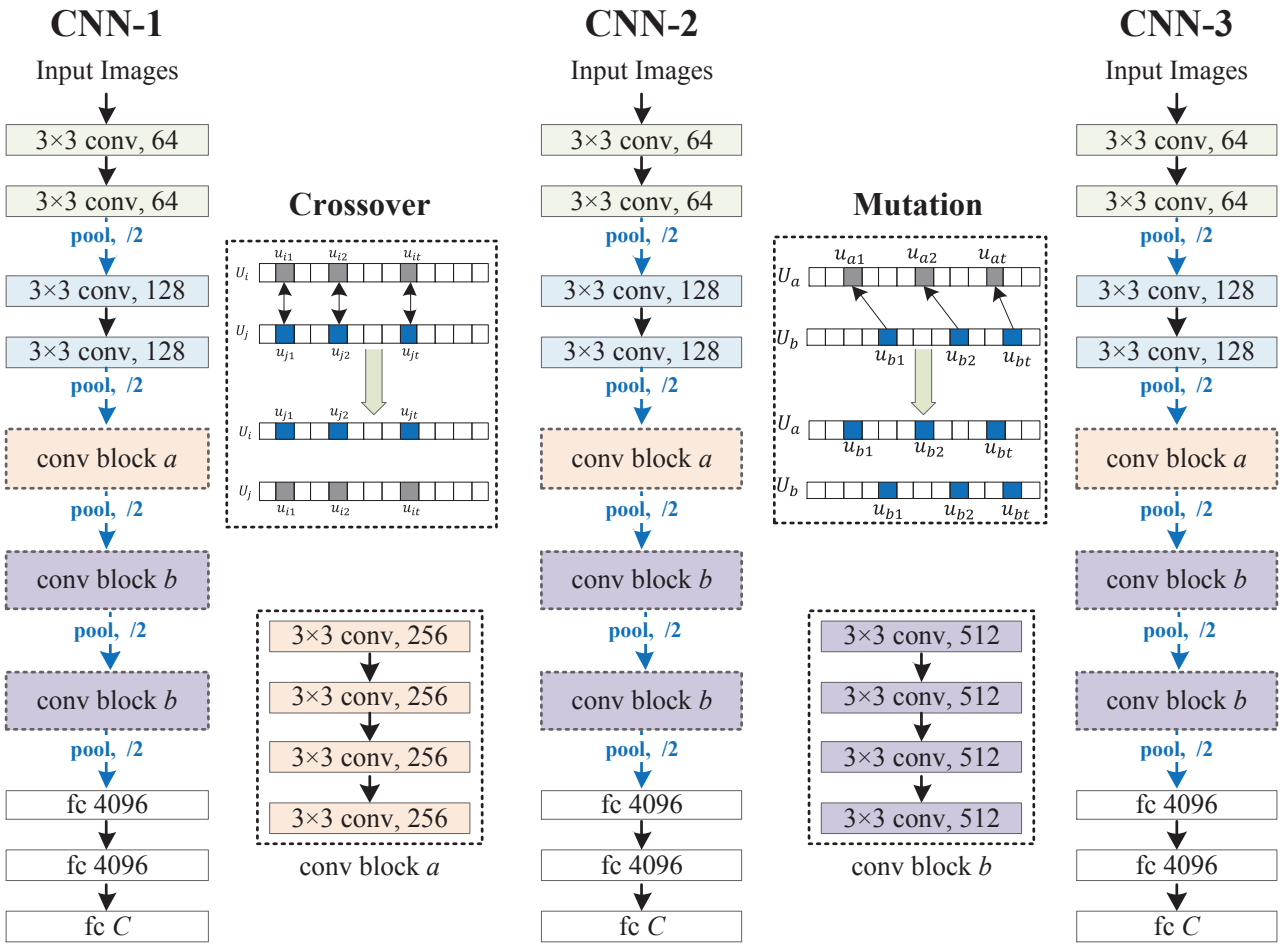


Fig. 2. Architecture of our GANet network. There are three nets CNN-1, CNN-2 and CNN-3, each one being a 19 layer VGG-Net. Here, "3 × 3 conv 256" implies a convolution with filters of size $3 \times 3 \times 256$, where 256 is the number of feature channels in this layer. "fc n" denotes a fully connected layer of size n, where $n = C$ (final layer) is the number of texture classes. The GA includes two operations — crossover and mutation — which are only applied to the hidden filters in the convolutional and pooling layers. By this way, we promote the network learning in a global and optimized way to reduce the number of non-semantic filters and to increase the number of semantic ones. To undertake the GA operations, all of the filters in a convolutional/pooling layer are concatenated into one string. For crossover we choose two filter strings U_i and U_j from two nets at random, and then exchange some elements between these two strings. For example, $\{u_{i1}, u_{i2}, ..., u_{it}\}$ from U_i are exchanged with $\{u_{j1}, u_{j2}, ..., u_{jt}\}$ from U_j. For mutation we similarly choose two filter strings from two nets, and then perform mutation by using some elements in one string to replace some in the other. For example, we use elements $\{u_{b1}, u_{b2}, ..., u_{bt}\}$ from U_b to replace certain elements in U_a, but keep U_b unchanged.

that it searches the scale proposals in real time (300 fps) and returns almost all of the potential texels in an image.

As demonstrated in Fig. 3 (*a*), to be sensitive to a range of scales we resize an input image $\mathbf{I} \in \mathbb{R}^{W_0 \times H_0}$ to a sequence of quantized sizes characterized by scale ratio s. In our experiments, we choose $s = 0.95$ and generate an image pyramid of resized images of sizes $\{(W_0 s^m, H_0 s^n)\}$, to some lower limit (here set to ten pixels), determined by the region of support of the features being extracted, with an 8×8 feature extraction window recommended by Cheng *et al.* [24]. We calculate the *normed gradient* (NG) [1] feature [24] (shown in Fig. 3 (*b*)) of the entire pyramid. Note that we deliberately downsample separately along each axis, to account for textures having different aspect ratios (as illustrated in Fig. 3 (*d*)).

To find texels within a texture image, we scan over its entire image pyramid with an 8×8 BING feature [24]. As shown in Fig. 3 (*c*), at any particular scale level l a number of texels $\Omega = \{T_{l,k}\}$, indexed by location k, could be proposed. To keep the concept of scale clear, every location in the image pyramid is characterized by its rescaling relative to the original image; that is, pyramid image of size $\{(W_0 s^m, H_0 s^n)\}$ is said to be at scale $l = (s^m, s^n)$. We denote all texels found over the entire image pyramid as $\Omega = \{T_{l,k}\}$.

In practice, we typically find thousands of potential texels over an entire image pyramid by using small thresholds for the BING searcher. As shown in Fig. 4, the scales of these texels form the scale proposals for this texture image. Specifically, we denote the set of all the scale proposals

$$\mathbb{S}_{sp} = \{l | T_{l,k} \in \Omega\} \tag{2}$$

as the candidate scale proposal.

