Applied Soft Computing Journal 152 (2024) 111282

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A multilevel pooling scheme in convolutional neural networks for texture image recognition

Lucas O. Lyra [b], Antonio E. Fabris [b], Joao B. Florindo [a,*]

[a] Institute of Mathematics, Statistics and Scientific Computing - University of Campinas, Rua Sérgio Buarque de Holanda, 651, Cidade Universitária "Zeferino Vaz" - Distr. Barão Geraldo, CEP 13083-859, Campinas, SP, Brazil

[b] Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão, 1010, CEP 05508-090, São Paulo, Brazil

ARTICLE INFO

Keywords:
Texture recognition
Convolutional neural networks
Fisher vector
Image descriptors

ABSTRACT

Convolutional neural networks have shown successful results in image classification achieving real-time results superior to the human level. However, texture images still pose some challenge to these models due, for example, to the limited availability of data for training in several problems where these images appear, high inter-class similarity, the absence of a global viewpoint of the object represented, and others. In this context, the present study is focused on improving the accuracy of convolutional neural networks in texture classification. This is done by a hierarchical application of deep filter bank modules combined with Fisher vector pooling. Mid-level local features are extracted from earlier convolutional layers of a pre-trained backbone and combined with high level ones from the last convolutional layer. All local features are treated as equally important and aggregated into a single set of features used for pooling by Fisher vectors. No fine tuning is necessary. The rationale behind this approach is obtaining information that is less domain specific. We verify the effectiveness of our method in texture classification of benchmark databases, as well as on a practical task of Brazilian plant species identification. In both scenarios, Fisher vectors calculated on multiple layers show competitive results with state-of-the-art methods, confirming that early convolutional layers provide important information about the texture image for classification.

1. Introduction

Texture recognition is a task that involves extracting information of the spatial arrangement of the pixel intensities in an image and classifying it. They play an important role in remote sensing [1], materials science [2], medicine [3], agriculture [4], and other fields.

Although CNNs have been quite successful for general image classification, textures are still challenging. This is a consequence of characteristics such as the high inter-class similarity, the lack of a global viewpoint on the analyzed object and the limited availability of data for training in several areas of application, such as medicine [5,6], for example. Training a deep CNN from scratch can lead to overfitting, given limited availability of data. Even if we consider the fine-tuning of pre-trained CNN models on large databases, such as ImageNet, there can be significant domain shift between those large databases and the field of research interest. In this context, the literature has presented a growing number of studies combining CNNs with classical texture descriptors [7–9].

In recent works, transfer learning approaches have been proposed to use the pre-trained CNN capacity of extracting features. Such approaches consist of building new modules around a CNN backbone [10–12]. They leverage mid-level features extracted from earlier layers, which encode more general and fundamental information about the image and are not application-specific. [13]. However, these approaches are built for end-to-end training and associating them with very deep CNN architectures would require large texture datasets. The use of traditional visual texture encoders, such as Fisher Vectors, is suitable in this scenario where data availability is limited [14]. But the full potential of very deep CNNs in feature extraction has not been accordingly explored.

In this context, we propose a method (MultiFisherNet), which aims to explore feature extraction capability of earlier layers of a CNN. This is achieved by proposing a method to combine mid-level local features with high-level ones into a single set of features suitable for traditional encoders. Although high-level features are domain-specific, some information from one domain may be useful in another. With the proposed scheme, we aim to use features from earlier layers that are less domain-specific without losing information from later layers that might be useful. The extracted features are encoded in an orderless manner using a Fisher vector representation. The aggregated features are finally used

* Corresponding author.
E-mail addresses: lucas.oliveira.lyra@alumni.usp.br (L.O. Lyra), aef@ime.usp.br (A.E. Fabris), florindo@unicamp.br (J.B. Florindo).

https://doi.org/10.1016/j.asoc.2024.111282
Received 19 May 2023; Received in revised form 28 December 2023; Accepted 10 January 2024
Available online 17 January 2024

as input to a Support Vector Machine (SVM) classifier. For the feature extraction we use an EfficientNet-B5 backbone [15], pre-trained on ImageNet. This choice takes into consideration the compromise of that architecture between accuracy and computational cost. No fine tuning is necessary in the proposed solution. The major contributions of this study are:

1. We propose a method for hierarchically combining local features extracted from multiple convolutional layers of a CNN;
2. We investigate strategies to combine multiple features with different dimensions based on projection operators;
3. We improve CNN accuracy in visual texture classification;
4. We improve the state-of-the-art performance on the Flickr Material Database (FMD) [16] and the Describable Textures Dataset (DTD) [17];
5. The proposed model achieves a remarkable accuracy of 97.2% in a practical task of identifying Brazilian plant species, significantly outperforming results previously published in the literature [18].

In Section 2, we mention and briefly describe some related works. In Section 3 the theoretical background necessary for the presentation of the proposed method is described, with Section 3.1 giving a brief general description of CNNs and Section 3.2 focusing on how Fisher kernels can be used in texture descriptors. In Section 4 we present the proposed method for visual texture classification. Section 5 shows our procedures to test and validate the performance of our method. In Section 6 we present and discuss the obtained results. Finally, Section 7 presents the general conclusions of our research.

2. Related works

Earlier works on texture recognition were based on using handcrafted features that are invariant to scale, illumination and translation. Scale Invariant Feature Transform (SIFT) [19], Local Binary Patterns (LBP) [20] and variants [21,22] are prominent examples in this regard in the literature. On top of those handcrafted feature extractors, an encoder is needed to combine features into a single descriptor vector that can be used in a discriminative classifier. Traditional encoders include Bag-of-Visual-Words and its variations [23–26], Vector of Locally Aggregated Descriptors (VLAD) [27] and Fisher Vectors (FV) [28,29].

In the last decade, a shift has been made from handcrafted feature extractors to deep neural networks. Since texture recognition databases are frequently very small to train deep neural networks from scratch, most of the proposed methods use pre-trained CNNs on large databases, like ImageNet. One approach is pooling feature maps extracted from CNNs with traditional encoders. Such approach is taken by Cimpoi et al. [14], where VGG architecture [30] is used as feature extractor. Local features are extracted from the last convolutional layer and encoded with FV. Such approach is shown to overcome handcrafted feature extractors in terms of accuracy. Ways to improve FV encoding of local features extracted from the last convolutional layer are studied in Song et al. [31]. They proposed the use of a neural network of locally-connected layers to extract relevant information from FV descriptors. Classification is later performed by a hinge loss layer. A hybrid approach combining SIFT FV descriptors and CNN is explored by Jbene et al. [32]. They propose a neural network composed by two streams. One extracting handcrafted features and the other using a CNN as feature extractor. Both features are concatenated using fully-connected layers. Such approach is evaluated on Xception [33] and Resnet-50 [34] and it is shown to improve CNN accuracy on visual texture classification.

In recent literature, new modules have been built around the pre-trained CNN backbone in order to allow end-to-end training. Such methods usually employ Resnet-18 or Resnet-50 [34] as using deeper CNN backbones would require more data for training. In this context, Zhang et al. [35] build an orderless encoding layer on top of feature maps extracted from the last convolutional layer. Such layer learns a dictionary through soft-assignment clustering and performs sum pooling. Xue et al. [36] combines an orderless encoding layer with global average pooling. The orderless encoding is similar to [35]. Features extracted from both layers are processed through a bilinear model. In Chen et al. [10] feature maps are extracted from multiple convolutional blocks. Those feature maps are resized using bilinear interpolation to a fixed width and height and concatenated on the third dimension, i.e., number of channels. Differential box-counting dimensions are calculated on the resulting feature map and their histograms are later concatenated with global average pooling. Classification is performed by a softmax layer.

Improvements in end-to-end learning are proposed by Mao et al. [11]. They attempt to obtain faster training by removing dictionary learning. This is accomplished by using the last convolutional layer feature map as a dictionary. Feature maps from earlier layers are hard-assigned by a residual encoding module. Those features are later pooled using global average pooling. Yang et al. [12] attempt to build a model more suitable for visual texture, learning both first and second-order feature information. Feature maps from multiple convolutional layers are transformed by frequency attention mechanism based on discrete cosine transform. Extraction is later performed by an attention-based network. Features are encoded by a layer based on bilinear models. Xu et al. [37] attempt to learn more robust and discriminative descriptors by proposing a feature encoding module that combines fractal average pooling and global average pooling. Fractal average pooling is calculated by a hierarchical fractal dimension analysis. Both pooled feature vectors are combined with bilinear pooling.

Other family of works propose not fine-tuning CNN weights. Such approaches can take advantage of very deep CNN models as done in Scabini et al. [38]. They build a feature map using the same approach as in [10]. This feature map is used to train a Randomized Autoencoder module and its weights are used to compose a single dimensional feature vector. The CNN backbone used is ConvNext [39] and they show that their method is suitable for very deep versions of this architecture. SVM is used to perform classification. Florindo et al. [40] propose an encoding approach on top of global average pooling. Visibility graphs are built with the output of the penultimate CNN layer and transformed into feature vectors by calculating node degrees. Classification is performed with SVM and linear discriminant analysis. In another work [41], the input image is transformed calculating an entropy measure over it. Image descriptors are extracted from the penultimate CNN layer for both the original and transformed image and concatenated. They also attempt to perform orderless encoding in [42]. Feature maps extracted from the last convolutional layer are clustered using k-means. Encoding is performed with fuzzy equivalence measures as a way to avoid strong assumptions on the distribution of local features.

Even though the proposed method employs some elements and ideas of the aforementioned approaches, such as the use of a pre-trained backbone with no fine tuning, our approach significantly differs in its strategy to provide the final texture representation. The multi-level Fisher vector scheme has not been previously explored in the literature and ensures significant gains in terms of classification accuracy without adding relevant computational burden, given that no extra learning procedure over the CNN backbone is necessary.

3. Background

In this section, we describe the concepts needed to understand the proposed model. In Section 3.1, we set the basic theory and describe the functioning of CNNs. In Section 3.2, we present a concise summary of Fisher Vector.

3.1. Deep convolutional features

A CNN is a neural network usually developed to handle images. Nodes in each layer can be organized in a multi-dimensional space. Using three dimensions, for example, it is possible to explore relations among neighbor pixels and among color channels.

This type of neural network can be decomposed into two main parts. The first one is used for extracting features from images. It is usually composed by convolutional, pooling, activation and normalization layers. The second part is composed by fully-connected layers whose purpose is classification.

