Multiscale Graph Texture Network

Ravishankar Evani, Deepu Rajan, and Shangbo Mao

College of Computing and Data Science, Nanyang Technological University, Singapore
{S220007,ASDRajan,MAOS0003}@e.ntu.edu.sg

Abstract. Texture recognition has predominantly relied on methods based on handcrafted features and more recently, on Convolutional Neural Network (CNN)-based methods. However, many of these approaches do not capture the underlying directional relationships between visual vocabularies, attributes and features. In this study, we introduce a graph-based deep learning framework for texture and material recognition called Graph Texture Network (GTN) that models the underlying directional associations among latent texture attributes, that are hierarchically related to visual texture attributes, facilitating information exchange among them and consequently improving the discriminative capability among different texture and material categories. GTN, designed to handle non-Euclidean data structures, provides flexibility to learn complex underlying relationships among latent texture attributes via a learnable masked adjacency matrix. To ensure robustness of GTN to noise, especially on graphs with fewer vertices, we facilitate re-calibration of self-loop edge weights to preserve salient texture information within each vertex. We then utilize message passing mechanisms to enrich the representations of latent texture attributes. Furthermore, GTN is able to facilitate interaction across multiple graphs, representing texture information across a range of scales. Finally, GTN can be easily incorporated into a variety of CNN architectures for end-to-end training and does not require fine-tuning of pre-trained CNN backbones. Experimental results demonstrate that GTN achieves state-of-the-art performance on several benchmark texture and material datasets. Our code is available [1].

Keywords: Graph Neural Network · Texture Recognition

1 Introduction

All objects have surface properties like texture, color and shape. Unlike color or shape, which can be easily quantified and described, texture poses unique challenges due to its complexity and subjective interpretation. Our ability to pre-attentively perceive material properties e.g., roughness, is comparable to that of recognising objects [14]. However, compared to object recognition, material perception has received relatively less attention in computer vision literature. A good understanding of surface texture properties of objects enables Artificial General Intelligence (AGI) agents to better interact with their environment. For

[1] Code: https://github.com/RavishankarEvani/GTN

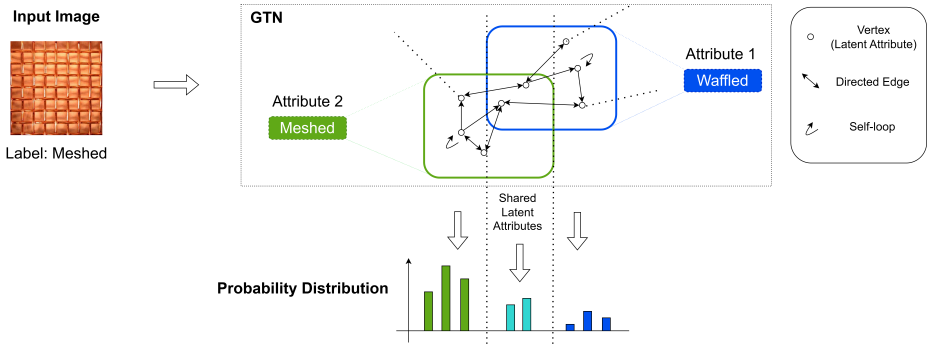


Fig. 1: Illustration of basic idea of GTN which encodes feature maps associated with a texture image from DTD [8] dataset as a graph, where vertices represent various latent texture attributes. Information exchange among these related latent texture attributes leads to the formation of a probability distribution over visual texture attributes.

instance, the action sequence required to clean up liquid spillage on a spongy sofa cushion would differ significantly from that on a wooden tabletop.

There are multiple definitions to texture due to its subjective interpretation and widespread application; however, texture can be viewed as a function of spatial variation in pixel intensities [42]. It can exhibit a variety of internal characteristics, ranging from high degree of stochasticity to predictable regularity. Influences from external factors (e.g., changes in illumination, scale and occlusion) can lead to appearance deformation, resulting in significant intraclass variability and potentially indistinguishable differences between classes.

The perceptual characteristics of visual texture have been examined in earlier studies, leading to the identification of a lexicon of terms used to describe texture [2,8]. Texture attributes are semantic concepts represented in visual textures and multiple texture attributes can be perceptually similar [51]. For instance a marbled texture can exhibit veined, stratified and cracked surface properties. Inspired by these phenomena, our study aims to uncover how latent attributes underlying these perceptually similar texture attributes are geometrically and directionally related, and how neighboring latent attributes can enrich their latent representations. We further model the hierarchical association of latent texture attributes with visual texture attributes. In this paper, we will examine the effectiveness of using graphs, which are non-Euclidean data structures, to model structural relationships among latent texture attributes.

We propose a graph-based deep learning framework called Graph Texture Network (GTN). First, we convert feature maps, extracted from several layers of a backbone pre-trained on ImageNet [37], into graph vertex representations of latent texture attributes, each associated with one or more perceptual texture attributes. Next, learn a masked adjacency matrix that describes the relationships among these latent texture attributes. To maintain GTN's performance even with a small number of vertices in the graph, we re-calibrate self-loop edge

weights to mitigate the influence of noise from neighboring vertices, which can corrupt latent texture representations during message passing and aggregation. Next, we perform residual message passing and aggregation of related information to enhance latent texture attributes. Finally, to account for the translationally invariant property of texture [28], we aggregate along the spatial dimension to obtain orderless representations of the enhanced latent texture attributes. The general idea of GTN is illustrated in Fig. 1, where each visual attribute (Meshed and Waffled) has a set of latent texture attributes. Meshed and waffled textures may share structural similarities, resulting in some vertices being represented in both visual attributes. This selective node sharing and directional neighborhood connectivity of latent texture information facilitate message passing, enriching the latent representations of visual attributes and enhancing the discrimination among closely related texture and material categories.

GTN was purposefully designed to exhibit robustness across textures represented at different scales. This versatility is essential, as texture images may contain varying sizes of textons, or images within a texture category may each be represented at a different scale. To tackle this variability, GTN constructs a graph from the feature maps extracted at each layer of the backbone. Each graph captures texture information at a specific scale present in the input image. By enabling interaction among these graphs, GTN enhances its ability to make scale-invariant predictions.