Scale Proposal Reduction. The candidate scale proposals $\mathbb{S}_{\text{sp}}$ are usually redundant for an input image. For efficiency consideration, we need to reduce the number of the scale

[1]The normed gradient represents Euclidean norm of the gradient.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2019.2903300, IEEE Transactions on Image Processing

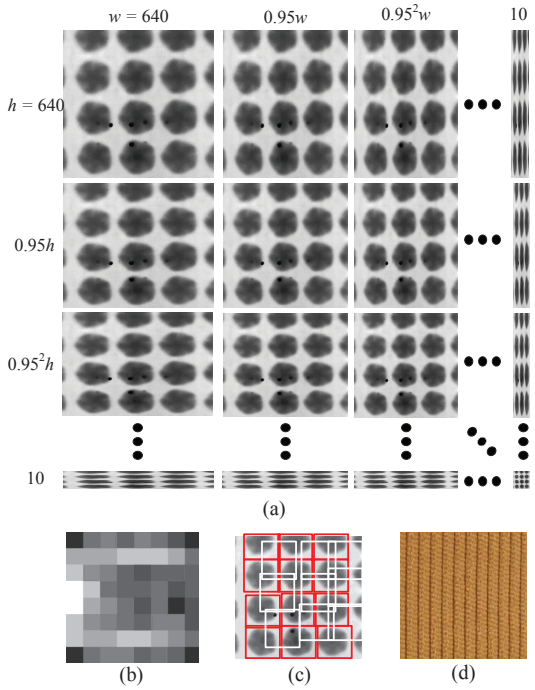


Fig. 3. Scale proposal searching using an image pyramid and the BING searcher [24]: (a) Images are downsized to obtain an image pyramid; (b) A single 64D linear model for selecting texture element proposals based on normed-gradient features; (c) Candidate texture element windows; (d) A texture image whose texels have a significantly different aspect ratio from those in (c).

proposals by finding the basic texels that most frequently occur in the image.

Textures, whether they are regular or stochastic, contain repetitive patterns that exhibit stationary statistics of some sort [20]. Hence, the texels found by the BING searcher are expected to appear repetitively. As shown in Fig. 4, for each texel proposal $T \in \Omega$, we search its similar texels over the texel proposal set from the same scale level. We only keep those texels having sufficiently many similar texels and remove the others from the candidate texel set Ω, with the remaining texels forming a new set Ω^{re}, having a corresponding reduced scale-proposal set $\mathbb{S}_{\text{sp-re}}$, a significant reduction in the number of scale proposals.

Based on our analysis, in order to obtain a good reduced scale proposal set, two more issues have to be taken into consideration: how to evaluate the similarity between two texels, and how many texels to be reserved.

Regarding the similarity measure between two texels T_m and T_n, we propose to compute the distance between the LBP histograms [4] of them. Herein, the rationale behind the Local Binnary Pattern (LBP) is that it works in real time and it achieved state-of-the-art performance for texture analysis compared to other methods [21].

After getting the LBP features for T_m and T_n, we compute their LBP histograms H_m and H_n. In our case, we take P =8 and R =1 and use the uniform patterns for LBP (i.e., $\text{LBP}^{u2}_{8,1}$). We then compute the histogram intersection as the similarity between T_m and T_n.

In terms of which texels in the candidate texel proposal set should survive, for each $T \in \Omega$, we find the set Ω' of similar texels whose similarities are larger than some threshold η, such that candidate T is preserved only if the number of similar texels surpasses threshold K, only keeping the dominant, most frequently occurring texels, essentially those which are more stable and removing noise. The setting of the two threshold parameters η, K will be discussed in Section IV-A.

C. Scale Boundary

As demonstrated in Fig. 1, we argue that the category of a texture image only remains unchanged during some scale interval. In other words, when the scale of a texture image changes significantly, the category of this texture image may also change. The interesting question is the location of the *scale boundary* which separates the two images of the same physical material as different texture classes. Olshausen and Field [39] reconstruct any given image in a sparse way based on a selected group of patches. Inspired by this finding, we similarly develop a patch based method to infer boundary in scale.

Given a texture dataset $\mathbb{S} = \{\mathbb{S}_c\}$ with C classes and for each class $\mathbb{S}_c$, we randomly select 10 images and then compute its basis functions $\mathbb{X}_c = \{x_c\}$ on 16×16-pixel patches, as in [39]. For any image $\mathbf{I} \in \mathbb{S}_c$, we compute its reconstructed image $\hat{\mathbf{I}}$ and the reconstructed error

$$\delta = \|\mathbf{I} - \hat{\mathbf{I}}\| / \|\mathbf{I}\| \tag{3}$$

If $\delta > \xi$, we consider that category of the texture image changes, *i.e.* $\mathbf{I} \notin \mathbb{S}_c$. In our case, we set $\xi = 0.1$ as suggested by [39].

We adopt this approach to group our synthesized image set SForrest $\Theta^+(\mathbb{S}_f)$ and ESVaT (detailed later in Section IV) into subcategories. The process is detailed in Algorithm 1, where dataset SForrest is used as example. After applying Algorithm 1, each texture class in SForrest is regrouped into a number of subcategories, *i.e.* $\Theta^+(\mathbb{S}_f) = \{\mathbb{S}^{syn}_{c,p}\}$, indexed by class c and scale level p, where there are P_c distinct scale levels associated with class c. As a result, dataset SForrest with C original texture classes has been regrouped into $\sum_{c=0}^{C-1} P_c$ categories [2]. All of the images are also manually checked after this automatic regrouping. Likewise, ESVaT is also regrouped with Algorithm 1.

D. Summary of Proposed Framework

Our texture classification pipeline is summarized in Fig. 5, consisting of the following steps:

1) For each image $\mathbf{I}$ in training/testing set $\Theta_{\mathbf{I}}$, we compute its reduced scale proposals $\mathbb{S}_{\text{sp-re}}$ as described in Section III-B.
2) We downsize each image $\mathbf{I}$ to obtain its image pyramid $\{\mathbf{I}_p\}$ using its reduced scale proposals $\mathbb{S}_{\text{sp-re}}$. For the downsized images, we let $\mathbb{I} = \mathbf{I} \bigcup \{\mathbf{I}_p\}$ be the expanded

[2]In fact we manually combine some subsets in $\{\{\mathbb{S}^{syn}_{c,p}\}_p\}_c$ as one category since some of them are basically textureless and only show a region of a certain color.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2019.2903300, IEEE Transactions on Image Processing