Given a network layer $L_{i,j}$, which is a function that takes as input a tensor and outputs another tensor, a convolutional block B_j can be written as a composition of layers:

$$B_j(X) = L_{1,j} \circ L_{2,j} \circ \cdots (X). \tag{1}$$

CNN architectures for image classification are usually built by composing convolutional blocks, that is, $N = B_1 \circ B_2 \circ \cdots (X)$. Normally, two convolutional blocks B_j and B_k are composed of the same sequence of layers, that is, if $L_{i,j}$ is a pooling layer, then $L_{i,k}$ will also be a pooling layer.

Classical extraction of features is performed by applying convolutional filters to the input image [43]. In this sense, the feature extraction part of a CNN can be seen as a bank of filters, where each channel from each convolutional layer is a particular filter. These features can later be encoded by Fisher Vectors.

3.2. Fisher vector

Let $X = \{\mathbf{x}_t, t = 1 \cdots T \mid \mathbf{x}_t \in \mathbb{R}^D\}$ denote a sample of T observations. Assume that the generation process of X can be modeled by the probability density function u_λ with parameters λ. Then one can characterize the observations in X by the following gradient vector

$$G_\lambda^X = \nabla_\lambda \log u_\lambda(X). \tag{2}$$

The gradient vector given by Eq. (2) can be classified using any classification algorithm. In [44], the Fisher information matrix F_λ is suggested for this purpose and given by

$$F_\lambda = \mathbb{E}_X[G_\lambda^X G_\lambda^{X\prime}], \tag{3}$$

where $\mathbb{E}$ denotes the mathematical expectation of X. From this observation, a Fisher Kernel (FK) to measure similarity between two samples X and Y was proposed. Such kernel is defined by:

$$K_{FK}(X,Y) = G_\lambda^{X\prime} F_\lambda^{-1} G_\lambda^Y. \tag{4}$$

As F_λ^{-1} is positive semi-definite, so is F_λ. Using the Cholesky decomposition $F_\lambda^{-1} = L_\lambda{}' L_\lambda$, the FK can be re-written as:

$$K_{FK}(X,Y) = \mathcal{G}_\lambda^{X\prime} \mathcal{G}_\lambda^Y \tag{5}$$

where

$$\mathcal{G}_\lambda^X = L_\lambda G_\lambda^X. \tag{6}$$

The vector $\mathcal{G}_\lambda^X$ is called *Fisher Vector* (FV). We have that FV and G_λ^X have the same dimensionality [45]. Therefore, we can conclude that performing classification with a linear kernel machine using an FV as feature vector is equivalent to performing a non-linear kernel machine using K_{FK} as kernel. More details on Fisher Vectors can be found in [45].

4. Proposed method

Here we propose an approach to use information from multiple layers of a CNN and Fisher vector encoding to perform classification. The current section is divided into two subsections. In Section 4.1, we show the proposed strategy to build feature vectors. In Section 4.2, we show the classification process using such feature vectors.

4.1. Feature extraction

In the first stage of our methodology, we are interested in combining features extracted from multiple layers of a convolutional neural network. Initially, we take a CNN architecture pre-trained on ImageNet and use it as a feature extractor. We present the texture image as input to the pre-trained CNN and collect the outputs of an arbitrary number of convolutional blocks. All chosen layers contain feature information about the image. However, features extracted from later layers contain higher-level information than features extracted from earlier ones.

Definition 1. Let $X_n = \{\mathbf{x}_t, t = 1, 2, \dots, T_n \mid \mathbf{x}_t \in \mathbb{R}^{D_n}\}$ denote the set of outputs of the nth convolutional block, where D_n and the scalar product $T_n = W_n \cdot H_n$ are, respectively, the number of channels and number of pixels of each channel. We call $\mathbf{x} \in X_n$ a local feature and X_n the set of local features extracted from the nth convolutional block.

Let n and m denote two convolutional blocks, where $D_n \geq D_m$. We denote by $X_{n\to m}$ the set composed by local features from X_n mapped to the space of local features from X_m, that is,

$$X_{n\to m} = \{f_{n,m}(\mathbf{x}_t), t = 1, 2, \dots, T_n \mid \mathbf{x}_t \in X_n\}, \tag{7}$$

where $f_{n,m} : \mathbb{R}^{D_n} \to \mathbb{R}^{D_m}$ is a dimension reduction function. If the two convolutional blocks have the same number of channels, that is, $D_n = D_m$, the function $f_{n,m}$ is an identity function. Otherwise, if $D_n > D_m$, each component $f_{n,m}^d$ in $f_{n,m}$ is given by

$$f_{n,m}(\mathbf{x}_t)_d = \begin{cases} \max_{m=0}^{K} x_t^{S\cdot d+m} & \text{for max pooling,} \\ \frac{1}{K+1}\sum_{m=0}^{K} x_t^{S\cdot d+m} & \text{for average pooling,} \end{cases} \tag{8}$$

where x_t^d is the dth component of $\mathbf{x}_t$ and

$$S = \left\lfloor \frac{D_{n-i}}{D_{n-k}} \right\rfloor, \tag{9}$$

$$K = D_{n-i} - D_{n-k} \cdot S. \tag{10}$$

Additionally to average and max pooling, we also propose the use of Principal Component Analysis (PCA) to calculate $f_{n,m}$.

Given $k+1$ convolutional blocks $n, n-1, \dots, n-k$, the combined set of local features X is constructed by pooling the sets $X_n, X_{n-1}, \dots, X_{n-k}$. Such pooling is given by Eq. (11). The proposed scheme for feature extraction is exemplified in Fig. 1, where $k = 2$.

$$X = \bigcup_{i=0}^{k} X_{n-i\to n-k} \tag{11}$$

Fisher vectors are calculated with the combined set of local features $X = \{\mathbf{x}_t, t = 1, 2, \dots, T \mid \mathbf{x}_t \in \mathbb{R}^D\}$, where $T = \sum_{i=0}^{k} T_{n-i}$ and $D = D_{n-k}$. We assume that each local feature $\mathbf{x}_t$ is independently generated by the u_λ distribution. In this case, Eq. (6) becomes:

$$\mathcal{G}_\lambda^X = L_\lambda \frac{1}{T} \sum_{t=1}^{T} \nabla_\lambda \log u_\lambda(\mathbf{x}_t). \tag{12}$$

We choose u_λ to be a Gaussian Mixture Model (GMM) composed by K Gaussian distributions, which represent each visual word in the learned dictionary. In this model, u_λ is written as

$$u_\lambda(\mathbf{x}) = \sum_{i=1}^{K} w_i u_i(\mathbf{x}), \tag{13}$$

where $\lambda = \{w_i, \mu_i, \Sigma_i, i = 1, \dots, K\}$ and w_i, μ_i, Σ_i denote, respectively, the weight, mean and covariance matrix associated with Gaussian u_i.

Let $\gamma_i(\mathbf{x}_t)$ denote the probability of an observation $\mathbf{x}_t$ to be generated by the Gaussian u_i:

$$\gamma_i(\mathbf{x}_t) = \frac{w_i u_i(\mathbf{x}_t)}{\sum_{j=1}^{K} w_j u_j(\mathbf{x}_t)}. \tag{14}$$

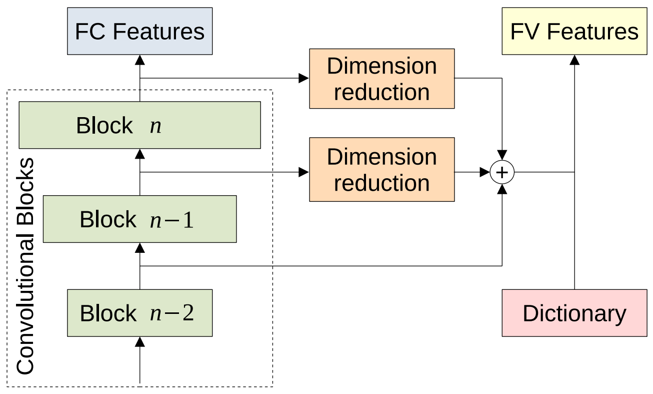


Fig. 1. Feature extraction with the proposed method for the particular case of three sets. Here $D_{n-2} < \min(D_{n-1}, D_n)$, thus, dimensionality reduction needs to be applied to local features in X_n and X_{n-1}. FC Features are extracted from fully-connected layers after block n. FV Features are built from combined local features and a learned Dictionary.

We assume that covariance matrices are diagonal given that any distribution can be approximated with an arbitrary precision by a weighted sum of Gaussians with diagonal covariances [28]. We denote $\sigma_i^2 = \text{diag}(\Sigma_i)$. Using the values of L_λ and $\nabla_\lambda \log u_\lambda(X)$ derived in [28], we can rewrite Eq. (6) as:

$$\mathcal{G}^X_{w_i^d} = \frac{1}{T\sqrt{w_i}} \sum_{t=1}^{T} \left(\gamma_i(\mathbf{x}_t) - w_i\right), \tag{15}$$

$$\mathcal{G}^X_{\mu_i^d} = \frac{1}{T\sqrt{w_i}} \sum_{t=1}^{T} \gamma_i(\mathbf{x}_t) \left(\frac{x_t^d - \mu_i^d}{\sigma_i^d}\right), \tag{16}$$

$$\mathcal{G}^X_{\sigma_i^d} = \frac{1}{T\sqrt{2w_i}} \sum_{t=1}^{T} \gamma_i(\mathbf{x}_t) \left[\frac{(x_t^d - \mu_i^d)^2}{(\sigma_i^d)^2} - 1\right]. \tag{17}$$

Eqs. (15), (16), (17) are used to calculate each component of a Fisher Vector from any set of local features extracted from CNN's convolutional layers. This feature vector is here denoted as FV. Additionally, we extract information from the last fully-connected layer. The extraction process is performed by removing the output layer. The resulting feature vector is here called FC.

4.2. Classification

In order to perform classification, we employ L2 and Power normalization over FV features as proposed in [29]. Although there are various normalization algorithms applied with different image types, they may lead to increase in computational costs. Therefore, an efficient normalization algorithm has been used in this work to obtain high performance. Additionally, we combine information from FC and normalized FV features by concatenating the two vectors. The resulting feature vector is referred as FC+FV.

Classification is performed with Support Vector Machine (SVM), using the Bhattacharyya coefficient given in Definition 2 as kernel.