Our contributions can be summarised as follows:

1. We propose GTN, a novel graph-based deep learning framework for texture recognition. GTN does not require fine-tuning of the backbone as it has an encoder that can effectively translate feature maps to latent texture attributes. Their representations are then enriched through message passing. To our knowledge, this is the first time a graph-based deep learning approach is used for *multiscale* texture recognition.
2. We developed a hierarchical graph encoding scheme to generate multiple graphs containing texture information at various scales to enable multiscale interaction.
3. GTN is able to achieve state-of-the-art results on several challenging texture and material recognition datasets.

2 Related Work

Handcrafted Features & Statistical Methods. The field of texture classification and recognition has evolved over several decades, marked by key contributions and methodologies. Early work by Julesz [19] on textons and Haralick *et al.* [15] on co-occurrence matrices laid the foundation for statistical approaches. Laws [21,22] introduced texture energy measures using convolution masks, which were widely adopted for texture classification. Notable frameworks also emerged, including Bag of Textons (BoT) [11, 24] and Bag of Words (BoW) [10] methods, where images are represented as orderless histograms over a dictionary of textons. These frameworks involve encoding local patches, aggregating global

features, and using algorithms such as Support Vector Machines (SVM) to classify an image into one of the texture categories. Varma *et al.* [43, 44] advanced the field by emphasizing the learning of texture descriptors from training data. Additionally, powerful residual encoders have been developed; Jégou *et al.* [20] introduced VLAD, which aggregates first-order statistics by accumulating differences of local features assigned to each codeword, while Perronnin *et al.* [35] used Fisher Vectors (FV, a probabilistic variant of VLAD) to encode local image descriptors, capturing first and second-order statistics with respect to the visual vocabulary.

Automatic Feature Learning. Recent literature focused more on leveraging pretrained CNN models. FV-CNN [9] and LFV [41] use CNNs to extract features that are then encoded using FV. However, methods that adopt FV, do not support end-to-end training. Zhang *et al.* [52] addressed this issue by proposing Deep-TEN, a framework that integrates dictionary learning and residual encoding within an encoding layer. This encoding layer which acts as a pooling layer (equivalent to VLAD and FV) is trainable end-to-end alongside a CNN backbone. While RPNet [33] simplified the Deep-TEN architecture by replacing dictionary learning component with a residual pooling layer, DEP [46] improved on Deep-TEN by combining orderless texture information with local spatial features in order to balance both the orderless component and ordered spatial information. Hu *et al.* [17] proposed MuLTER which improves over DEP by simultaneously leveraging features from various layers allowing it to maintain both texture details and spatial information. Song *et al.* [40] introduced MSBFEN which leverages a multiscale feature encoding mechanism combined with boosting to effectively capture discriminative texture features. Some studies [5,6] have explored cross-layer statistical features for texture recognition. Chen *at el.* [5] introduced the concept of cross-layer statistical self-similarity to texture recognition, which captures the consistency of statistical properties from feature maps extracted from different layers of a CNN backbone. In a later contribution, Chen *at el.* [6] explored the use of cross-layer features in CNNs for local feature extraction. They developed a local encoding scheme that is inspired by LBP, for feature extraction. Their method, DTPNet, facilitates rotation invariance and gray-scale robustness to extracted local codes.

Yang *et al.* [48] proposed DFAEN, which applies frequency attention mechanism to extract salient features from the input image. In addition, DFAEN leverages on both first and second order texture encoding information to capture more texture details. Scabini *et al.* [38] utilizes a randomized aggregation mechanism to fuse deep activation maps at different layers of a convolutional neural network (CNN). By incorporating randomness into the aggregation process, RADAM can effectively capture diverse texture patterns while reducing overfitting. Similar to RADAM, we will evaluate GTN's effectiveness at modeling textures without fine-tuning pre-trained ConvNeXt backbones.

Peeples *et al.* [34] proposed HistNet, a localized histogram layer that maintains spatial context. This layer allows bin centers and widths to be learned during training. By incorporating Radial basis functions, HistNet ensures it is

robust against small intra-class variations. Some studies [45,51] utilised spatially aware aggregation operations; Yong *et.al.* [45] proposed FENet, which leverages multi-fractal geometry and uses a hierarchical fractal analysis process for encoding the regularity of spatial arrangement in a feature map. Zhai *et al.* [49] replaced spatially invariant orderless aggregation operation with DSR-Net that captures spatial dependency of local primitive characteristics as inherent structural representations for texture recognition. In their earlier contribution that is more relevant to our study, Zhai *et al.* [51] proposed MAP-net that progressively learns visual texture attributes in a mutually reinforced manner. A spatially-adaptive Global Average Pooling (GAP) is then applied to each branch to aggregate the features. In their more recent work, Zhai *et al.* [50] introduced MPAP, which adopts a multi-branch architecture to model both bottom-up structure and top-down attributes relations by progressively learning the association between local structure and global attributes. While relevant literature [50, 51] studied correlations between visual texture attributes and associations between local structure and global attributes, directional information of visual and latent texture attributes were not considered, e.g., waffled visual texture attribute might be strongly dependent on checkered characteristics but the inverse relationship may be weaker. Reale *et al.* [36] developed a technique to represent triangular patches within an image as nodes in a graph, aiming to encode texture information. However, their approach did not consider directional characteristics among texture attributes and the multiscale nature of texture.

In our study we explore the effectiveness of a graph-based framework that aims to capture the relationships among latent characteristics of texture that are indicative of perceptual texture properties. In addition, we will investigate how GTN facilitates directional information flow among latent texture attributes enriching their representations. Furthermore, we will evaluate GTN's ability to accommodate texture information at various scales.

3 Graph Texture Network

The overall architecture of the Graph Texture Network (GTN) is shown in Fig. 2 (a). GTN first transforms feature maps, $\mathcal{F} = \{\mathbf{X}^{o_m} \mid 0 \leq m \leq M-1\}$, derived from M layers of a deep neural network backbone into a set of unsigned homogeneous directed graphs, $G = \{G^m \mid 0 \leq m \leq M-1\}$. A graph generated from feature maps from the m^{th} layer is defined as $G^m = (\mathbf{X}^{g_m}, \mathcal{E}^m)$ where $\mathbf{X}^{g_m} = \{\mathbf{X}_i^{g_m} \mid 0 \leq i \leq N_m - 1\}$ is a set of $N_m = |\mathbf{X}^{g_m}|$ vertices with $\mathcal{E}^m$ edges. The vertices represent latent texture attributes while the unsigned masked adjacency matrix, $\tilde{\mathbf{A}}^m \in \mathbb{R}^{N_m \times N_m}$ represents the relationship between these attributes.