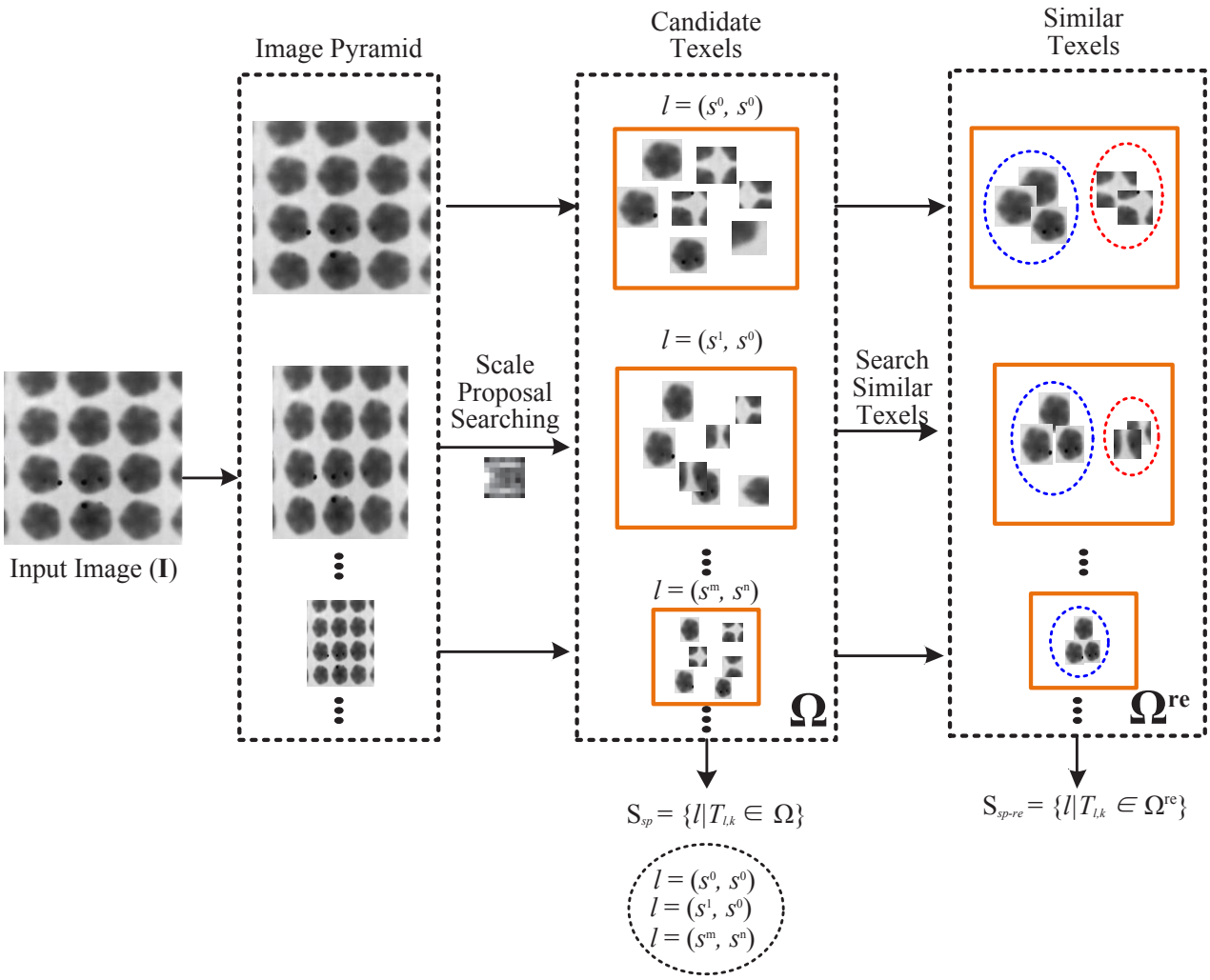


Fig. 4. Pipeline for generating scale proposal. Given a texture, we first generate its image pyramid using the scale (s^m, s^n) as shown in Fig. 3, from which we search the candidate texels. We then collect the scale proposals by which we find the candidate texels in the image pyramid. For each candidate texel, we find its similar texles. For those candidate texels who have more than K similar texles, we keep them and collect their scales to form the reduced scales proposals S_{sp-re}.

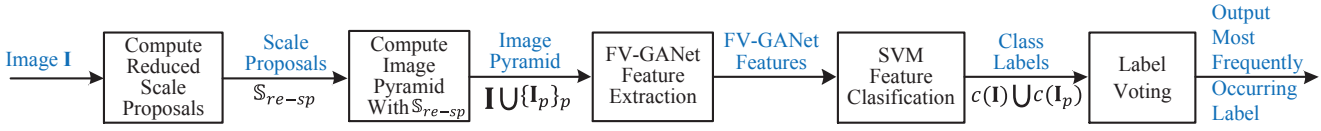


Fig. 5. The proposed texture classification pipeline

set for image **I**. Thus, we have a new training/testing set $\Theta_{\mathbf{I}}^{+} = \{\mathbb{I}\}$.

3) For each image in the new training/testing set, we use the proposed FV-GANet to extract global texture feature representation, following FV-CNN [12].
4) The extracted FV-GANet features are classified using an Support Vector Machine (SVM) classifier. For each image **I**, if its reduced scale proposals $\mathbb{S}_{\text{sp-re}}$ has M scale levels (*i.e.* $|\mathbb{S}_{\text{sp-re}}| = M$), then we have $M + 1$ category labels for it after classification. Image **I** will be assigned the category label which occurs the most frequently.

In our case, we use SVM for classification, which constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space [40]. It can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

The evaluation of the texture classification performance in step 3 uses FV-CNN [12] for texture feature description. FV-CNN truncates a CNN network and regards the last

convolutional layer of a CNN as a filter bank, performing orderless pooling of CNN descriptors using the Fisher Vector, as is commonly done in standard bag of words approaches. FV pools local features densely within the described regions removing global spatial information, and is therefore more apt at describing textures than objects. The pooled convolutional features are extracted immediately after the last linear filtering operator and are not normalized. These features are pooled into a FV representation with 64 Gaussian components. FV-CNN is remarkably flexible and effective. First, the convolutional layers behaving like non-linear filter banks are better local texture descriptors than the fully connected layers, which may be useful for representing the overall shape of an object. Second, the FV pooling encoder is suitable for texture description since it is orderless and multi-scale. Third, it avoids expensive resizing of input images since any image size can be processed by convolutional layers.

We have to emphasize that the training set is divided into subcategories via the scale boundary search algorithm presented in Algorithm 1. During classification, a testing image is considered being correctly classified only it is assigned the correct subcategory label.

Algorithm 1: Divide each texture class in the synthesized SForrest into subcategories to find its scale boundary

Input: The synthesized dataset SForrest $\Theta^+(\mathbb{S}_f)$
Output: Regrouped SForrest $\Theta^+(\mathbb{S}_f) = \{\{\mathbb{S}^{syn}_{c,p}\}_p\}_c$
for *each texture class c in SForrest* **do**
  Let Θ^+_c be all the samples of class c in SForrest;
  Set $p = 0$;
  if $\Theta^+_c \neq \varnothing$ **then**
    1. Randomly select ten texture samples with the *largest* scale level $\mathbb{I}^{\text{se}}_c \subset \Theta^+_c$;
    2. Compute basis functions $\mathbb{X}_{c,p}$ based on $\mathbb{I}^{\text{se}}_c$;
    3. Reconstruct all the samples in Θ^+_c with basis functions $\mathbb{X}_{c,p}$;
    4. Determine the samples whose reconstructed error is less than ξ and denote them as $\mathbb{S}^{syn}_{c,p}$: $\mathbb{S}^{syn}_{c,p} = \{\mathbf{I}_k | \frac{\|\mathbf{I}_k - \hat{\mathbf{I}}_k\|}{\|\mathbf{I}_k\|} < \xi, \mathbf{I}_k \in \Theta^+_c\}$;
    5. Update $\Theta^+_c = \Theta^+_c - \mathbb{S}^{syn}_{c,p}$;
    6. $p = p + 1$.
  end
end

IV. Datasets and Experimental Setup

We test the proposed framework on four datasets: the synthesized dataset SForrest derived from Forrest [41], ESVaT, KTHTIPS2b [15] and OS (OS) [16]. Some example texture images from Forrest, KTHTIPS2b and OS are shown in Fig. 6 and some examples from ESVaT are shown in Fig. 1.