Definition 2. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$. The Bhattacharyya coefficient is given by the following measure of distance:

$$K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \text{sign}(x_i y_i)\sqrt{|x_i y_i|}. \tag{18}$$

Note that the Bhattacharyya coefficient can be rewritten as

$$K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y}), \tag{19}$$

where $\phi(\mathbf{x})$ is a vector whose coordinates are given by

$$\phi(\mathbf{x})_i = \text{sign}(x_i)\sqrt{|x_i|}. \tag{20}$$

Thus, applying the transformation given by Eq. (20) to the feature vectors, we are able to perform the classification with a linear SVM. As we are dealing with multi-class problems, we use one-vs-rest strategy to perform classification.

5. Experimental setup

In this section we describe how we evaluate our proposed methodology. Our base model uses the EfficientNet-B5 architecture [15] with pre-trained ImageNet weights as feature extractor.

The databases used for method evaluation are KTH-TIPS2-b, FMD, DTD, UIUC, UMD, and GTOS. The database used in our practical task is 1200Tex. All these databases are described in the following paragraphs.

KTH-TIPS2-b [46] consists of 4 samples of images from 11 materials. Each sample is presented in 9 different scales, 3 poses and 4 lighting conditions. This represents a total of 108 images with varying sizes per material per sample. In each round, we use 3 samples for training and 1 sample for testing.

FMD [16] consists of 10 classes containing 100 images each. Each image has a size of 512 × 384. We run 10 training/testing rounds, each randomly selecting half of the database for training and using the other half for testing.

DTD [17] consists of 5640 images with varying sizes divided into 47 categories. This results in 120 images per class, which are divided into three equal parts: training, validation and testing. The database contains 10 splits of the data. For each one, we use training and validation parts for adjusting our model and the remaining part for testing.

UMD [47] consists of 25 classes containing 40 images each. All images have a dimension of 1280 × 960. We evaluated our method following the same protocol as FMD.

UIUC [24], as UMD, consists of 1000 images evenly divided into 25 classes. Each image has resolution of 640 × 480. In order to evaluate our method in this dataset, we use the same protocol applied to FMD.

1200Tex [18] consists of 1200 leaf surface images of 20 Brazilian plant species (classes). Each class contains 60 samples. We applied the same protocol followed in FMD to choose training and testing datasets.

GTOS [48] consists of over 34,243 images divided into 40 material categories. Each class contains a different amount of images. Each material sample was taken in 19 viewing angles. The database contains 5 splits. In each split, the dataset is divided in training and testing parts with a varying ratio. Given computational costs of GMM algorithm, we undersample the training set, randomly choosing 2 out of 19 viewing angles from each sample for training purposes.

In order to fairly compare the methodology behavior in all datasets, we fix image width to obtain a similar number of local features. In the case of datasets with different image sizes, we also fix height to the same value as width. The standard value adopted for the image width is 320. Additionally, we do not apply any denoising methods to the images.

Fisher Vectors are calculated using 16, 32, 48, and 64 kernels. Unless otherwise specified, we use 16 kernels. This value corresponds to the number of components in the GMM. All remaining parameters are left unaltered. The GMM is initialized using k-means algorithm. We set the non-negative regularization, a value that is added to the diagonal covariance, to the local features standard deviation multiplied by 10^{-4}. The algorithm tolerance is set to 10^{-3}.

Local features are extracted from 1 to 4 convolutional layers. Unless stated otherwise, we use two convolutional layers. The layers are chosen as the last convolutional layer of each used convolutional block. Convolutional blocks are selected sequentially starting from the last block. Thus, for n layers, the last n convolutional blocks are used. One layer means that the last convolutional layer is used, as proposed in [14]. We call high-level features those that are extracted from the last convolutional layer while mid-level features are the ones extracted from earlier layers.

Table 1
Comparison of different dimensionality reduction methods with features provided by Fisher Vectors.

Dataset	Method	Accuracy	Precision	Recall	F1-Score
KTH-TIPS2-b	PCA	$92.3_{\pm 5.2}$	$93.6_{\pm 4.4}$	$92.3_{\pm 5.2}$	$91.7_{\pm 5.9}$
	Avg Pooling	$92.3_{\pm 4.8}$	$93.2_{\pm 4.4}$	$92.3_{\pm 4.8}$	$91.7_{\pm 5.5}$
	Max Pooling	$92.5_{\pm 5.2}$	$93.5_{\pm 4.7}$	$92.5_{\pm 5.2}$	$91.8_{\pm 5.9}$
	AutoEncoder	$92.6_{\pm 5.0}$	$93.5_{\pm 4.4}$	$92.6_{\pm 5.0}$	$92.0_{\pm 5.6}$
FMD	PCA	$88.8_{\pm 0.8}$	$88.9_{\pm 0.8}$	$89.0_{\pm 0.8}$	$88.8_{\pm 0.8}$
	Avg Pooling	$81.0_{\pm 0.7}$	$81.0_{\pm 0.7}$	$81.3_{\pm 0.7}$	$80.7_{\pm 0.8}$
	Max Pooling	$78.2_{\pm 0.9}$	$78.1_{\pm 0.9}$	$78.5_{\pm 0.9}$	$77.8_{\pm 1.0}$
	AutoEncoder	$90.3_{\pm 0.6}$	$90.3_{\pm 0.6}$	$90.5_{\pm 0.7}$	$90.2_{\pm 0.6}$
DTD	PCA	$80.2_{\pm 0.5}$	$80.3_{\pm 0.5}$	$80.2_{\pm 0.5}$	$80.0_{\pm 0.5}$
	Avg Pooling	$77.1_{\pm 0.5}$	$77.1_{\pm 0.6}$	$77.1_{\pm 0.5}$	$76.8_{\pm 0.6}$
	Max Pooling	$75.3_{\pm 0.7}$	$75.4_{\pm 0.8}$	$75.3_{\pm 0.7}$	$75.1_{\pm 0.8}$
	AutoEncoder	$80.1_{\pm 0.4}$	$80.2_{\pm 0.4}$	$80.1_{\pm 0.4}$	$79.9_{\pm 0.4}$
UMD	PCA	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$
	Avg Pooling	$99.7_{\pm 0.1}$	$99.7_{\pm 0.1}$	$99.7_{\pm 0.1}$	$99.7_{\pm 0.1}$
	Max Pooling	$99.7_{\pm 0.1}$	$99.7_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.7_{\pm 0.1}$
	AutoEncoder	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$
UIUC	PCA	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$
	Avg Pooling	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$
	Max Pooling	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$
	AutoEncoder	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$
GTOS	PCA	$85.0_{\pm 1.1}$	$84.8_{\pm 2.4}$	$85.1_{\pm 1.4}$	$82.9_{\pm 1.9}$
	Avg Pooling	$84.1_{\pm 1.2}$	$83.7_{\pm 2.2}$	$84.0_{\pm 1.4}$	$81.8_{\pm 1.7}$
	Max Pooling	$84.0_{\pm 1.6}$	$83.7_{\pm 2.5}$	$83.8_{\pm 1.9}$	$81.9_{\pm 2.3}$
	AutoEncoder	$85.4_{\pm 1.1}$	$85.3_{\pm 2.0}$	$85.3_{\pm 1.3}$	$83.4_{\pm 1.7}$

In all experiments, we use EfficientNet-B5 [15] architecture for feature extraction. We also show the behavior of the model in other CNN backbones. In these cases, the backbone used is explicitly mentioned.

In all considered backbones, local features of the previous convolutional layers belong to a lower dimensional space than those of the later layers. Thus, for every experiment using 2 or more convolutional layers for feature extraction, we use a method for dimensionality reduction. The standard method used is PCA, using the randomized Singular Value Decomposition proposed in [49]. We also consider Average pooling, Max pooling and AutoEncoder. The proposed AutoEncoder consists of two linear layers, one for encoding, and the other for decoding. No activation function is applied. Average pooling and Max pooling are 1-dimensional and used exclusively for dimensionality reduction.

For classification purposes, linear SVM is used with regularization parameter set to 1.0, class weights set to 1.0 and tolerance set to 10^{-3}. We consider the use of other classifiers, such as SVM with Radial Basis Function (RBF) kernel, Multi-Layer Perceptron (MLP), and Linear Discriminant Analysis (LDA). For the SVM with RBF kernel, we use the same parameters as in the linear SVM. For MLP, we use one hidden layer with 100 neurons, rectified linear unit (ReLU) as activation function and Adam [50] for optimization. The learning rate is kept constant and equal to 0.001. In the specific case of linear SVM, Eq. (20) is applied to the feature vectors.

We evaluate how the accuracy of Fisher Vectors in describing the original image is affected by the number of kernels and the number of local features. In the first case, we change the number of GMM components. In the second case, we vary the width of the input image. The image height is changed accordingly to maintain the original aspect ratio.

We also evaluate the model robustness by adding noise in the input signal. We use Gaussian noise with zero mean and variance equal to the image mean divided by the Signal-to-Noise Ratio (SNR). The noise is added after resizing and normalizing the input image. We compared the model accuracy with SNR ranging from 100 to 10.

Additionally, we include ablation studies where we evaluate how the model behaves when mid-level information is removed and when high-level information is removed. The default setting is the same as previous experiments, using two convolutional layers, the last one from block 6 representing high-level features and the last one from block 5 representing mid-level features.

Finally, we compare our base model with alternative state-of-the-art approaches. In this comparison, we use optimized parameters for our model. Such parameters are determined as those that achieve the best overall performance on the validation sets. We conclude our experiments by applying our model to a practical task that consists in the identification of Brazilian plant species based on the scanned image of the leaf surface.

6. Results and discussion

In this section we present the results obtained using the experimental setup described in Section 5. We show how they accomplished to verify the effectiveness of the proposed methodology in texture classification. All results include confidence intervals.

The accuracy of our model depends on how well the proposed methods for dimensionality reduction of local features perform in preserving information. We show the effectiveness of each method across all benchmark databases in Table 1. For both FMD and DTD, PCA performs better with statistical significance in preserving local feature information, which is reflected in the FV accuracy. In KTH-TIPS2-b, UMD, and UIUC, the methods show similar effectiveness in preserving information. On the remaining datasets, PCA and Autoencoder are numerically advantageous over Average and Max Pooling. In the particular case of FMD, AutoEncoder outperforms all the other methods. One particular characteristic of this dataset is the presence of object-like data. This may indicate that AutoEncoder might be suitable for visual textures “in-the-wild”. An important point that should be observed, however, is that PCA is significantly less expensive in terms of computational resources than AutoEncoder, which makes it a natural choice in practical application.