3.1 Vertex Representation of Latent Texture Attributes

The first step is to encode feature maps $\mathbf{X}^{o_m} \in \mathbb{R}^{H_m \times W_m \times C_m}$ as nodes in a graph as depicted in Fig. 2 (b), where C_m are the number of channels and H_m and W_m

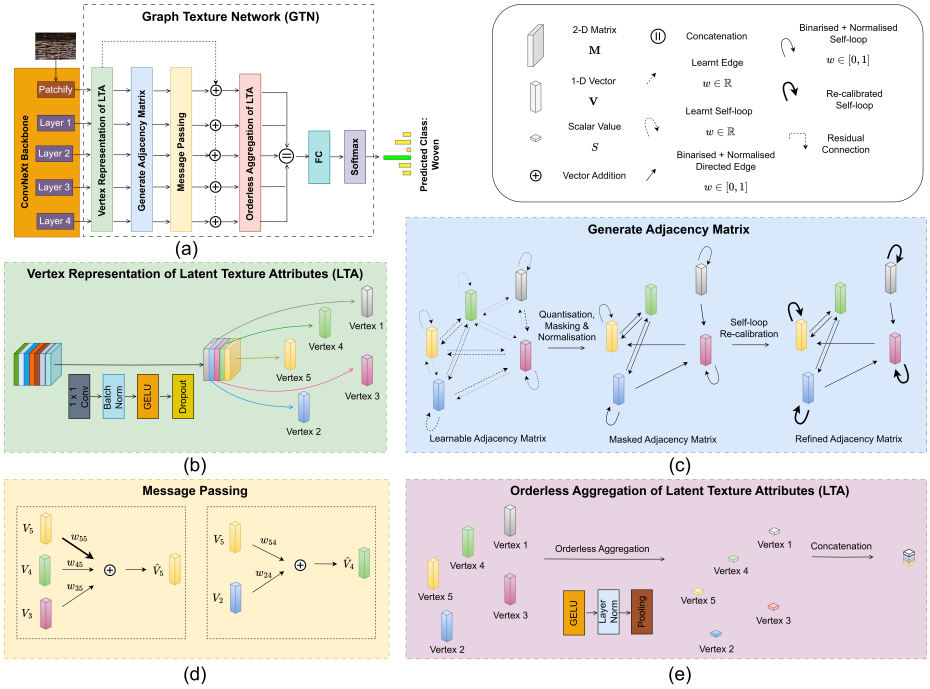


Fig. 2: GTN Architecture and associated components. (a) Architecture of GTN applied to ConvNeXt backbone, (b) conversion of feature maps from backbone to graph vertices representing latent texture attributes, (c) generation of adjacency matrix relating the latent texture attributes, (d) how messages are aggregated and vertices are updated, (e) orderless aggregation of latent texture attributes and subsequent concatenation.

are height and width of the feature maps from the m^{th} layer of the backbone. A 1×1 convolution layer is used to compress the representation of features to obtain $\mathbf{X}^{c_m} \in \mathbb{R}^{H_m \times W_m \times \lfloor \frac{C_m}{r} \rfloor}$, where r is the compression ratio. Batch Normalisation [18] and GELU [16] activations are then applied to $\mathbf{X}^{c_m}$ to reduce internal co-variate shift and to ensure non-saturating activations respectively. Finally, dropout is applied to regularise the training process and diversify the feature space. $\mathbf{X}^{c_m}$ is then reshaped to obtain graph vertices $\mathbf{X}^{g_m} \in \mathbb{R}^{F_m \times \lfloor \frac{C_m}{r} \rfloor}$, where $F_m = H_m W_m$ represents the flattened spatial dimension of the feature maps and r is set to 4 so that the number of vertices from all M graphs generated from the GTN module (stemming from each of the M layers of the ConvNeXt-T [29] backbone) are approximately or at least 367 (i.e. $N \gtrapprox 367$), the number of words associated with texture lexicon [2].

3.2 Masked Adjacency Matrix

Relationship Matrix. We need to understand how the vertices in the graph representing latent texture attributes are related. As depicted in Fig. 2 (c), first,

a relationship matrix $\boldsymbol{\xi}^m \in \mathbb{R}^{N_m \times N_m}$ is learnt through backpropogation. The m^{th} relationship matrix corresponds to the the m^{th} graph (associated with the m^{th} layer of the backbone). Each element in the relationship matrix, (ξ_{ij}^m) represents the strength of the relationship between latent texture attribute i and its neighbor j. The relationship matrix can serve as an alternative to bilinear models [12, 26, 27] for representing second-order information.

Binary Quantization. Deep learning models often exhibit sensitivity to noise, particularly when trained on small texture and material datasets. To address this challenge, we apply binary quantization to the relationship matrix. This ensures robustness to small variations within the data, enhancing the model's ability to generalize effectively. As part of the binary quantization process, we first apply a sigmoid activation function to scale values in ξ^m between the range 0 and 1. An indicator function $\mathbb{1}(\cdot)$ is then applied to mask edges with weights that are less than or equal to the threshold, λ which is set as 0.5, generating a masked adjacency matrix expressed as

$$\Gamma_{ij}^m = \mathbb{1}\left[\frac{1}{1+e^{-\xi_{ij}^m}} > \lambda\right]. \tag{1}$$

The activated edges imply a positive relationship between corresponding latent texture attributes. We then perform normalisation of the masked adjacency matrix by scaling it the with the neighborhood size using the in-degree matrix, $\hat{D}_i^m = \sum_j \xi_{ij}^m$, where the in-degree of a node i is the sum of the incoming edges to that node. These elements represent edge weights between pairs of vertices. The normalised masked adjacency matrix is formulated as

$$\tilde{\mathbf{A}}^m = \hat{\mathbf{D}}^{m^{-1}} \boldsymbol{\Gamma}^m, \tag{2}$$

where $\boldsymbol{\Gamma}^m \in \{1,0\}^{N_m \times N_m}$. $\tilde{\mathbf{A}}^m \in \mathbb{R}^{N_m \times N_m}$ may exhibit directionality ($\tilde{a}_{j,i}^m$ may not be the same as $\tilde{a}_{i,j}^m$), and self-loops for specific latent texture attributes could be learnt to be important through backpropagation. The generated masked adjacency matrix is non-negative, i.e., $\tilde{a}_{j,i}^m \geq 0,\ \forall i, j$, implying an unsigned graph.