SForrest is synthesized based on the Forrest dataset $\mathbb{S}_f$, which contains 17 texture classes and 935 images captured in the wild. The method for synthesizing SForrest is as follows. For each image $\mathbf{I}_i \in \mathbb{S}_f$, we firstly generate an image pyramid $\Theta_{\mathbf{I}_i} = \{\mathbf{I}_{i,p}\}$ with scale $s = 0.95$ using the method detailed at the beginning of Section III-B and illustrated in Fig. 3 (*a*). We synthesize new images $\Theta_{\mathbf{J}_i} = \{\mathbf{J}_{i,p}\}$ based on $\Theta_{\mathbf{I}_i}$, in that for each $\mathbf{I}_{i,p} \in \Theta_{\mathbf{I}_i}$, we stitch several reduplications of $\mathbf{I}_{i,p}$ together to generate a larger image $\mathbf{J}_{i,p}$, which is cropped at random, if needed, to have the same size as $\mathbf{I}_i$. We define $\Theta(\mathbb{S}_f) = \{\Theta_{\mathbf{I}_i}\}_i$ to be the image pyramids of all images in image set $\mathbb{S}_f$, and $\Theta^+(\mathbb{S}_f)$ to be the combined image set $\mathbb{S}_f \bigcup \{\Theta_{\mathbf{J}_i}\}_i$. $\Theta^+(\mathbb{S}_f)$ is our final synthesized dataset. For image editing, we use the method proposed by Perez *et al.* [42], introduced for the seamless editing of image regions.

ESVaT is composed of 15,747 texture images from 15 material categories[3], each of which has extreme scale variations and is further annotated to several subcategories by the approach detailed in Section III-C. **KTHTIPS2b** [15] has 11 texture categories and four physical samples per category. Each physical sample is imaged with 3 viewing angles, 4 illuminants and 9 different scales to obtain different images. From **OS** [16] we use the same dataset as in [12]. It has 53,915 annotated material segments in 10,422 images spanning 22 different classes.

Most scale variations in KTHTIPS2b and OS are small compared with SForrest and ESVaT. SForrest and ESVaT were specifically designed to test texture classification under extreme scale variations. However we do continue to test the performance of our framework on KTHTIPS2b and OS to show that the proposed can give significantly improved performance, even though our method is specifically designed for extreme scale variations.

For SForrest, half of the class samples were selected at random for training and the remaining half for testing, and results are reported over ten random partitions of training and testing sets. For ESVaT, we split images evenly into training, validation and testing subsets. For KTHTIPS2b, one sample is available for training and the remaining three for testing, following [12]. For OS, we also use the same setup as in [12]. SForrest and ESVaT are regrouped per Algorithm 1. KTHTIPS2b and OS are augmented by building the image pyramids using the scale proposals as discussed in Sections III-B, but without regrouping.

Implementation Details. We finetune VGGNet using CUReT [14] and UIUC [2]. Our finetuning is carried out for the whole network. The original training set of CUReT and UIUC are expanded as follows. For each image $\mathbf{I}$ in the training set $\Phi = \{\mathbf{I}\}$, we compute its reduced scale proposal set $\mathbb{S}_{\text{sp-re}}$, then downsize it to obtain $M = |\mathbb{S}_{\text{sp-re}}|$ downsized images $\mathbb{I}^+ = \{\mathbf{I}_p\}$ using its scale proposals $\mathbb{S}_{\text{sp-re}}$. Thus, we have a new training set $\Phi^+ = \mathbb{I}^+ \bigcup \Phi$. Note that each image in $\mathbb{I}^+$ has the same class label as $\mathbf{I}$. We further perform data augmentation and window cropping on the new training set, following the method in [26]. By these means, the number of training samples increases significantly, to 3.96 million, which are split at random into three even subsets ($\mathbb{S}_1$, $\mathbb{S}_2$ and $\mathbb{S}_3$), which are then used to train the three CNN models shown in Fig. 2 respectively.

In our GANet, we train three CNNs to perform crossover and mutation for the filter strings from the same layer because they show similar semantic features. Since we propose to use

[3]bark, bubble, brick, carpet, concrete, fabric, grass, granite, laminate, plastic, stone, tile, wood, wheat and tree

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2019.2903300, IEEE Transactions on Image Processing

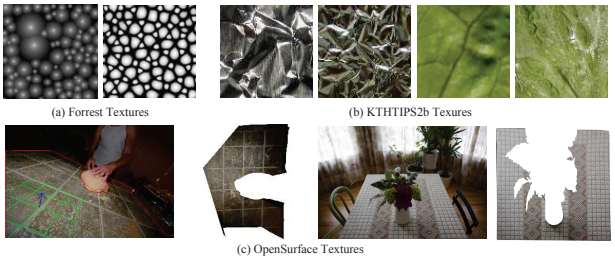


Fig. 6. Some example textures from (a) Forrest, (b) KTHTIPS2b and (c) OS (Original images and their corresponding annotated texture segments).

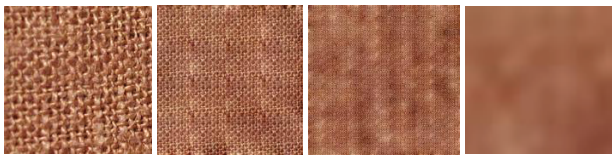


Fig. 7. Example of a synthesized texture at different scales.

TABLE I
CLASSIFICATION SCORES OF EACH CNN IN GANET ON THE SYNTHESIZED SFORREST DATASET.

Methods	SForrest
FV-CNN ($\mathbb{S}_1$)	85.3%
FV-CNN ($\mathbb{S}_1+\mathbb{S}_2+\mathbb{S}_3$)	87.3%
FV-CNN-GA-1 ($\mathbb{S}_1$)	88.1%
FV-CNN-GA-2 ($\mathbb{S}_1$)	87.8%
FV-CNN-GA-3 ($\mathbb{S}_2$)	88.3%
FV-GANet	**90.4**%

TABLE II
PERFORMANCE EVALUATION OF EACH COMPONENT OF OUR PROPOSED SCALE SEARCHER ON THE SYNTHESIZED SFORREST DATASET.

Dataset	SForrest			
Methods	Without GA	Crossover	Mutation	Crossover+Mutation
FV-CNN-GA-1 ($\mathbb{S}_1$)	85.3%	87.3%	87.1%	88.1%
FV-CNN-GA-2 ($\mathbb{S}_2$)	84.9%	87.1%	86.8%	87.8%
FV-CNN-GA-3 ($\mathbb{S}_3$)	85.4%	87.5%	87.4%	88.3%

three CNNs, we have three SVM classifiers. We use these three classifiers to vote for texture categories. Following the work in [12], we also normalize descriptors by $L2$ norm and let the learning constant be $C = 1$.