Our second experiment shows the impact of using mid-level local features in combination with high-level ones. This is done by increasing the number of layers for feature extraction. The choice of layers is done as described in Section 5. The accuracy of our model in ranking each benchmark database is shown in Fig. 2. For DTD, KTH-TIPS2-b, UIUC, and GTOS databases the use of mid-level features extracted

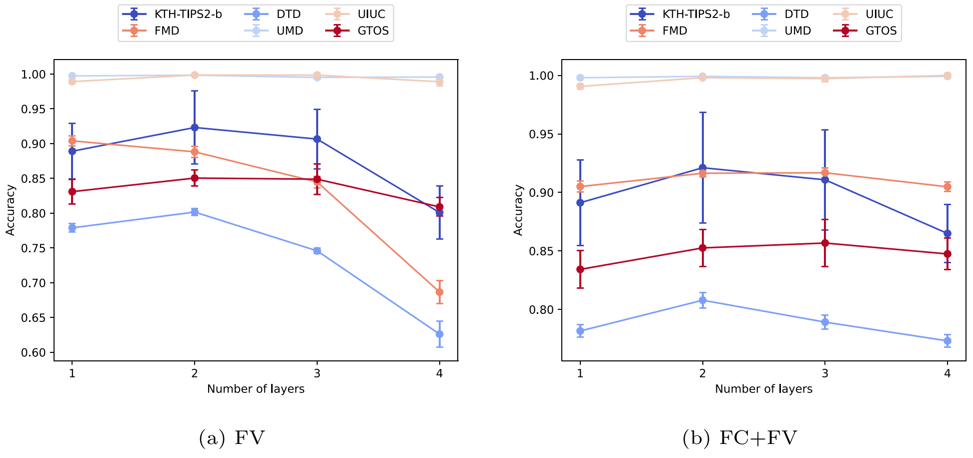


Fig. 2. Variation of accuracy with different number of layers. 1 layer means no mid-level features are used. Two layers seems to yield the best performance overall.

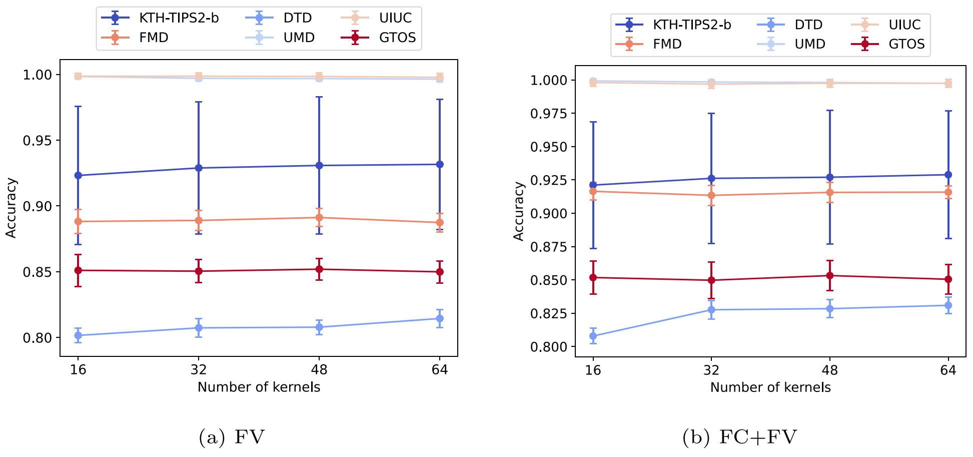


Fig. 3. Variation of accuracy of our method according to the number of Gaussian distributions (kernels) used in GMM. Few variation is observed, which means 16 to 32 Gaussian distributions is enough to model the underlying distribution of local features.

from convolutional blocks prior to the last block increased accuracy. This was expected since such databases have a greater domain shift from ImageNet. The decrease in accuracy when using 4 layers is a consequence of the loss of high-level information that dimensionality reduction causes. The choice of best number of layers to be used depends on the database, therefore for new applications, a small portion of data should be reserved in order to test for best configuration.

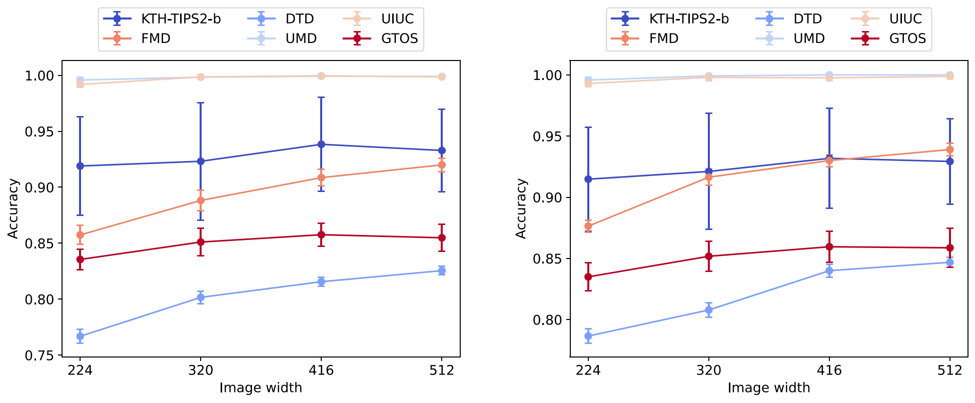
As mentioned in Section 5, the accuracy of our method can be affected by the number of Gaussian distributions that we choose to model u_λ. Those distributions are also called number of *kernels* or *visual words*. We used 16, 32, 48 and 64 kernels in benchmark tests. The results are shown in Fig. 3. We observed very little variation of accuracy across all databases. This probably means that 16 Gaussian distributions are enough to model the underlying distribution that generates local features for most databases. An exception is observed in the DTD, where at least 32 Gaussian distributions are needed to achieve the best results. This might be due to the fact that DTD is the most challenging database analyzed here.

Continuing with our experimental protocol, we evaluate how increasing the number of local features by varying image size affects the model. As mentioned in Section 5, we set all images to the same width in order to compare the model behavior with a similar number of features for all databases. The results are presented in Fig. 4. As expected, the increase in image resolution improved accuracy in most databases. This improvement is not only due to the number of local features, but also how specific a local feature is. If the image resolution is too low, information from small regions of the image may be lost. A condition for the use of generative models to be beneficial for accuracy is that local features should describe small regions rather than large ones. In KTH-TIPS2-b, the model seems to have achieved its limit with image width around 416, making further increases in the number of local features detrimental.

We proposed to associate the calculated Fisher Vectors with a linear SVM, but we also evaluated how the model behaves with different classifiers. In Table 2, we show the performance of four different classifiers on our benchmark datasets. The parameters used in each classifier are described in Section 5. The linear SVM with Bhattacharyya coefficient as kernel numerically outperforms all classifiers considered on all datasets. This seems to indicate that the proposed classifier is the optimal choice for the specific task of visual texture classification.

We proceed to evaluate the impact of noise on our model accuracy. Noise may occur in real-world scenarios and it is important for the model accuracy to be relatively tolerant to that. The proposed scheme for adding noise to the images is explained in Section 5. As seen in Fig. 5, in KTH-TIPS2-b, UIUC, and UMD, the accuracy is not negatively impacted by noise addition. In the remaining databases, accuracy is not impacted in a statistically significant way with SNR greater or equal to 60. This shows that the model is robust to noise levels where SNR is not smaller than 60, which corresponds to the most realistic scenarios in most applications.

We include an ablation study to verify how the model behaves when mid-level and high-level features are removed. The results are presented in Table 3 and were obtained using FC+FV as feature vector. For all the challenging datasets, we see that removing mid-level features



(a) FV (b) FC+FV

Fig. 4. Accuracy of our method according to image size. The number of local descriptors extracted from the neural network is directly proportional to image size. Original shape is maintained in databases where all images have same shape, otherwise, image height is set to be equal to image width.

Table 2
Comparison of different classifiers when FV is used as feature vector to describe the input image. In parentheses, we indicate the kernel used in each SVM.

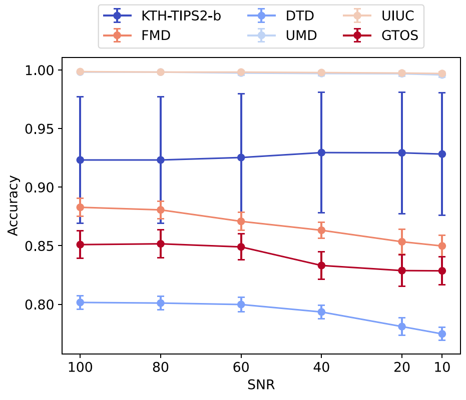
Dataset	Classifier	Accuracy	Precision	Recall	F1-Score
KTH-TIPS2-b	SVM (Linear)	$92.3_{\pm 5.2}$	$93.6_{\pm 4.4}$	$92.3_{\pm 5.2}$	$91.7_{\pm 5.9}$
	SVM (RBF)	$92.2_{\pm 5.6}$	$93.4_{\pm 4.8}$	$92.2_{\pm 5.6}$	$91.6_{\pm 6.3}$
	MLP	$92.2_{\pm 5.4}$	$93.2_{\pm 4.6}$	$92.2_{\pm 5.4}$	$91.5_{\pm 6.1}$
	LDA	$92.1_{\pm 5.3}$	$93.4_{\pm 4.2}$	$92.1_{\pm 5.3}$	$91.7_{\pm 5.7}$
FMD	SVM (Linear)	$88.8_{\pm 0.8}$	$88.9_{\pm 0.8}$	$89.0_{\pm 0.8}$	$88.8_{\pm 0.8}$
	SVM (RBF)	$85.4_{\pm 1.0}$	$86.2_{\pm 0.9}$	$85.6_{\pm 1.0}$	$85.5_{\pm 1.0}$
	MLP	$86.6_{\pm 1.0}$	$86.8_{\pm 1.0}$	$86.8_{\pm 1.0}$	$86.5_{\pm 1.1}$
	LDA	$82.8_{\pm 1.1}$	$83.6_{\pm 1.1}$	$83.0_{\pm 1.1}$	$82.9_{\pm 1.1}$
DTD	SVM (Linear)	$80.2_{\pm 0.5}$	$80.3_{\pm 0.5}$	$80.2_{\pm 0.5}$	$80.0_{\pm 0.5}$
	SVM (RBF)	$79.3_{\pm 0.6}$	$80.2_{\pm 0.5}$	$79.3_{\pm 0.6}$	$79.4_{\pm 0.6}$
	MLP	$80.4_{\pm 0.3}$	$80.7_{\pm 0.3}$	$80.4_{\pm 0.3}$	$80.3_{\pm 0.3}$
	LDA	$52.6_{\pm 4.5}$	$59.7_{\pm 3.2}$	$52.6_{\pm 4.5}$	$54.3_{\pm 4.3}$
UMD	SVM (Linear)	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$
	SVM (RBF)	$99.6_{\pm 0.2}$	$99.6_{\pm 0.1}$	$99.6_{\pm 0.1}$	$99.6_{\pm 0.1}$
	MLP	$99.5_{\pm 0.2}$	$99.6_{\pm 0.2}$	$99.6_{\pm 0.1}$	$99.5_{\pm 0.2}$
	LDA	$98.8_{\pm 0.3}$	$98.9_{\pm 0.3}$	$98.9_{\pm 0.3}$	$98.8_{\pm 0.3}$
UIUC	SVM (Linear)	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$
	SVM (RBF)	$99.4_{\pm 0.2}$	$99.4_{\pm 0.2}$	$99.5_{\pm 0.2}$	$99.4_{\pm 0.2}$
	MLP	$99.5_{\pm 0.2}$	$99.5_{\pm 0.2}$	$99.5_{\pm 0.2}$	$99.5_{\pm 0.2}$
	LDA	$98.4_{\pm 0.3}$	$98.5_{\pm 0.3}$	$98.5_{\pm 0.3}$	$98.4_{\pm 0.3}$
GTOS	SVM (Linear)	$85.0_{\pm 1.1}$	$84.8_{\pm 2.4}$	$85.1_{\pm 1.4}$	$82.9_{\pm 1.9}$
	SVM (RBF)	$82.7_{\pm 1.7}$	$83.3_{\pm 1.7}$	$81.6_{\pm 2.2}$	$79.9_{\pm 2.0}$
	MLP	$84.8_{\pm 1.3}$	$84.8_{\pm 2.0}$	$84.6_{\pm 1.5}$	$82.5_{\pm 1.9}$
	LDA	$83.4_{\pm 1.7}$	$84.9_{\pm 1.3}$	$82.4_{\pm 1.9}$	$81.2_{\pm 1.9}$