Self-loop Re-calibration. Inspired by Cluster-GCN [7] that uses "diagonal enhancement" to improve convergence and accuracy, we introduce a learnable variant which adapts to the masking of the adjacency matrix called, self-loop re-calibration to GTN to ensure it continues to perform well even when compression ratios, r are high. When r is high, each vertex will have fewer neighbors, and therefore, noise from neighboring vertices can have more influence on latent texture representations during message passing and aggregation. To avoid this phenomenon, the weights for self-loops that are preserved after masking the adjacency matrix are re-calibrated to emphasize the importance of associated latent texture attributes, preventing neighboring latent texture attributes from dominating during message aggregation. Re-calibration of self-loops helps regularize the training process by proportionately reducing the contribution of neighboring vertices when aggregating neighborhood messages. Self-loop re-calibration

is done by learning an inflation function, $g(\psi_m)$ parameterized by a learnable parameter, ψ_m to control the inflation of weights of self-loops as

$$\text{diag}(\tilde{\mathbf{A}}^m) \leftarrow \text{diag}(\tilde{\mathbf{A}}^m) \times \underbrace{\left[\frac{2+e^{-\psi_m}}{1+e^{-\psi_m}}\right]}_{g(\psi_m)}, \qquad \begin{array}{l} \forall\ \text{diag}(\tilde{\mathbf{A}}^m) \neq 0, \\ \psi_m \in (-\infty, \infty), \\ g(\psi_m) \in (1, 2). \end{array} \tag{3}$$

As the number of graph vertices differ between layers, ψ_m is learnt separately for every layer.

3.3 Residual Message Passing

Next, we enhance the latent texture attributes via message passing and aggregation of neighboring latent texture attributes as illustrated in Fig. 2 (d). Each element in the masked adjacency matrix, $\tilde{A}^m$ represents an edge weight, $\tilde{a}^m_{j,i}$ between i^{th} texture attribute, $\mathbf{X}^{g_m}_i$ and it's neighbors, $\mathbf{X}^{g_m}_j$ (j could also represent self-loops if they are preserved after masking as described in Sec. 3.2). We scale the weighted aggregation of neighboring latent texture attributes and add them to $\mathbf{X}^{g_m}_i$ as follows

$$\hat{\mathbf{X}}^{g_m}_i = \mathbf{X}^{g_m}_i + \rho \textstyle\sum_{j \in \mathscr{N}(i) \bigcup \{i\}} \tilde{a}^m_{j,i} \mathbf{X}^{g_m}_j, \qquad \forall\ i \in \{0, ..., N-1\} \tag{4}$$

where $\mathscr{N}(i) = \{j \mid (j \rightarrow i) \in \mathcal{E}^m\}$ represents a set containing the in-degree neighbors of the i^{th} texture attribute and $\{i\}$ represents the i^{th} texture attribute, if self-loop is learnt. The scaling factor, ρ can be used to proportionately adjust the collective influence of re-calibrated and neighboring vertices. The process of message passing and aggregation enhances latent texture attributes by incorporating information from related latent attributes. Finally, we apply GELU to activate the enhanced attribute values. Next, we perform Layer Normalisation [1] of each layer's graph representation to mitigate internal covariate shift to stabilize the training process and accelerate convergence. The activation of vertex values and graph normalisation is formulated as

$$\mathbf{Z}^m = \gamma^{g_m} \left(\frac{f_a(\hat{\mathbf{X}}^{g_m}) - E[f_a(\hat{\mathbf{X}}^{g_m})]}{\sqrt{Var[f_a(\hat{\mathbf{X}}^{g_m})] + \epsilon}} \right) + \beta^{g_m}, \tag{5}$$

where f_a represents GELU activation function. $E[.]$ and $Var[.]$ represent expectation and variance respectively. γ^{g_m} and β^{g_m} are affine parameters learnt for m^{th} layer's graph representation. ϵ of $1e-5$ is added for numerical stability.

3.4 Multi-Graph Cross-Layer Latent Attribute Interaction

Next, we allow interaction of latent attributes from multiple graphs representing various scales of texture information. However, the enhanced latent texture attributes, $\mathbf{Z}^m$, are correlated with spatial coordinates of feature maps extracted

from the backbone, making them sensitive to intraclass variability in visual textures. To address this issue, we transform $\mathbf{Z}^m$ into a lower-dimensional, orderless representation that is invariant to spatial deformations. We achieve this by using the first statistical moments (mean) to obtain N_m orderless representations for m^{th} layer's graph. These representations are then concatenated across M layers to obtain a multiscale representation of texture information, as follows

$$\mathbf{Y} = \Big\Vert_{m=0}^{M-1} \left[\Big\Vert_{i=0}^{N_m-1} \left[\frac{1}{L_m} \sum_{k=0}^{L_m-1} z_{i_k}^m \right] \right], \tag{6}$$

where $\Vert$ is a concatenation operation and L_m is the length of m^{th} layer's embeddings; each element in the embedding has a positional index, k. The resulting vector, $\mathbf{Y}$, represents orderless texture information across different scales. To facilitate interaction among texture information across a range of scales and to model the hierarchical relationship between latent and visual texture attributes, we pass $\mathbf{Y}$ through a fully connected (FC) layer. This produces a vector of logits, which is then passed through a Softmax function to generate a probability distribution over the classes. The class with the highest probability is the predicted class.

4 Experiments

4.1 Datasets

We evaluated GTN using five benchmark datasets, with their characteristics summarized as follows: (a) Ground Terrain in Outdoor Scenes (GTOS) [47] is a dataset of 34,105 outdoor ground material images, separated into 40 categories, with 5 train/test splits. (b) GTOS-Mobile (GTOS-M) [46] is a dataset collected from GTOS via mobile phone, consisting of 100,011 material samples from 31 categories, with a single train/test split. (c) KTH-TIPS2-b (KTH) [4] is a dataset composed of 4,752 images from 11 material categories, with 4 train/test splits. (d) Flickr Material Dataset (FMD) [39] is composed of 10 material categories, with 100 images in each category. Validation is done through 10-fold cross-validation. Random sampling for train-validation is done with stratification to ensure the percentage of samples for each class is preserved. (e) Describable Texture Dataset (DTD) [8] contains 5,640 images belonging to 47 categories, with 10 train/validation/test splits.