The network is implemented following the parameter setting of VGG net [27]. Specifically, the hyper parameters of this network include: mini-batch size (4), learning rate (1e-6), momentum (0.9), weight decay (0.002), and maximum number of training iterations (600,000). For the cost function, we use the same as that of VGG net [27].

A. Experimental Tests

GANet: The classification results on SForrest are listed in Table I. FV-CNN means the original VGGNet. FV-CNN ($\mathbb{S}_1$) means that FV-CNN is finetuned using $\mathbb{S}_1$. FV-CNN ($\mathbb{S}_1+\mathbb{S}_2+\mathbb{S}_3$) means that FV-CNN is finetuned using all the three subset $\mathbb{S}_1+\mathbb{S}_2+\mathbb{S}_3$. FV-CNN-GA-*n* ($n = 1, 2, 3$) are the three CNN models finetuned using GA. FV-GANet is to combine the three FV-CNN-GA-*n* models by voting for texture categories. The majority vote algorithm is an algorithm for finding the majority of a sequence of elements using linear time and constant space. In its simplest form, the algorithm finds a majority element, if there is one: that is, an element that occurs repeatedly for more than half of the elements of the input [43].

From Table I, we can observe that including GA improves performance, since the individual models FV-CNN-GA-*n* outperform both FV-CNN ($\mathbb{S}_1$) and FV-CNN ($\mathbb{S}_1+\mathbb{S}_2+\mathbb{S}_3$). When combined, one can observe that FV-GANet is significantly better than FV-CNN.

The performance evaluation how crossover and mutation change the performance of the proposed method is shown Table II. We used three subsets to train three CNNs separately. we can observe that including GA components (crossover and/or mutation) improves performance. In other words, FV-CNN-GA-n(Sn) with the two components (crossover+mutation) works better than FV-CNN-GA-n(Sn) with only one component (crossover or mutation). However, both of them, i.e., FV-CNN-GA-n(Sn) with one or two components (crossover and/or mutation) works better than FV-CNN-GA-n (Sn) without GA components. In addition, from this table, we can observe that crossover works slightly better than mutation. One reason might be that we get two new children strings for crossover but only one new child string for mutation. Thus, the former brings more diversity into the network.

TABLE III
PERFORMANCE EVALUATION OF EACH COMPONENT OF OUR PROPOSED FRAMEWORK ON SFORREST AND KTHTIPS2B.

Methods	SForrest	KTHTIPS-2b
FV-GANet	90.4%	82.6%
FV-GANet+SP	91.5%	83.0%
FV-GANet+SP+RE	96.3%	86.7%

TABLE IV
PERFORMANCE EVALUATION USING DIFFERENT SCALE PROPOSAL SEARCHERS ON SFORREST AND KTHTIPS2B.

Methods	SForrest	KTHTIPS2b
FV-GANet	90.4%	82.6%
FV-GANet+FFT	89.8%	81.6%
FV-GANet+Lindeberg	92.1%	83.3%
FV-GANet+SP+RE	96.3%	86.7%

Component of Scale Searcher: Scale searcher results are shown in Table III. FV-GANet means that we only use FV-GANet. FV-GANet+SP means we use FV-GANet and the Scale Proposals (SP) component. FV-GANet+SP+RE means we use FV-GANet and the REduced SP. The pipeline about how to get the training set for these three cases is shown in Fig. 8.

The results in Table III clearly show that the combination of FV-GANet, SP and RE improves the classification performance significantly. It demonstrates that texture classification in extreme scale variations can benefit from SP. After we combine RE, the number of scale proposals drops significantly since RE discards many errors in scale proposals. To check the accuracy of the predicted scale levels for each image by FV-GANet+SP+RE, we use the mode of the predicted scale levels

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2019.2903300, IEEE Transactions on Image Processing

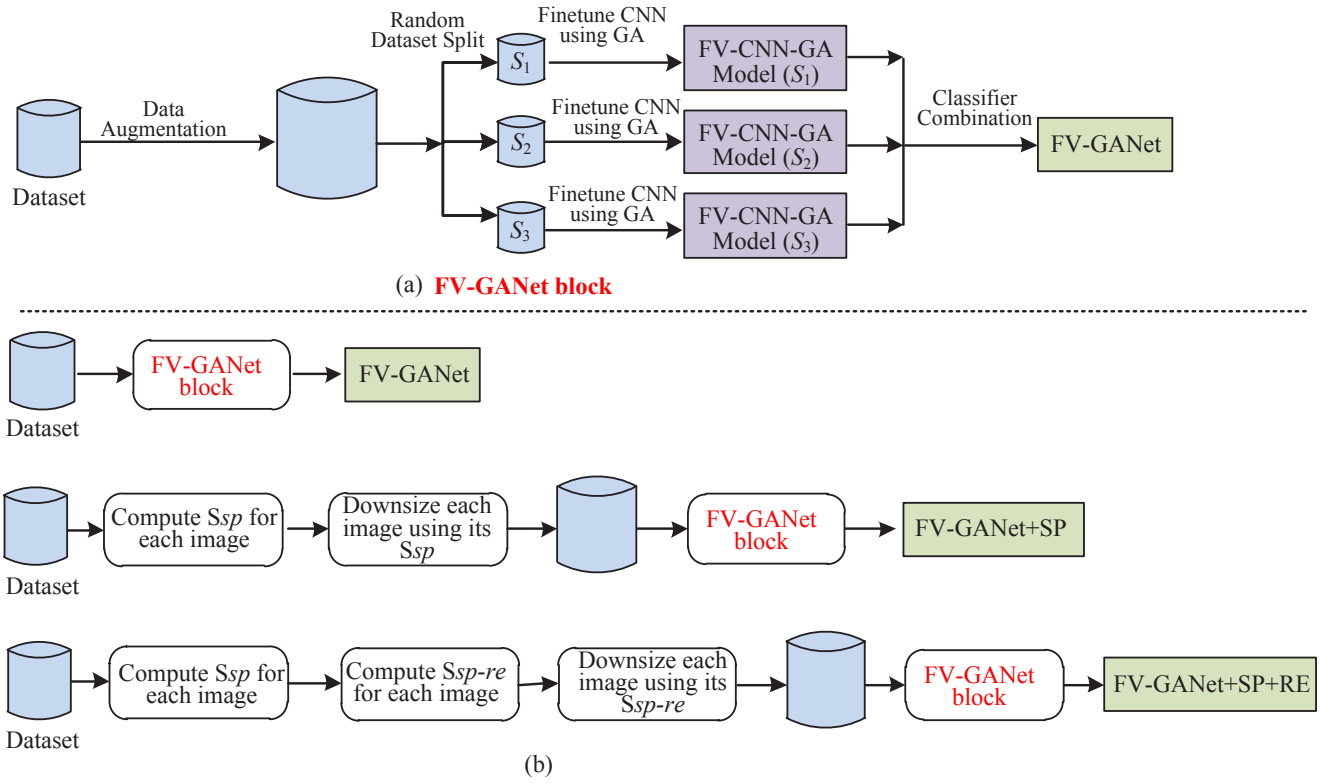


Fig. 8. (Illustration of training set for network training. (a) FV-GANet block. (b) training set using for FV-GANet, FV- GANet+SP, and FV-GANet+SP+RE.)

to compare with ground truth; the accuracy for the synthesized dataset is 97.4%, clearly demonstrating the accuracy of texture classification in the extreme scales of SForrest.