impact negatively on all measures while removing high-level features has almost no impact. This later result is probably because high-level information are being encoded in FC.

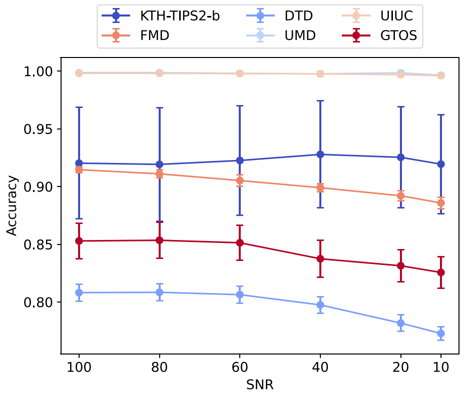
The following results were obtained by an optimal configuration determined over validation sets. The image width is kept at 512, the number of kernels at 48, the number of layers at 2 and PCA is used as the method for dimensionality reduction. The SVM and GMM parameters (except for the number of components) are not changed. These parameters do not affect the results significantly. Although GMM using expected-maximization (EM) algorithm is known to be sensitive to the choice of parameters [51], under certain conditions, even poor initialization converges to near-globally optimum solution [52]. This turns out to be the case in our context, as changing the parameters in GMM did not result in significant changes in accuracy results of our model.

In the following paragraphs we detail how our method behaves in the benchmark databases by showing how much confusion is presented in each database. In UMD and UIUC, no significant confusion can be observed, therefore such databases are left out of our analysis. All confusion matrices were obtained using FC+FV as feature vector. In order to consider all train/test rounds, the presented matrix for each dataset is the sum of the matrices from each round. We also include t-SNE visualization of the two most important components of FC+FV. In this case, we plot the visualization of the worst round in terms of accuracy.

In KTH-TIPS2-b, most noticeable problems are the classification of examples from class 5 (cotton) and class 11 (wool), as shown in Fig. 6(a). In the case of cotton, it is mostly confused with class 8 (linen) and wool. Wool is mostly confused with cotton, but there is also confusion with linen and class 3 (corduroy). Interestingly, most part of the confusion is among textile textures, which are indeed challenging to classify, given that they frequently share fairly similar texture patterns. In Fig. 6(b), we see a small overlap between classes 5 and 10 (wood). The most noticeable overlap is between wool and corduroy. Not all



(a) FV



(b) FC+FV

Fig. 5. Accuracy of our method according to SNR variation. Gaussian noise was added to images after the application of resizing and normalization.

Table 3
Impact of excluding high and mid level features. Dashes in column *Excluded Features* represents the basic case, when both features are used. In the case of high-level features, local features from the last convolutional block are not used. In the case of mid-level features, local features from the convolutional block previous to the last one are not used.

Dataset	Excluded features	Accuracy	Precision	Recall	F1-Score
KTH-TIPS2-b	–	$92.1_{\pm 4.7}$	$93.4_{\pm 4.0}$	$92.1_{\pm 4.7}$	$91.5_{\pm 5.4}$
	High-level	$92.1_{\pm 4.7}$	$93.2_{\pm 4.2}$	$92.1_{\pm 4.7}$	$91.5_{\pm 5.4}$
	Mid-level	$89.1_{\pm 3.6}$	$90.8_{\pm 3.4}$	$89.1_{\pm 3.6}$	$88.4_{\pm 3.9}$
FMD	–	$91.6_{\pm 0.3}$	$91.7_{\pm 0.3}$	$91.8_{\pm 0.4}$	$91.6_{\pm 0.3}$
	High-level	$91.8_{\pm 0.3}$	$91.8_{\pm 0.3}$	$91.9_{\pm 0.3}$	$91.8_{\pm 0.3}$
	Mid-level	$90.5_{\pm 0.5}$	$90.6_{\pm 0.5}$	$90.6_{\pm 0.5}$	$90.5_{\pm 0.5}$
DTD	–	$80.8_{\pm 0.7}$	$80.9_{\pm 0.7}$	$80.8_{\pm 0.7}$	$80.6_{\pm 0.7}$
	High-level	$80.8_{\pm 0.6}$	$80.9_{\pm 0.6}$	$80.8_{\pm 0.6}$	$80.6_{\pm 0.6}$
	Mid-level	$78.2_{\pm 0.5}$	$78.3_{\pm 0.6}$	$78.2_{\pm 0.5}$	$78.0_{\pm 0.6}$
UMD	–	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$
	High-level	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.8_{\pm 0.1}$
	Mid-level	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$
UIUC	–	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$
	High-level	$99.8_{\pm 0.1}$	$99.8_{\pm 0.1}$	$99.9_{\pm 0.1}$	$99.8_{\pm 0.1}$
	Mid-level	$99.1_{\pm 0.2}$	$99.1_{\pm 0.2}$	$99.1_{\pm 0.2}$	$99.1_{\pm 0.2}$
GTOS	–	$85.3_{\pm 1.6}$	$84.8_{\pm 2.8}$	$85.1_{\pm 1.7}$	$83.0_{\pm 2.2}$
	High-level	$85.1_{\pm 1.4}$	$84.8_{\pm 2.6}$	$84.9_{\pm 1.7}$	$82.8_{\pm 2.1}$
	Mid-level	$83.4_{\pm 1.6}$	$82.6_{\pm 2.6}$	$83.2_{\pm 1.6}$	$81.0_{\pm 1.9}$

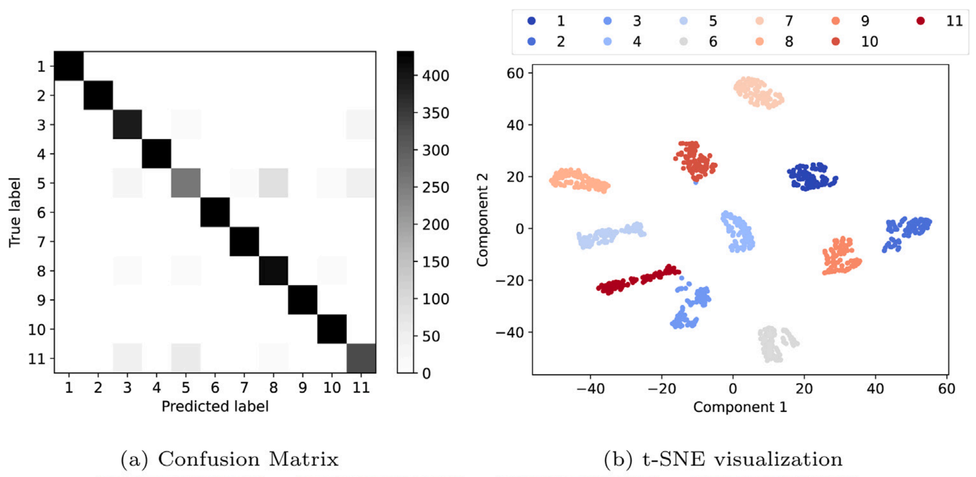
confusion is expected to be seen in t-SNE visualization. Superposition may not allow seeing overlaps and some confusion may belong to other rounds of train/test.

In FMD, our model had most problems distinguishing classes 5 (metal) and 6 (paper) from other classes, as shown in Fig. 7. Paper is mostly confused with class 1 (fabric), while 5 (metal) is confused with classes 7 (plastic) and 3 (glass). The presence of confusion in this case could be explained by the fact that underlying objects made out from these materials can have similar shapes and/or colors. In Fig. 7(b), the t-SNE visualization shows how much overlap exists among classes, being almost impossible to distinguish them using only two components. One of the major sources of confusion, classes 5 and 7, for example, are completely mixed together and overlap other classes.

In DTD, the most notorious classification problem of our model is perceived in class 2 (blotchy), where less than 50% of samples are correctly classified, as shown in Fig. 8. These samples are mostly mistaken by classes 38 (stained), 43 (veined) and 24 (marbled). The confusion between blotchy and stained was expected, as images from both classes are very similar. Confusion with veined images can be explained by the presence of veins in images of blotched leaves of the blotchy class. In the case of marbled images, the details in marble can be interpreted as blotches, justifying confusion between marbled and blotchy. In t-SNE visualization, we see a significant overlap among almost all classes. This is expected as some ambiguity exists among classes. The overlap also indicates that two components are not enough to successfully distinguish classes.