4.2 Implementation Details

Our model is implemented in PyTorch and runs on a single NVIDIA RTX A6000 GPU. For all datasets, the learning rate is initialized to 0.0001 and follows a cosine annealing schedule [30]. We use the Adam optimizer with decoupled weight decay [31] and focal loss [25]. For each dataset, we train GTN until the final epoch and report the model's performance on the test set. Detailed training

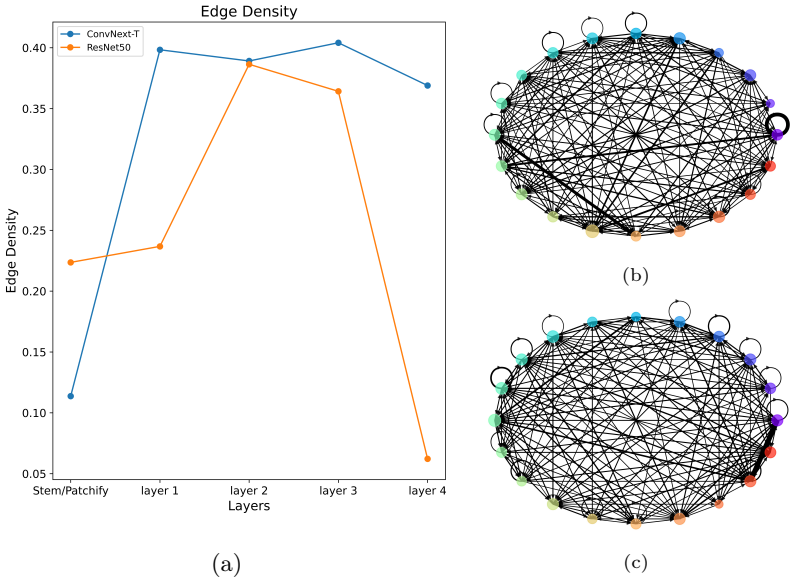


Fig. 3: (a) Comparing edge density extracted at each layer by pre-trained backbones (ConvNeXt-T and ResNet50) using the KTH dataset. (b) Layer 3 and (c) Layer 4 Subgraph using ConvNeXt-L backbone trained on DTD Dataset.

hyperparameters and data augmentation techniques used in the experiments are provided in the supplementary material.

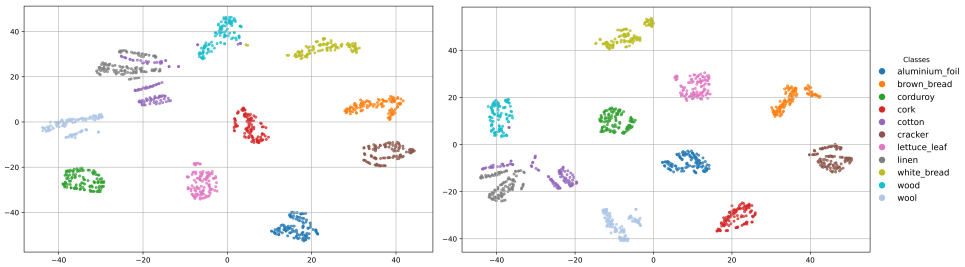
4.3 Backbone Selection & Comparison Against State of the Art

Edge information in images can be used to compute local measures of textural variation [13]. Fig. 3 (a) shows the variation in average edge density, determined by the Canny edge detection algorithm [3], across network depths in the filter responses of ConvNeXt-T and ResNet50 backbones. This is based on 500 randomly selected images from KTH, where edge density is measured as the proportion of pixels identified as edges in a feature map. The results show that ConvNeXt-T maintains a relatively high and stable edge density from layers 1 to 4, indicating that texture information remains consistently accessible throughout these network layers. This characteristic enhances the network's effectiveness in multi-scale texture recognition. In contrast, ResNet50 exhibits a trend where high-level semantic features are prioritized over fine-grained edge details, especially in the deepest layers. This makes ResNet50 more suitable for tasks such as object recognition, where high-level contextual information is more critical than edge details. Recent literature [38] has also quantitatively shown that ConvNeXt backbones are better at extracting texture-related features compared to ResNet backbones. Hence, our paper will focus more of our analysis using a range of ConvNeXt backbone variants. For completeness, we will also evaluate GTN us-

ing ResNet50 backbone.

Tab. 1 depicts how GTN's performance compares with the current state-of-the-art RADAM across various ConvNeXT backbones pre-trained on ImageNet. Following RADAM [38], we have frozen our ConvNeXt backbones and the average and standard deviation measures are calculated across the provided splits, where applicable. We also analyze classification results for GTN using ResNet50 backbone, pre-trained on ImageNet. The classification accuracies are shown in Tab. 2. Following [48], we have fine-tuned the pre-trained backbone during the training of GTN and following [5, 45] the results on DTD and KTH are based on 5-time statistics, and the results on GTOS-M are averaged over 2 runs.

Analysis. GTN performed relatively well on most benchmark datasets. For DTD, KTH, and GTOS (using the ConvNeXt backbone), the performance gap between GTN and RADAM generally increases with the size of the backbone. This may be because a larger backbone would result in more latent texture attributes describing any visual attribute at each scale, enhancing GTN's capacity to represent texture information. Conversely, GTN underperformed on FMD (using the ConvNeXt backbone) and DTD (using the ResNet50 backbone). This may be due to GTN creating graph vertices at an early stage of the architecture, removing high-level shape information while preserving low to mid-level information like texture. FMD and DTD contain relatively more object level information compared to the remaining datasets. For instance, in DTD dataset, several images in the freckles texture category contain objects that have rounded shapes. Similarly, object shapes are visible in FMD dataset. The t-SNE [32] plots in Fig. 4 were generated using embeddings extracted before they are fed into the final fully connected (FC) layer of GTN and the SVM classifier of RADAM. These embeddings were obtained using KTH's fourth test split and the ConvNeXt-L backbone. The clusters generated by GTN appear to be more homogeneous, which may be attributed to GTN's ability to create diverse latent attributes and subsequently enhance them through message passing.