Different SP Searchers. We compare our SP method with other possible SP searchers such as Fast Fourier Transform (FFT) and the method by Lindeberg [22]. As shown in Fig. 9, we use the same pipeline as FV-GANet+SP+RE shown in Fig. 8. The difference is that FV- GANet+FFT using FFT to find the texture period instead of the method proposed in Section III-B. To find the texture period by FFT, we use the method proposed in Matsuyama et al. [44]. Results are shown in Table IV, which clearly demonstrate that our method works the best. The reason that BING works better than Lindeberg [22] is that BING return almost all of the potential texels in an image. Although BING might not be the best method to find texels without redundancy, these redundant texel candidates satisfy the prerequisite for texture classification with extreme scale variations to be successful, i.e., finding all of the existing scale proposals. In addition, BING run in real time (300 fps), which speeds up the training of a network.

One possible reason that FV-GANet+FFT works poorly is that the texture images in the test set have other uninformative variations, such as illumination and rotation changes, besides scale variations.

Parameter Evaluation: In our approach, we have two important threshold parameters: η, the threshold similarity measure of two texels, and K, the number of similar texels of a candidate texel, as discussed in Section III-B. For parameter η, we use the following statistical value. Starting with dataset CUReT having 61 texture classes $\mathbb{S} = \{\mathbb{S}_1, ..., \mathbb{S}_C\}$, we adopt LBP and nearest neighbor to classify the textures in CUReT. For those correctly classified textures in each class we compute the image-wise similarity based on LBP histograms, letting ϕ_i denote the minimal similarity in class $\mathbb{S}_i$, and then setting $\eta = \min\{\phi_i\}$. We find that although η is derived based on CUReT it works well for other datasets.

For parameter K, we determine its value empirically as shown in Fig. 10. The performance (and computational complexity) of our method increases with K, with performance very much levelling off at $K = 20$, so we have set $K = 20$ for all experiments.

Results for KTHTIPS2b: The results for KTHTIPS2b are

TABLE V
A COMPARISON OF OUR PROPOSED METHOD WITH THE STATE OF THE ART IN TEXTURE DESCRIPTORS.

Methods	SForrest (%)	KTHTIPS2b (%)	OS (%)	ESVaT (%)
LBP [4]	61.8	49.9	35.4	40.5
SIFT [45]	63.4	52.7	38.9	39.1
IFV+DeCAF [6]	78.4	77.5	55.6	53.3
PR-proj [30]	73.7	72.6	53.8	52.7
RAID-G [8]	N/A	81.3	61.1	59.7
FV-CNN [12]	85.3	82.1	60.3	56.2
BCNN [33]	81.6	79.0	56.7	54.8
SWM [46]	67.8	64.4	38.4	42.5
DRLBP [47]	65.1	65.7	39.5	41.4
DeepTen [48]	91.6	84.5	67.9	55.7
FASON [49]	89.5	76.5	64.7	51.6
Ours	**96.3**	**86.7**	**68.4**	**67.9**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2019.2903300, IEEE Transactions on Image Processing

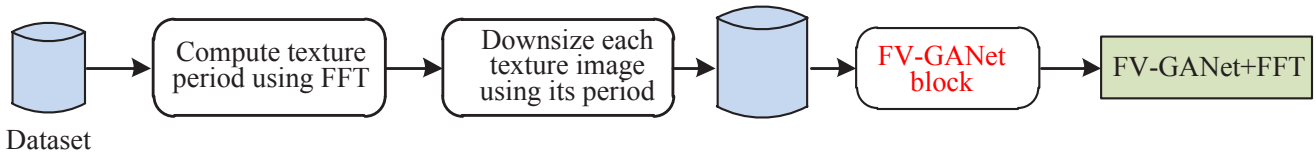


Fig. 9. Illustration of training set for training the network FV-GANet+FFT, where FV-GANet block is shown in Fig. 8 (a).

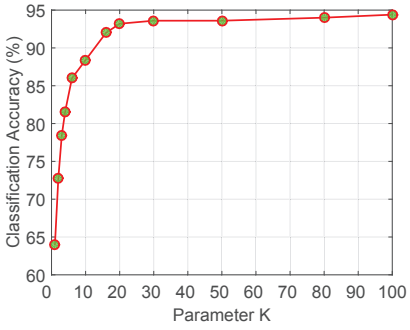


Fig. 10. Classification performance as a function of paramter K.

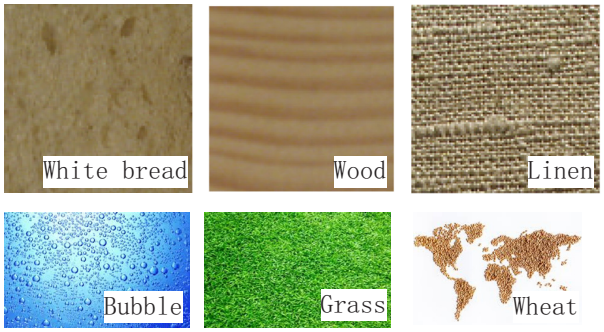


Fig. 11. Example of a synthesized texture at different scales.

shown in Table IV. One can observe that the performance improvement for KTHTIPS2b is similar to that for the SForrest. This demonstrates that our scale proposal approach generalizes to improving the texture classification in KTHTIPS2b which has small scale variations. Note that although images in KTHTIPS2b are spread over nine scales, we can find that there also several scale variations within one image.

V. Comparative Evaluation

For comparison purposes, Table V compares our proposed approach with recent state of the art texture features including LBP [4], SIFT [45], IFV+DeCAF [6], PR-proj [30], RAID-G [8] and B-CNN [33]. For methods IFV+DeCAF, PR-proj and FV-CNN, we adopt the code provided by the original authors. For LBP and SIFT we use our own implementation. Note that IFV+DeCAF, PR-proj and FV-CNN are all fine-tuned using the same training set as our approach. The results of RAID-G [8] and B-CNN [33] are quoted directly from their original papers.

From Table V, we can see that our proposed approach achieves consistently and significantly better results than all methods in comparison on all four evaluated datasets, particularly providing a significant performance margin in excess of 10% on SForrest and ESVaT over previous state of the art, almost certainly because of the extreme scale variations present in those datasets, relative to the more modest gains in KTHTIPS2b, which has less extensive scale variations. Some texture classification example are shown in Fig. 11. Although some of textures are quite challenging, our method classifies them correctly.

In Table V, we can find the two methods SWM and DRLBP does not work so well for ESVaT although they perform very well for rotated textures [46], [47] . For example, SWM got 42.5% and DRLBP got 41.4% compared to 67.9% by our method. It is because both SWM and DRLBP are designed for rotation invariation and ESVaT is a dataset showing extreme scale variation. In addition, we also show the experimental comparsion between our methods and DeepTen [48] and FASON [49]. From Table V, we can find that both DeepTen and FASON achieve quite good perfomrance for KTHTIPS-2b but the performance for ESVat dataset is also not so good compared to our method. It is because that the dataset ESVaT is a dataset showing extreme scale variation and most of existing methods can not work so well. It is also verified that developing a new method for extreme-scale-variation texture analysis is an important issue.