In GTOS, there are three classes where less than half of samples are correctly classified: 9 (Dry Grass), 27 (Rusted Cover) and 37 (Stone Mud), as shown in Fig. 9. Most of images from Dry Grass are classified as class 12 (Grass). This confusion may be due to fragments of grass in the middle of the dry grass in those images. In the case of Rusted Cover images, almost 40% are misclassified as class 2 (Aluminum). The presence of rust in Aluminum images can explain this confusion. About one third of Stone Mud samples are misclassified as class 19 (Mud Puddle). The confusion between those classes was expected as there are stones in some puddle images. In t-SNE visualization, we see that, although classes do not seem to overlap, most are spreaded in more than one cluster. Some clusters from different classes are too close together for a linear SVM to be capable of correctly separating them.

In Table 4, we list the accuracy of several methods in the literature of texture recognition compared with the proposed approach. We use three CNN backbones associated with our method in this comparison: EfficientNet-B5 [15], ConvNeXt-T [39], and RegNetY 6.4G [53]. All these backbones have a similar number of parameters of around 30



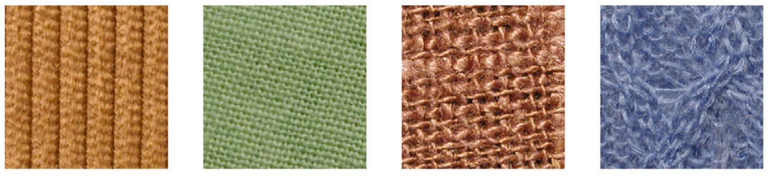
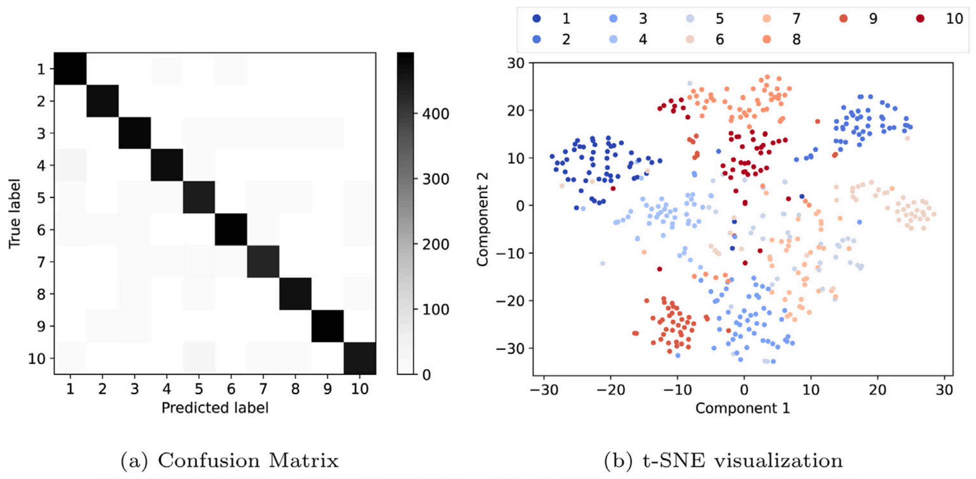

(c) Class examples

Fig. 6. On top: confusion matrix for KTH-TIPS2-b and t-SNE visualization of the worst train/test round in terms of accuracy. On bottom: from left to right, examples of classes Corduroy, Cotton, Linen, and Wool.



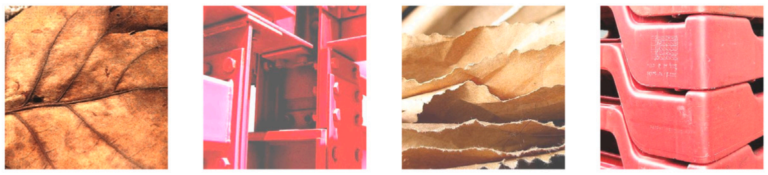
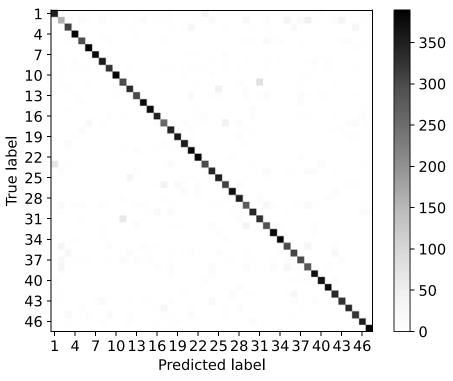

(c) Class examples

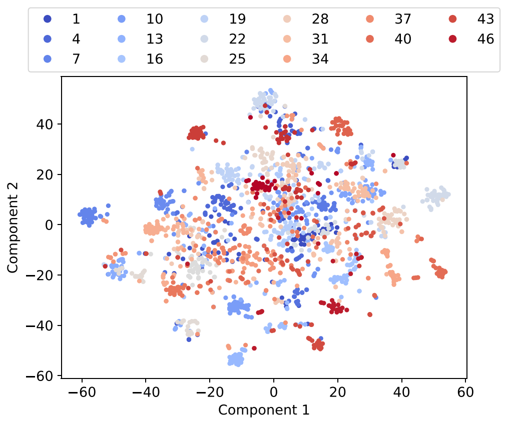
Fig. 7. On top: confusion matrix for FMD and t-SNE visualization of the worst train/test round in terms of accuracy. On bottom: from left to right, examples of classes Foliage, Metal, Paper, and Plastic.

million. We indicate in parentheses which backbone was used to perform feature extraction. We also include the fine-tuning of all network architectures to show how this straightforward approach can perform in texture analysis. Parameters for fine-tuning the CNNs were empirically determined as being the best configuration in benchmark databases.

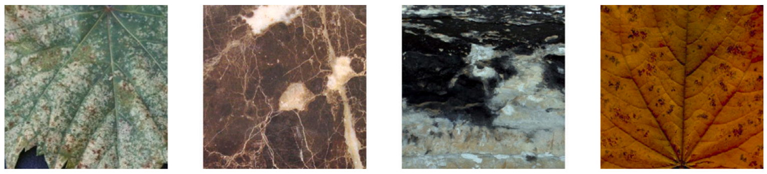
The proposed method, when associated with EfficientNet, is competitive with state-of-the-art deep learning approaches on all databases evaluated. When using ConNeXt as backbone, we have superior results in most databases, except KTH-TIPS2-b, where RAdam is statistically equivalent. With RegNetY, results are competitive in FMD, DTD, UMD, and UIUC. In general, EfficientNet seems to be an optimal choice for local feature extraction. Local features extracted from ConvNeXt or RegNetY, on the other hand, may not be optimal for visual texture classification. The dimensions of local features vary among these backbones, with RegNetY having the greatest number of dimensions and EfficientNet having the least. The increase in the number of dimensions



(a) Confusion Matrix

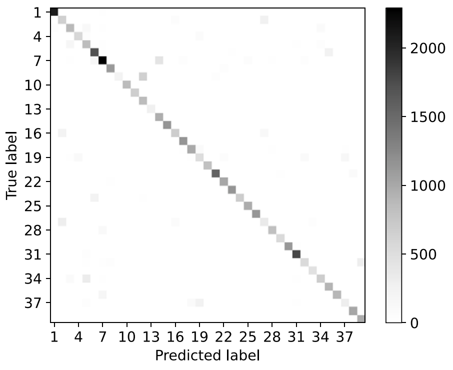


(b) t-SNE visualization

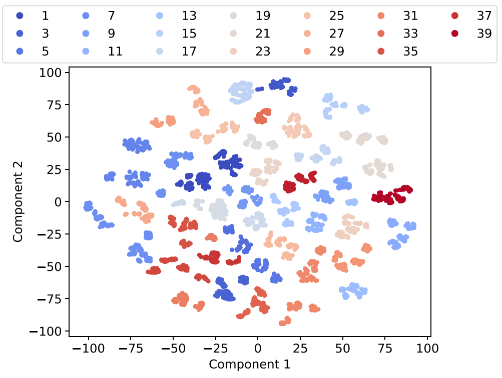


(c) Class examples

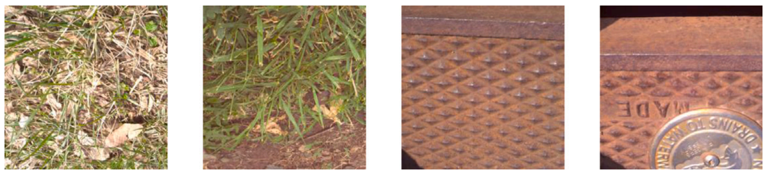
Fig. 8. On top: confusion matrix for DTD and t-SNE visualization of the worst train/test round in terms of accuracy. On bottom: from left to right, examples of classes Blotchy, Marbled, Stained, and Veined.



(a) Confusion Matrix



(b) t-SNE visualization



(c) Class examples

Fig. 9. On top: confusion matrix for GTOS and t-SNE visualization of the worst train/test round in terms of accuracy. On bottom: from left to right, examples of classes Dry Grass, Grass, Rusted Cover, and Aluminum.

leads to increased complexity of the GMM optimization, which may result in poor choices of local minima.

When we compare our method with the fine-tuned CNN backbones, we see how CNN accuracy in texture classification can be enhanced by the proposed multilevel pooling scheme. The main improvement provided by our method can be noticed in RegNetY. In DTD and GTOS, the application of our method increased accuracy in about 13%. In DTD and FMD, fine-tuned EfficientNet and ConvNeXt were capable of achieving competitive results with state-of-the-art. In both datasets, our method shows significant improvements in CNN accuracy, improving the state-of-the-art.

In UIUC and UMD, our method with EfficientNet is capable of correctly classifying all test samples, a result not yet achieved in the literature. Specifically in UMD, all backbones provided the same result. In the particular case of GTOS, all compared methods use the entire training set for training purposes while we use a small portion of it. Even using less data, our model, using EfficientNet or ConvNeXt, achieved similar results to CLASSNet and FENet, which are the current

Table 4
Accuracy comparison with other methods in literature. In the first three rows, we include the fine-tuning performance of the CNN architectures we are using. All results shown are obtained directly from the original paper of each method. Non-published results are represented by dashes.