(a) (b)

Fig. 4: Visualisation of embeddings from (a) RADAM and (b) GTN using t-SNE [32]. Backbone: ConvNeXt-L. Dataset: KTH (Split 4 Test Set)

Table 1: Comparison of GTN with RADAM [38] in terms of classification accuracy (%) on texture benchmarks. ConvNeXt-* variants are used as backbones. Pre-trained weights are frozen. The best result for each backbone-dataset combination is **bolded**

Model	-*	DTD	FMD	KTH	GTOS	GTOS-M
RADAM	N	74.9 ±0.7	**87.1** ±0.4	89.6 ±3.8	**83.7** ±1.5	81.8
GTN (ours)	N	**75.4** ±1.1	86.9 ±3.1	**90.3** ±4.7	82.9 ±1.7	**90.1**
RADAM	T	77.0 ±0.7	**88.7** ±0.4	90.7 ±4.0	84.2 ±1.7	85.3
GTN (ours)	T	**77.5** ±1.0	87.2 ±3.4	**92.0** ±4.9	**84.7** ±1.6	**91.9**
RADAM	B	76.4 ±0.9	**90.2** ±0.2	87.7 ±5.6	84.1 ±1.6	82.2
GTN (ours)	B	**78.3** ±0.8	88.8 ±3.0	**90.8** ±6.2	**84.2** ±1.8	**94.8**
RADAM	L	77.4 ±1.1	**89.3** ±0.3	89.3 ±3.4	84.0 ±1.8	85.8
GTN (ours)	L	**79.3** ±0.8	88.1 ±3.0	**92.5** ±4.7	**84.9** ±1.7	**93.9**

Table 2: Comparison of GTN with different methods in terms of classification accuracy (%) on texture benchmarks. ResNet50 is used as backbone.

Model	DTD	KTH	GTOS-M
Deep-TEN [52]	69.6	82.0 ±3.3	-
MAP-net [51]	76.1 ±0.6	84.5 ±1.3	86.6 ±1.5
DSR-Net [49]	77.6 ±0.6	85.9 ±1.3	87.0 ±1.5
CLASSNet [5]	74.0 ±0.5	87.7 ±1.3	85.7 ±1.4
FENet [45]	74.2 ±0.1	88.2 ±0.2	85.2 ±0.4
RPNet [6, 33]	73.0 ±0.6	87.2 ±1.8	77.9 ±0.3
MSBFEN [40]	77.8 ±0.5	86.2 ±1.1	87.6 ±1.6
MSTH-Net [23]	71.45 ±0.6	87.7 ±1.0	87.5 ±0.8
DFAEN [48]	73.2	86.3	86.9
MPAP [50]	**78.0** ±0.5	89.0 ±1.0	88.1 ±1.3
RADAM [38]	75.6 ±1.1	88.5 ±3.2	81.0
DTPNet [6]	73.5 ±0.4	88.5 ±1.6	88.0 ±1.2
GTN (ours)	74.6 ±0.2	**89.3** ±0.3	**91.6** ±1.4

Table 3: Evaluate effectiveness of VR and RMP components of GTN in terms of classification accuracy (%). ConvNeXt-T is used as the backbone.

VR	RMP	DTD	KTH
✗	✗	27.6 ±4.3	46.5 ±5.0
✓	✗	77.0 ±0.8	91.2 ±4.6
✓	✓	**77.5** ±1.0	**92.0** ±4.9

Table 4: Evaluate effectiveness of SR in terms of classification accuracy (%). Compression ratios, $r = [8, 16]$. ConvNeXt-T is used as the backbone.

r	SR	DTD	KTH
8	✗	76.6 ±0.9	**92.3** ±5.2
8	✓	**76.7** ±0.9	**92.3** ±5.2
16	✗	71.4 ±0.5	92.1 ±5.2
16	✓	**71.5** ±0.6	**92.2** ±5.2

4.4 Subgraph Visualization

Fig. 3 shows randomly sampled subgraphs of 20 vertices with associated edges generated from feature maps extracted from layers 3 and 4 of the ConvNeXt-L backbone, which was trained on the DTD dataset. The thickness of the edges is proportional to the edge weights, determined by the neighborhood size and the re-calibration of self-loops. We can observe that directionality is important; some

latent texture attributes depend on others, while the inverse relationship does not exist. Additionally, some vertices do not have self-loops, indicating that they rely solely on neighboring attribute representations. In contrast, vertices with self-loops find it important to retain a proportion of their own latent attribute information when aggregating attribute information from their neighbors.

4.5 Ablation Study

We perform ablation studies to understand the effectiveness of vertex representation of latent texture attributes (VR), residual message passing and aggregations (RMP) and self-loop re-calibration (SR) components of GTN on a texture dataset, DTD and a material dataset, KTH. Quantitative evaluation of the effectiveness of VR and RMP components are depicted in Tab. 3. First, we evaluate the effectiveness of the RMP component by replacing it with a linear layer; parameters of which are learnt through backpropagation. The 5 projection matrices from the 5 layers are concatenated and passed through a fully connected (FC) layer and a Softmax function to determine the predicted class. The RMP component has demonstrated to be quite useful when applied to both texture, DTD and material, KTH datasets, suggesting that the residual message passing and aggregation framework effectively represents latent texture attributes.

Next, to evaluate the effectiveness of GTN module as a whole we replace both VR and RMP components with a linear layer; parameters of which are learnt through backpropagation. We reshape the feature maps from the backbone by flattening the spatial dimension, $\mathbb{R}^{H_m W_m \times C_m}$ before feeding them into the linear layer. The 5 projection matrices from the 5 layers are concatenated and passed through a fully connected (FC) layer and a Softmax function to determine the predicted class. The linear layers fail to learn a meaningful representation of the input texture and material information and exhibit underfitting when the backbone is frozen. This underscores the representational capacity of the VR component, which can effectively learn latent texture attributes without requiring fine-tuning of any variant of the ConvNeXt backbone.

Finally, we evaluate effectiveness of SR. Tab. 4 demonstrates a slight improvement in both texture and material datasets when compression ratio, $r = 16$. When $r = 8$, only DTD has improved slightly while GTN's performance on KTH remained the same. This might be because the benchmark datasets are well-prepared and contain minimal noise. Any small amount of noise present has likely been corrected and regularized by GTN through SR, which serves to reduce the influence of neighboring nodes.

4.6 Effect of Multi-layer Representation and Compression Ratio

We conducted experiments to evaluate the effectiveness of injecting various scales of texture information and the number of latent texture attributes in each graph. This was done by incrementally adding layers (from deepest to shallowest) that feed into the GTN and by adjusting the compression ratio, r. Results are depicted in Tab. 5. We achieved the best performance on DTD and GTOS-M when using

Table 5: Performance of GTN in terms of classification accuracy (%) with varying number of layers (M). ConvNeXt-T is used as the backbone, r=4.