As we mentioned in Section II, existing CNN architectures are not robust to appearance variations such as rotation, scale and noise. Specifically, for the scale variation, as shown in Table V, both FV-CNN and IFV+DeCAF are CNN architectures. They got very good performance for KTHTIPS-2b (82.1% and 77.5%) but not so well for ESVaT (56.2% and 53.3%). The major difference between KTHTIPS-2b and ESVaT is the extreme scale variation in ESVaT. For the noise variations, please refer to [7]. In [7], Liu et al. perform extensive experiments to evaluate the perfomance variations of CNN architectures when textures are degraged by noise. For the rotation variation, CNN architectures have the capability of processing transforms including small rotations. Such capability is endowed with the inherent properties of convolutional operations, redundant convolutional filters, and hierarchical spatial pooling. However, their ability in handling significant local and global image rotations remains limited [50], [51]. Zhou et al proposed the Oriented Response Networks for rotation robustness, but it need extra rotating filters.

VI. Conclusions

Overall, we have proposed a highly effective framework for recognizing textures with extreme scale variations, first searching scale proposals and then discarding errors in scale proposals by exploring dominant texture primitives. We have proposed a novel GANet network for better texture feature learning. Extensive experiments on four challenging texture benchmarks show that the proposed framework works much better than existing methods, especially for those textures with extreme scale variations. In addition, our method is efficient since the scale proposal searching and reducing methods are very fast.

References

[1] M. Pietikäinen, A. Hadid, G. Zhao, and T. Ahonen, *Computer vision using local binary patterns*. London, UK: Springer, 2011.

[2] S. Lazebnik, C. Schmid, and J. Ponce, "A sparse texture representation using local affine regions," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1265–1278, 2005.

[3] L. Liu, P. Fieguth, Y. Guo, X.Wang, and M. Pietikäinen, "Local binary features for texture classification: Taxonomy and experimental study," *Pattern Recognition*, vol. 62, pp. 135–160, 2017.

[4] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, July 2002.

[5] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[6] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3606–3613.

[7] L. Liu, P. Fieguth, X. Wang, M. Pietikäinen, and D. Hu, "Evaluation of lbp and deep texture descriptors with a new robustness benchmark," in *European Conference on Computer Vision*, 2016, pp. 69–86.

[8] Q. Wang, P. Li, W. Zuo, and L. Zhang, "RAID-G: Robust estimation of approximate infinite dimensional gaussian with application to material recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4433–4441.

[9] L. Liu, S. Lao, P. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Median robust extended local binary pattern for texture classification," *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1368–1381, 2016.

[10] Y. Dong, J. Feng, L. Liang, L. Zheng, and Q. Wu, "Multiscale sampling based texture image classification," in *IEEE Signal Processing Letters*, 2017, pp. 614—-618.

[11] Y. Xu, X. Yang, H. Ling, and H. Ji, "A new texture descriptor using multifractal analysis in multiorientation wavelet pyramid," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 161–168.

[12] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3828–3836.

[13] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*. New York, USA: Dover Publications, 1966.

[14] K. Dana, B. V. Ginneken, S. Nayar, and J. Koenderink, "Reflectance and texture of real-world surfaces," *ACM Transactions on Graphics*, vol. 18, no. 1, pp. 1–34, 1999.

[15] B. Caputo, E. Hayman, and P. Mallikarjuna, "Class-specific material categorization," in *IEEE International Conference on Computer Vision*, 2005, pp. 1597–1604.

[16] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Opensurfaces: A richly annotated catalog of surface appearance," *ACM Transactions on Graphics*, vol. 32, no. 4, 2013.

[17] L. Liu and P. Fieguth, "Texture classification from random features," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 3, pp. 574 –586, March 2012.

[18] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: a comprehensive study," *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[19] B. Julesz, "Textons, the elements of texture perception, and their interactions," *Nature*, vol. 290, no. 5802, pp. 91–97, 1981.

[20] X. Xie and M. Mirmehdi, *A galaxy of texture features–Handbook of texture analysis*. London, UK: Imperial College Press, 2008.

[21] L. Liu, J. Chen, P. Fieguth, G. Zhao, R. Chellappa, and M. Pietikainen, "From bow to cnn: Two decades of texture representation for texture classification," in *International Journal of Computer Vision*, 2018.

[22] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 79–116, 1998.

[23] M. Mirmehdi and M. Petrou, "Segmentation of color textures," *IEEE transactions on pattern analysis and machine intelligence*, vol. 22, no. 2, pp. 142–159, 2000.

[24] M. Cheng, Z. Zhang, W. Lin, and P. Torr, "BING: Binarized normed gradients for objectness estimation at 300fps," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3286–3293.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[27] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Annual Conference on Neural Information Processing Systems*, 2012, pp. 1097–1105.

[28] L. Zhang and S. Rusinkiewicz, "Learning to detect features in texture images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[29] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, and J. Hays, "Texturegan: Controlling deep image synthesis with texture patches," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[30] K. Simonyan, A. Vedaldi, and A. Zisserman, "Learning local feature descriptors using convex optimisation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 8, pp. 1573–1585, 2014.

[31] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene cnns," in *International Conference on Learning Representations*, 2014.

[32] O. David and I. Greental, "Genetic algorithms for evolving deep neural networks," in *Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 1451–1452.

[33] J. Lamos-Sweeney, "Deep learning using genetic algorithms," *Thesis of Rochester Institute of Technology*, 2012.

[34] C. Steininger, "Genetic algorithms with deep learning for robot navigation," *Thesis*, 2016.

[35] L. Xie and A. Yuille, "Genetic CNN," in *IEEE International Conference on Computer Vision*, 2017.

[36] S. Ding, H. Li, C. Su, J. Yu, and F. Jin, "Evolutionary artificial neural networks: A review," *Artificial Intelligence Review*, vol. 39, no. 3, pp. 251–260, 2013.

[37] A. Webb and K. Copsey, *Statistical Pattern Recognition (Third Edition)*. Wiley, 2011.

[38] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[39] B. Olshausen and D. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[40] C. Cortes and V. N. Vapnik, "Support-vector networks," in *Machine Learning*, 1995, pp. 273—-297.

[41] "The forrest dataset," http://textures.forrest.cz/.

[42] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *ACM Transactions on Graphics*, vol. 22, no. 3, 2003, pp. 313–318.

[43] R. S. Boyer and J. S. Moore, "Mjrty - a fast majority vote algorithm," in *The Netherlands: Kluwer Academic Publishers*, 1991, pp. 105—-117.

[44] T. Matsuyama, S. Miura, , and M. Nagao, "Structural analysis of natural textures by fourier transformation," in *Computer Vision, Graphics, and Image Processing*, 1983, pp. 347–362.

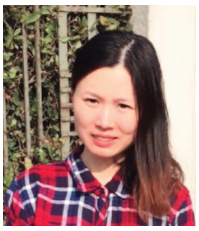
[45] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[46] A. Depeursinge, Z. Püspöki, J. P. Ward, and M. Unser, "Steerable wavelet machines (swm): Learning moving frames for texture classification," in *IEEE Transactions on Image Processing*, 2017.