Method	KTH-TIPS2-b	FMD	DTD	UMD	UIUC	GTOS
EfficientNet-B5	$87.0_{\pm5.9}$	$87.4_{\pm0.6}$	$77.6_{\pm0.4}$	$99.9_{\pm0.1}$	$98.4_{\pm0.4}$	$78.7_{\pm2.0}$
ConvNeXt-T	$87.9_{\pm5.3}$	$88.4_{\pm0.4}$	$76.3_{\pm0.7}$	$99.8_{\pm0.1}$	$98.8_{\pm0.3}$	$78.1_{\pm2.1}$
RegNetY 6.4G	$78.7_{\pm2.5}$	$78.9_{\pm0.8}$	$66.3_{\pm0.6}$	$94.0_{\pm0.7}$	$93.4_{\pm0.6}$	$70.1_{\pm1.8}$
FV-VGGVD [14]	$81.8_{\pm2.4}$	$79.8_{\pm1.8}$	$72.3_{\pm1.0}$	$99.9_{\pm0.1}$	$99.9_{\pm0.1}$	–
SIFT-FV [14]	$81.5_{\pm2.0}$	$82.2_{\pm1.4}$	$75.5_{\pm0.8}$	$99.9_{\pm0.1}$	$99.9_{\pm0.1}$	–
LFV [31]	$82.6_{\pm2.6}$	$82.1_{\pm1.9}$	$73.8_{\pm1.0}$	–	–	–
DeepTEN [35]	$82.0_{\pm3.3}$	$80.2_{\pm0.9}$	–	–	–	$84.3_{\pm1.9}$
Xception + SIFT-FV [32]	–	$86.1_{\pm1.6}$	$75.4_{\pm1.0}$	–	–	–
DSRNet [54]	$85.9_{\pm1.3}$	$86.0_{\pm0.8}$	$77.6_{\pm0.6}$	–	–	$85.3_{\pm2.0}$
VisGraphNet [40]	–	77.3	–	98.1	97.6	–
Non-Add Entropy [41]	84.4	77.7	–	98.8	98.5	–
Residual Pooling [11]	–	85.7	76.6	–	–	
FENet [37]	$88.2_{\pm0.2}$	$86.7_{\pm0.1}$	$74.2_{\pm0.1}$	–	–	$85.7_{\pm0.1}$
CLASSNet [10]	$87.7_{\pm1.3}$	$86.2_{\pm0.9}$	$74.0_{\pm0.5}$	–	–	$85.6_{\pm2.2}$
DFAEN [12]	86.6	87.6	76.1	–	–	–
RADAM [38]	$90.7_{\pm4.0}$	$88.7_{\pm0.4}$	$77.0_{\pm0.7}$	–	–	$84.2_{\pm1.7}$
Capsule [55]	71.8	80.7	71.0	–	99.3	
Ours(EfficientNet)-FV	$93.4_{\pm3.6}$	$91.4_{\pm1.0}$	$83.1_{\pm0.3}$	$99.9_{\pm0.1}$	$100_{\pm0.0}$	$85.9_{\pm1.1}$
Ours(EfficientNet)-FV+FC	$92.9_{\pm3.7}$	$93.9_{\pm0.2}$	$83.6_{\pm0.4}$	$100_{\pm0.0}$	$100_{\pm0.0}$	$85.8_{\pm1.4}$
Ours(ConvNeXt)-FV	$87.8_{\pm4.1}$	$89.4_{\pm0.6}$	$82.7_{\pm0.4}$	$100_{\pm0.0}$	$99.8_{\pm0.1}$	$86.1_{\pm0.6}$
Ours(ConvNeXt)-FV+FC	$88.4_{\pm4.0}$	$90.6_{\pm0.4}$	$82.6_{\pm0.3}$	$100_{\pm0.0}$	$99.8_{\pm0.1}$	$85.6_{\pm0.7}$
Ours(RegNetY)-FV	$86.8_{\pm5.3}$	$88.9_{\pm0.6}$	$79.7_{\pm0.4}$	$100_{\pm0.0}$	$99.8_{\pm0.1}$	$83.0_{\pm1.4}$
Ours(RegNetY)-FV+FC	$86.9_{\pm5.2}$	$89.1_{\pm0.5}$	$79.7_{\pm0.4}$	$100_{\pm0.0}$	$99.8_{\pm0.1}$	$83.1_{\pm1.3}$

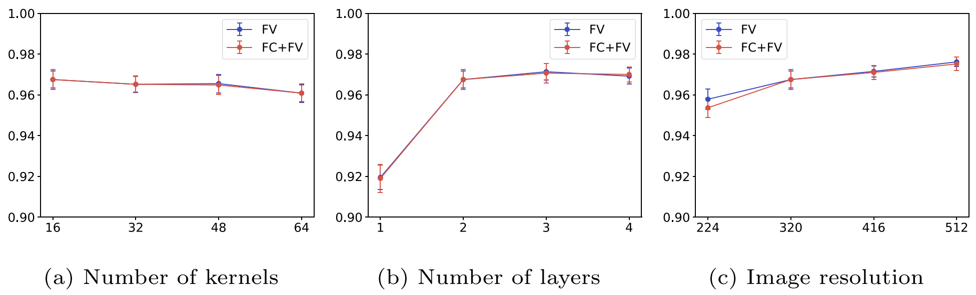


Fig. 10. Accuracy of our model for different values of parameters in 1200Tex database.

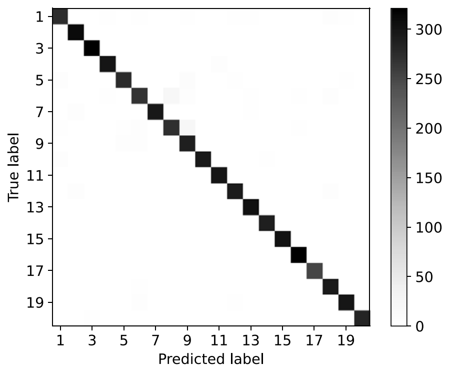
state-of-the-art. It is important to reinforce here that the larger confidence intervals in KTH-TIPS2-b is caused by the official train/test split. It forces the model to work on 4 significantly different tasks, whose accuracies are averaged out at the end to provide the final classification score. Methods such as FENet and others do not use the official split (e.g., FENet employs 10-fold) and this explains their tighter confidence intervals. Other methods such as FV-VGGVD and LFV use the official split, however train/test ratio employed is 1:3 while we use 3:1 as done in RADAM.

Finally, we apply our model to the classification task of Brazilian plant species based on the scanned leaf texture (1200Tex database [18]). We first evaluate the impact of parameter change on the database. In Fig. 10(a), we show that changing the number of kernels does not significantly affect accuracy. The same observation made for benchmark databases applies in this case, i.e. 16 Gaussian distributions are sufficient to model the underlying probability distribution of local features. In Fig. 10(b), we show that adding information from mid-level layers has a significant impact on accuracy. We can also see, in Fig. 10(c), that increasing the number of local features by changing image size affects accuracy positively as in all other databases tested.

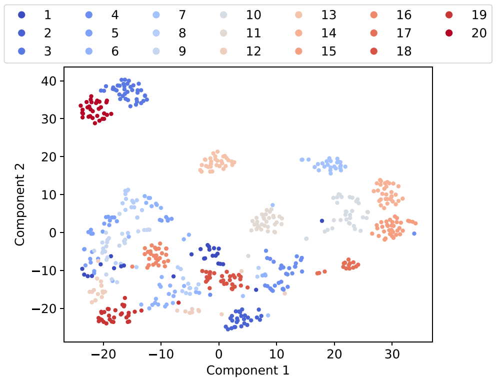
For the particular task of evaluating and comparing the behavior of our model in 1200Tex database, we use the same parameters we used for benchmark databases. We generate the confusion matrix for FC+FV feature vector using PCA as dimensionality reduction method and EfficientNet as backbone. It is presented in Fig. 11. We note that there is not much confusion when classifying the plant species. The classes that our model has most problems classifying are 8, wrongly labeling around 11.7% of samples, and 6, where 12.4% of samples are confused with other classes. Class 8, which presents a green leaf mostly dotted with few veins, is confused with classes 6, which is also veined, and 18, which is dotted and veined. Confusion in this case can be generated when examples from 8 have more veined areas than dotted, being wrongly labeled according to the proportions between those areas. Class 6 is mostly confused with class 8, which could be due to image or leaf imperfection in some examples from class 6, which are interpreted as dotted regions. In t-SNE visualization, we see all mentioned classes overlapping each other. We also see classes 14 and 17 as distinguishable from others using only the two main components.

In Table 5 we list the accuracy of the best previous results on 1200Tex database published in the literature, in comparison with our proposal. Here, the usage of our methodology made a huge difference in accuracy, scoring an accuracy from 5% to 9% better than the second best method (Non-Add Entropy), depending on the backbone used. Backbones are explicitly indicated in parentheses. In fact, up to our knowledge, our method outperforms state-of-the-art accuracy on this database.

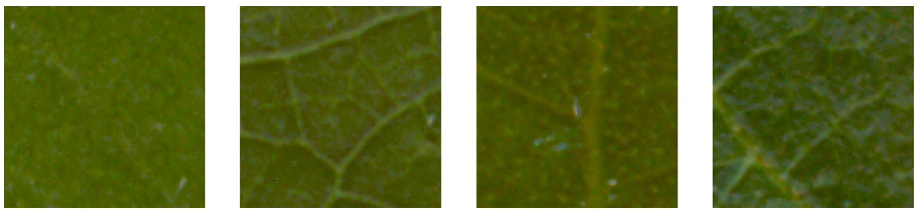
In terms of computational complexity, our model depends on the complexity of the CNN backbone, PCA, GMM, Fisher Vector encoding and SVM. The CNN backbone is not learned, no time is associated with training it, so we perform our analysis on the learned parts. The



(a) Confusion Matrix



(b) t-SNE visualization



(c) Class examples

Fig. 11. On top: confusion matrix for 1200Tex and t-SNE visualization of the worst train/test round in terms of accuracy. On bottom: from left to right, examples of classes 5, 6, 8, and 18.

Table 5
Comparison of accuracy in 1200Tex database with other methods in the literature. All results were obtained directly from the literature. When results were not found in the original paper, additional reference was given to where the result was taken from.