M	DTD	KTH	GTOS-M
1	75.0 ±1.1	90.5 ±5.2	89.9
2	77.4 ±1.0	91.9 ±5.1	90.9
3	**77.5** ±1.0	**92.2** ±5.1	91.0
4	**77.5** ±0.9	**92.2** ±5.0	91.4
5	**77.5** ±1.0	92.0 ±4.9	**91.9**

Table 6: Performance of GTN in terms of classification accuracy (%) with different compression ratios, r. ConvNeXt-T is used as the backbone, $M = 5$.

r	$\sum_{m=0}^{M-1} N_m$	DTD	KTH	GTOS-M
2	768	**77.9** ±0.8	91.8 ±5.6	90.8
4	384	77.5 ±1.0	92.0 ±4.9	**91.9**
8	192	76.7 ±0.9	**92.3** ±5.2	91.2
16	96	71.5 ±0.6	92.2 ±5.2	90.4

feature maps from all layers, while we noticed a slight drop in performance on KTH dataset. In general, all layers contain useful texture information at various scales that can be encoded as latent texture attributes within GTN.

We also tested GTN by altering the compression ratio, r, which, as discussed in Sec. 3.1, is used to encode texture features as graph vertices. A larger r results in a smaller number of graph vertices, $\sum_{m=0}^{M-1} N_m$, corresponding to the number of latent texture attributes. The results of the experiment are shown in Tab. 6. GTN performs best on DTD when $r = 2$, and on GTOS-M when $r = 4$, both resulting in latent texture attributes exceeding 367, the number of words associated with the lexicon for visual textures defined by Bhushan *et al.* [2]. KTH, having only 11 material categories, requires fewer latent attributes to model them. Conversely, DTD and GTOS-M, with 47 and 31 texture and material categories respectively, require more latent attributes. We observe a general drop in performance with higher r values, which can be attributed to the resulting smaller number of graph vertices, leading to a sub-optimal hierarchical mapping from latent to visual texture attributes.

5 Conclusion

This paper introduces GTN, a novel approach that represents each texture image as a graph comprising latent texture attributes. The GTN framework facilitates residual message passing to further enrich texture representations. GTN learns a hierarchical mapping between these latent attributes and visual texture characteristics. By utilizing graphs, a non-Euclidean data structure, GTN enables flexible connectivity between vertices, allowing for the representation of intricate directional relationships among neighboring latent texture attributes. Moreover, GTN's architecture includes an encoder that effectively translates feature maps into latent texture attributes, obviating the need for fine-tuning the backbone.

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization (2016), https://arxiv.org/abs/1607.06450
2. Bhushan, N., Rao, A.R., Lohse, G.L.: The texture lexicon: Understanding the categorization of visual texture terms and their relationship to texture images. Cognitive Science **21**(2), 219–246 (1997). https://doi.org/https://doi.org/10.1016/S0364-0213(99)80023-8, https://www.sciencedirect.com/science/article/pii/S0364021399800238
3. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-8**(6), 679–698 (1986). https://doi.org/10.1109/TPAMI.1986.4767851
4. Caputo, B., Hayman, E., Mallikarjuna, P.: Class-specific material categorisation. In: Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1. vol. 2, pp. 1597–1604 Vol. 2 (2005). https://doi.org/10.1109/ICCV.2005.54
5. Chen, Z., Li, F., Quan, Y., Xu, Y., Ji, H.: Deep texture recognition via exploiting cross-layer statistical self-similarity. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5227–5236 (2021). https://doi.org/10.1109/CVPR46437.2021.00519
6. Chen, Z., Quan, Y., Xu, R., Jin, L., Xu, Y.: Enhancing texture representation with deep tracing pattern encoding. Pattern Recognition **146**, 109959 (2024). https://doi.org/https://doi.org/10.1016/j.patcog.2023.109959, https://www.sciencedirect.com/science/article/pii/S003132032300657X
7. Chiang, W.L., Liu, X., Si, S., Li, Y., Bengio, S., Hsieh, C.J.: Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. p. 257–266. KDD '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3292500.3330925, https://doi.org/10.1145/3292500.3330925
8. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3606–3613 (2014). https://doi.org/10.1109/CVPR.2014.461
9. Cimpoi, M., Maji, S., Vedaldi, A.: Deep filter banks for texture recognition and segmentation. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3828–3836 (2015). https://doi.org/10.1109/CVPR.2015.7299007
10. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: In Workshop on Statistical Learning in Computer Vision, ECCV. pp. 1–22 (2004)
11. Cula, O., Dana, K.: Compact representation of bidirectional texture functions. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. vol. 1, pp. I–I (2001). https://doi.org/10.1109/CVPR.2001.990645
12. Dai, X., Ng, J.Y.H., Davis, L.S.: Fason: First and second order information fusion network for texture recognition. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6100–6108 (2017). https://doi.org/10.1109/CVPR.2017.646
13. Davis, L.S., Mitiche, A.: Edge detection in textures. In: ROSENFELD, A. (ed.) Image Modeling, pp. 95–109. Academic Press (1981). https://doi.org/https://doi.org/10.1016/B978-0-12-597320-5.50010-3, https://www.sciencedirect.com/science/article/pii/B9780125973205500103