[47] R. Mehta and K. Egiazarian, "Dominant rotated local binary patterns (DRLBP) for texture classification," in *Pattern Recognition Letters*, 2016, pp. 16–22.

[48] H. Zhang, J. Xue, and K. Dana, "Deep ten: Texture encoding network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[49] X. Dai, J. Y.-H. Ng, and L. S. Davis, "Fason: First and second order information fusion network for texture recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
[50] Y. Zhou, Q. Ye, Q. Qiu, , and J. Jiao, "Oriented response networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
[51] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikainen, "Deep learning for generic object detection: A survey," in *https://arxiv.org/abs/1809.02165*, 2018.



Li Liu received the BSc degree in communication engineering, the MSc degree in photogrammetry and remote sensing and the Ph.D. degree in information and communication engineering from the National University of Defense Technology (NUDT), China, in 2003, 2005 and 2012, respectively. She joined the faculty at NUDT in 2012, where she is an Associate Professor with the College of System Engineering. During her PhD study, she spent more than two years as a Visiting Student at the University of Waterloo, Canada, from 2008 to 2010. From 2015 to 2016, she spent ten months visiting the Multimedia Laboratory at the Chinese University of Hong Kong. From 2016 to 2018, she worked as a senior researcher at the Machine Vision Group at the University of Oulu, Finland. She was a cochair of five International Workshops at CVPR, ICCV, ECCV and ACCV. She is going to lecture a tutorial at CVPR'19. She was a guest editor of special issues for IEEE TPAMI and IJCV. Her current research interests include facial behavior analysis, texture analysis, image classification, object detection and recognition. She is a Senior Member of IEEE.

Jie Chen received his M.S. and Ph.D. degrees from Harbin Institute of Technology, China, in 2002 and 2007, respectively. He joined Peng Cheng Laboratory, China since 2018. He has been a senior researcher in the Center for Machine Vision and Signal Analysis at the University of Oulu, Finland since 2007. In 2012 and 2015, he visited the Computer Vision Laboratory at the University of Maryland and School of Electrical and Computer Engineering at the Duke University respectively. Dr. Chen was a cochair of International Workshops at ACCV, CVPR, and ICCV. He was a guest editor of special issues for IEEE TPAMI, IJCV and Neurocomputing. His research interests include pattern recognition, computer vision, machine learning, dynamic texture, deep learning, and medical image analysis. He is an Associate Editor of The Visual Computer. He is a member of the IEEE.

Paul Fieguth received the B.A.Sc. degree from the University of Waterloo, Ontario, Canada, in 1991 and the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, in 1995, both degrees in Electrical Engineering. He joined the faculty at the University of Waterloo in 1996, where he is currently a Professor in Systems Design Engineering. He has held visiting appointments at the University of Heidelberg in Germany, at INRIA/Sophia in France, at the Cambridge Research Laboratory in Boston, at Oxford University and the Rutherford Appleton Laboratory in England, and with postdoctoral positions in Computer Science at the University of Toronto and in Information and Decision Systems at MIT. His research interests include statistical signal and image processing, hierarchical algorithms, data fusion, and the interdisciplinary applications of such methods, particularly to remote sensing.

Guoying Zhao received the Ph.D. degree in computer science from the Chinese Academy of Sciences, Beijing, China, in 2005. She is a Professor with the Center for Machine Vision and Signal Analysis, University of Oulu, Oulu, Finland, where she has been a Senior Researcher since 2005 and an Associate Professor since 2014. In 2011, she was selected to the highly competitive Academy Research Fellow position. She was a Nokia Visiting Professor in 2016. She has authored or coauthored more than 180 papers in journals and conferences. Her papers have currently more than 9200 citations in Google Scholar (H-index 43). She is a CoPublicity Chair for the 2018 IEEE International Conference on Automatic Face and Gesture Recognition, has served as the Area Chair for several conferences, and is an Associate Editor for Pattern Recognition, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and Image and Vision Computing. She has lectured tutorials at the 2006 International Conference on Pattern Recognition, the 2009 IEEE International Conference on Computer Vision, the 2013 Scandinavian Conference on Image Analysis, and the 2018 IEEE International Conference on Automatic Face and Gesture Recognition, and has authored/edited three books and eight special issues in journals. She was a CoChair of the European Conference on Computer Vision, the IEEE International Conference on Computer Vision, the IEEE Conference on Computer Vision and Pattern Recognition, the Asian Conference on Computer Vision, and the British Machine Vision Conference. Her research has been reported by Finnish TV programs, newspapers, and MIT Technology Review. Her current research interests include image and video descriptors, facial-expression and microexpression recognition, gait analysis, dynamic-texture recognition, human motion analysis, and person identification.

Xilin Chen received the B.S., M.S., and Ph.D. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 1988,1991, and 1994, respectively. He was a professor with the Harbin Institute of Technology from 1999 to 2005. He was a visiting scholar with Carnegie Mellon University, Pittsburgh, PA, from 2001 to 2004. He has been a professor with the Institute of Computing Technology, Chinese Academy of Sciences (CAS), since August 2004. He is the Director of the Key Laboratory of Intelligent Information Processing, CAS. He has published one book and over 200 papers in refereed journals and proceedings in the areas of computer vision, pattern recognition, image processing, and multimodal interfaces. He is a leading editor of the Journal of Computer Science and Technology, and an associate editor in chief of the Chinese Journal of Computers. He served as an Organizing Committee / Program Committee Member for more than 50 conferences. He is a recipient of several awards, including the China State Scientific and Technological Progress Award in 2000, 2003, 2005, and 2012 for his research work. He is a Fellow of IEEE and the China Computer Federation (CCF).

Matti Pietikäinen received his Doctor of Science in Technology degree from the University of Oulu, Finland. He is a professor at the Center for Machine Vision and Signal Analysis, University of Oulu. From 1980 to 1981 and from 1984 to 1985, he visited the Computer Vision Laboratory at the University of Maryland. He has made fundamental contributions, e.g. to local binary pattern (LBP) methodology, texture-based image and video analysis, and facial image analysis. He has authored about 350 refereed papers in international journals, books and conferences. His papers have about 53,500 citations in Google Scholar (h-index 78), and eight of these have over 1,350 citations. He was Associate Editor of IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Pattern Recognition, IEEE Transactions on Forensics and Security journals, and Image and Vision Computing journals. Currently he serves as Associate Editor of IEEE Transactions on Biometrics, Behavior and Identity Science, and Guest Editor for special issues of IEEE TPAMI and International Journal of Computer Vision. He was President of the Pattern Recognition Society of Finland from 1989 to 1992, and was named its Honorary Member in 2014. From 1989 to 2007 he served as Member of the Governing Board of International Association for Pattern Recognition (IAPR), and became one of the founding fellows of the IAPR in 1994. He is IEEE Fellow for contributions to texture and facial image analysis for machine vision. In 2014, his research on LBP-based face description was awarded the Koenderink Prize for Fundamental Contributions in Computer Vision. He was the recipient of the prestigious IAPR King-Sun Fu Prize 2018 for fundamental contributions to texture analysis and facial image analysis. He was named a Highly Cited Researcher by Clarivate Analytics in 2018.