14. Fleming, R.W.: Material perception. Annual review of vision science **3**, 365–388 (2017). https://doi.org/10.1146/annurev-vision-102016-061429
15. Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural features for image classification. IEEE Transactions on Systems, Man, and Cybernetics **SMC-3**(6), 610–621 (1973). https://doi.org/10.1109/TSMC.1973.4309314
16. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
17. Hu, Y., Long, Z., AlRegib, G.: Multi-level texture encoding and representation (multer) based on deep neural networks. In: 2019 IEEE International Conference on Image Processing (ICIP). pp. 4410–4414 (2019). https://doi.org/10.1109/ICIP.2019.8803640
18. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift (2015), https://arxiv.org/abs/1502.03167
19. Julesz, B.: Visual pattern discrimination. IRE transactions on Information Theory **8**(2), 84–92 (1962)
20. Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 3304–3311 (2010). https://doi.org/10.1109/CVPR.2010.5540039
21. Laws, K.I.: Texture energy measures. In: Proceedings of Image Understanding Workshop (1979)
22. Laws, K.I.: Rapid Texture Identification. In: Wiener, T.F. (ed.) Image Processing for Missile Guidance. vol. 0238, pp. 376 – 381. International Society for Optics and Photonics, SPIE (1980). https://doi.org/10.1117/12.959169
23. Lee, D., Kim, H.C., Seo, J., Lee, S.: Materialnet: Multi-scale texture hierarchy and multi-view surface reflectance for material type recognition. In: 33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022. BMVA Press (2022), https://bmvc2022.mpi-inf.mpg.de/0361.pdf
24. Leung, T., Malik, J.: Representing and recognizing the visual appearance of materials using three-dimensional textons. International journal of computer vision **43**, 29–44 (2001). https://doi.org/10.1023/A:1011126920638
25. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(2), 318–327 (2020). https://doi.org/10.1109/TPAMI.2018.2858826
26. Lin, T.Y., Maji, S.: Visualizing and understanding deep texture representations. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2791–2799 (2016). https://doi.org/10.1109/CVPR.2016.305
27. Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear cnn models for fine-grained visual recognition. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 1449–1457 (2015). https://doi.org/10.1109/ICCV.2015.170
28. Liu, L., Chen, J., Fieguth, P., Zhao, G., Chellappa, R., Pietikäinen, M.: From bow to cnn: Two decades of texture representation for texture classification. International Journal of Computer Vision **127**, 74–109 (2019)
29. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11966–11976 (2022). https://doi.org/10.1109/CVPR52688.2022.01167
30. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts (2017), https://arxiv.org/abs/1608.03983
31. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization (2019), https://arxiv.org/abs/1711.05101

32. van der Maaten, L.: Accelerating t-sne using tree-based algorithms. Journal of Machine Learning Research **15**(93), 3221–3245 (2014), http://jmlr.org/papers/v15/vandermaaten14a.html
33. Mao, S., Rajan, D., Chia, L.T.: Deep residual pooling network for texture recognition. Pattern Recognition **112**, 107817 (2021). https://doi.org/https://doi.org/10.1016/j.patcog.2021.107817, https://www.sciencedirect.com/science/article/pii/S0031320321000042
34. Peeples, J., Xu, W., Zare, A.: Histogram layers for texture analysis. IEEE Transactions on Artificial Intelligence **3**(4), 541–552 (2022). https://doi.org/10.1109/TAI.2021.3135804
35. Perronnin, F., Dance, C.: Fisher kernels on visual vocabularies for image categorization. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8 (2007). https://doi.org/10.1109/CVPR.2007.383266
36. Reale, M.J., Ochrym, M., Church, M., Goutermout, N., Rubado, J., Cornacchia, M.: Shape and texture aware graph processing. In: 2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR). pp. 1–18 (2020). https://doi.org/10.1109/AIPR50011.2020.9425296
37. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
38. Scabini, L., Zielinski, K.M., Ribas, L.C., Gonçalves, W.N., De Baets, B., Bruno, O.M.: Radam: Texture recognition through randomized aggregated encoding of deep activation maps. Pattern Recognition **143**, 109802 (2023). https://doi.org/https://doi.org/10.1016/j.patcog.2023.109802, https://www.sciencedirect.com/science/article/pii/S0031320323005009
39. Sharan, L., Rosenholtz, R., Adelson, E.: Material perception: What can you see in a brief glance? Journal of Vision **9**(8), 784–784 (2009)
40. Song, K., Yang, H., Yin, Z.: Multi-scale boosting feature encoding network for texture recognition. IEEE Transactions on Circuits and Systems for Video Technology **31**(11), 4269–4282 (2021). https://doi.org/10.1109/TCSVT.2021.3051003
41. Song, Y., Zhang, F., Li, Q., Huang, H., O'Donnell, L.J., Cai, W.: Locally-transferred fisher vectors for texture classification. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 4922–4930 (2017). https://doi.org/10.1109/ICCV.2017.526
42. Tuceryan, M., Jain, A.K.: Texture analysis, pp. 235–276. World Scientific (1993). https://doi.org/10.1142/9789814343138_0010, https://www.worldscientific.com/doi/abs/10.1142/9789814343138_0010
43. Varma, M., Zisserman, A.: Classifying images of materials: Achieving viewpoint and illumination independence. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) Computer Vision — ECCV 2002. pp. 255–271. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
44. Varma, M., Zisserman, A.: A statistical approach to texture classification from single images. International journal of computer vision **62**, 61–81 (2005)
45. Xu, Y., Li, F., Chen, Z., Liang, J., Quan, Y.: Encoding spatial distribution of convolutional features for texture representation. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 22732–22744. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper_files/paper/2021/file/c04c19c2c2474dbf5f7ac4372c5b9af1-Paper.pdf

46. Xue, J., Zhang, H., Dana, K.: Deep texture manifold for ground terrain recognition. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 558–567 (2018). https://doi.org/10.1109/CVPR.2018.00065
47. Xue, J., Zhang, H., Dana, K., Nishino, K.: Differential angular imaging for material recognition. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6940–6949 (2017). https://doi.org/10.1109/CVPR.2017.734
48. Yang, Z., Lai, S., Hong, X., Shi, Y., Cheng, Y., Qing, C.: Dfaen: Double-order knowledge fusion and attentional encoding network for texture recognition. Expert Systems with Applications **209**, 118223 (2022). https://doi.org/https://doi.org/10.1016/j.eswa.2022.118223, https://www.sciencedirect.com/science/article/pii/S0957417422013756
49. Zhai, W., Cao, Y., Zha, Z.J., Xie, H., Wu, F.: Deep structure-revealed network for texture recognition. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11007–11016 (2020). https://doi.org/10.1109/CVPR42600.2020.01102
50. Zhai, W., Cao, Y., Zhang, J., Xie, H., Tao, D., Zha, Z.J.: On exploring multiplicity of primitives and attributes for texture recognition in the wild. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(1), 403–420 (2024). https://doi.org/10.1109/TPAMI.2023.3325230
51. Zhai, W., Cao, Y., Zhang, J., Zha, Z.J.: Deep multiple-attribute-perceived network for real-world texture recognition. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3612–3621 (2019). https://doi.org/10.1109/ICCV.2019.00371
52. Zhang, H., Xue, J., Dana, K.: Deep ten: Texture encoding network. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2896–2905 (2017). https://doi.org/10.1109/CVPR.2017.309