Method	Accuracy (%)
SIFT+BOVW [14]	86.0 [56]
FV-VGGVD [14]	87.1 [41]
Fractal [56]	86.3
VisGraphNet [40]	87.4
Non-Add Entropy [41]	88.5
BoFF [42]	87.2
Ours(EfficientNet)-FV	$97.4_{\pm 0.4}$
Ours(EfficientNet)-FV+FC	$97.2_{\pm 0.5}$
Ours(ConvNeXt)-FV	$96.4_{\pm 0.6}$
Ours(ConvNeXt)-FV+FC	$96.4_{\pm 0.4}$
Ours(RegNetY)-FV	$93.8_{\pm 0.5}$
Ours(RegNetY)-FV+FC	$93.8_{\pm 0.5}$

number of local features T increases quadratically as we increase the number of layers l and image width w. Each function $f_{n,m}$ learned through PCA using randomized SVD depends linearly on T_n and D_n and logarithmically on D_m [49]. As $D_m \leq D_n$, we estimate PCA complexity to be $\mathcal{O}(l^2 w^2 d \log d)$ where $d = D_n$. The GMM algorithm depends linearly on T and d and quadratically on the number of kernels K. The SVM depends quadratically on the number of components of FV, which depends linearly on k and d. Thus, we estimate the complexity of our method as $\mathcal{O}(l^2 w^2 k^2 d^2)$.

When no training is performed, PCA, GMM and SVM depend linearly on the parameters T, k and d. As T depends quadratically on l and w, our model has estimated complexity of $\mathcal{O}(l^2 w^2 kd)$. In previous paragraphs, we show how the model behaves well using 2 layers and 16 kernels in all tested datasets. This configuration should work for most applications. Thus, the main impact of using our model for evaluation purposes is image resolution and the choice of the CNN backbone.

In this section, we have shown that, for both benchmark databases and for the proposed application, extracting local features from earlier convolutional layers adds crucial information for texture classification. In fact, both FC and FC+FV accuracy was improved with the addition of one layer to the method proposed in [14] in most databases. When comparing our approach to more recent methods that also use information from earlier layers [11,12,37], the performance improvements may indicate that Fisher Vector remains a suitable method for encoding local features in the domain of visual textures.

Additionally, we verified that increasing the input image resolution has positive impact on accuracy. Such impact can be explained by the fact that changing the image resolution affects the number of local features and how representative a local feature is of a given area. Although expected, no significant impact was seen in changing the number of kernels in GMM, indicating that 16 kernels is enough to model the underlying distribution of local features. Furthermore, adding noise to images had small impact on accuracy in all tested databases, what shows robustness of our model.

7. Conclusions

In this work, we proposed and investigated the use of local features extracted from multiple convolutional layers and how this improves texture classification using Fisher Vector. More precisely, we computed the Fisher Vector on local features extracted from one to four convolutional layers and used them as texture descriptors.

We evaluated the performance of our method in visual texture classification, both in benchmark databases and in a practical problem of identifying plant species. In both situations, our method presented a significant improvement over other methods in the literature and reached competitive accuracy with the state-of-the-art.

One limiting factor of our proposed approach is the use of Gaussian Mixture Model algorithm for calculating Fisher Vectors. This algorithm is known to be dependent on the availability of memory, given that learning stage requires providing the full set of training local features. This fact was noticeable in the case of GTOS database, where it was necessary undersampling the training set. More efficient alternatives are intended to be investigated in future works. One possible approach is modifying the model for batch learning. This approach can be achieved using incremental GMM and incremental SVM algorithms based on works such as [57,58].

Overall, our model enhances accuracy of CNNs in visual texture classification. The method is suitable for integration with very deep CNN architectures. It is also appropriate for applications with limited availability of labeled data for training as, in this case, besides the high classification accuracy, the computational burden is also reduced. An example of such application is image-based medical diagnostic. In fact, data scarcity is a pretty common scenario in texture analysis, either in benchmark datasets or in real-world applications.

Code availability

The code developed in this study is publicly available at https://github.com/lolyra/multilayer.

CRediT authorship contribution statement

Lucas O. Lyra: Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Antonio E. Fabris:** Conceptualization, Formal analysis, Investigation, Project administration, Supervision, Writing – review & editing. **Joao B. Florindo:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Joao Florindo reports equipment, drugs, or supplies was provided by State of Sao Paulo Research Foundation. Joao Florindo reports financial support was provided by National Council for Scientific and Technological Development. Lucas Lyra reports financial support was provided by Coordination of Higher Education Personnel Improvement.

Data availability

Data will be made available on request.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance Code 001 (Grant #1796018). J. B. F. received financial support from São Paulo Research Foundation (FAPESP), Brazil (Grant #2020/01984-8) and from National Council for Scientific and Technological Development, Brazil (CNPq) (Grants #306030/2019-5 and #423292/2018-8).

References

[1] R.A. Ansari, K.M. Buddhiraju, R. Malhotra, Urban change detection analysis utilizing multiresolution texture features from polarimetric SAR images, Remote Sens. Appl.: Soc. Environ. 20 (2020) 100418.
[2] K. Nurzynska, S. Iwaszenko, Application of texture features and machine learning methods to grain segmentation in rock material images, Image Anal. Stereol. 39 (2) (2020) 73–90.
[3] E. Scalco, G. Rizzo, Texture analysis of medical images for radiotherapy applications, Br. J. Radiol. 90 (1070) (2017) 20160642.
[4] S. Jana, S. Basak, R. Parekh, Automatic fruit recognition from natural images using color and texture features, in: 2017 Devices for Integrated Circuit (DevIC), IEEE, 2017, pp. 620–624.
[5] S. Candemir, X.V. Nguyen, L.R. Folio, L.M. Prevedello, Training strategies for radiology deep learning models in data-limited scenarios, Radiol.: Artif. Intell. 3 (6) (2021) e210014.
[6] S. Montaha, S. Azam, A. Rafid, M.Z. Hasan, A. Karim, K.M. Hasib, S.K. Patel, M. Jonkman, Z.I. Mannan, MNet-10: A robust shallow convolutional neural network model performing ablation study on medical images assessing the effectiveness of applying optimal data augmentation technique, Front. Med. 9 (2022) 924979.
[7] D. Gibert, C. Mateu, J. Planes, R. Vicens, Classification of malware by using structural entropy on convolutional neural networks, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, 2018, pp. 7759–7764.
[8] W. Wan, J. Chen, T. Li, Y. Huang, J. Tian, C. Yu, Y. Xue, Information entropy based feature pooling for convolutional neural networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3405–3414.
[9] P. Liu, H. Zhang, W. Lian, W. Zuo, Multi-level wavelet convolutional neural networks, IEEE Access 7 (2019) 74973–74985.
[10] Z. Chen, F. Li, Y. Quan, Y. Xu, H. Ji, Deep texture recognition via exploiting cross-layer statistical self-similarity, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 5231–5240.
[11] S. Mao, D. Rajan, L.T. Chia, Deep residual pooling network for texture recognition, Pattern Recognit. 112 (2021) 107817.
[12] Z. Yang, S. Lai, X. Hong, Y. Shi, Y. Cheng, C. Qing, DFAEN: Double-order knowledge fusion and attentional encoding network for texture recognition, Expert Syst. Appl. 209 (2022) 118223.
[13] M. Jogin, M. Madhulika, G. Divya, R. Meghana, S. Apoorva, et al., Feature extraction using convolution neural networks (CNN) and deep learning, in: 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology, RTEICT, IEEE, 2018, pp. 2319–2323.
[14] M. Cimpoi, S. Maji, I. Kokkinos, A. Vedaldi, Deep filter banks for texture recognition, description, and segmentation, Int. J. Comput. Vis. 118 (1) (2016) 65–94.
[15] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 6105–6114.
[16] L. Sharan, R. Rosenholtz, E.H. Adelson, Accuracy and speed of material categorization in real-world images, J. Vis. 14 (9) (2014) 12, http://dx.doi.org/10.1167/14.9.12.
[17] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, A. Vedaldi, Describing textures in the wild, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3606–3613.
[18] D. Casanova, J.J. de Mesquita Sá Junior, O.M. Bruno, Plant leaf identification using gabor wavelets, Int. J. Imaging Syst. Technol. 19 (3) (2009) 236–243, http://dx.doi.org/10.1002/ima.20201.
[19] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[20] T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 971–987.
[21] A. Hafiane, K. Palaniappan, G. Seetharaman, Joint adaptive median binary patterns for texture classification, Pattern Recognit. 48 (8) (2015) 2609–2620.
[22] Y. Ruichek, et al., Local concave-and-convex micro-structure patterns for texture classification, Pattern Recognit. 76 (2018) 303–322.
[23] J. Malik, S. Belongie, T. Leung, J. Shi, Contour and texture analysis for image segmentation, Int. J. Comput. Vis. 43 (1) (2001) 7–27.
[24] S. Lazebnik, C. Schmid, J. Ponce, A sparse texture representation using local affine regions, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1265–1278, http://dx.doi.org/10.1109/TPAMI.2005.151.
[25] J. Zhang, M. Marszałek, S. Lazebnik, C. Schmid, Local features and kernels for classification of texture and object categories: A comprehensive study, Int. J. Comput. Vis. 73 (2) (2007) 213–238.
[26] L. Sharan, C. Liu, R. Rosenholtz, E.H. Adelson, Recognizing materials using perceptually inspired features, Int. J. Comput. Vis. 103 (3) (2013) 348–371.
[27] G. Amato, P. Bolettieri, F. Falchi, C. Gennaro, Large scale image retrieval using vector of locally aggregated descriptors, in: International Conference on Similarity Search and Applications, Springer, 2013, pp. 245–256.
[28] F. Perronnin, C. Dance, Fisher kernels on visual vocabularies for image categorization, in: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–8.
[29] F. Perronnin, J. Sánchez, T. Mensink, Improving the fisher kernel for large-scale image classification, in: European Conference on Computer Vision, Springer, 2010, pp. 143–156.
[30] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
[31] Y. Song, F. Zhang, Q. Li, H. Huang, L.J. O'Donnell, W. Cai, Locally-transferred fisher vectors for texture classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4912–4920.
[32] M. Jbene, A.D. El Maliani, M. El Hassouni, Fusion of convolutional neural network and statistical features for texture classification, in: 2019 International Conference on Wireless Networks and Mobile Communications, WINCOM, IEEE, 2019, pp. 1–4.
[33] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1251–1258.
[34] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[35] H. Zhang, J. Xue, K. Dana, Deep ten: Texture encoding network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 708–717.

[36] J. Xue, H. Zhang, K. Dana, Deep texture manifold for ground terrain recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 558–567.

[37] Y. Xu, F. Li, Z. Chen, J. Liang, Y. Quan, Encoding spatial distribution of convolutional features for texture representation, Adv. Neural Inf. Process. Syst. 34 (2021).

[38] L. Scabini, K. Zielinski, L. Ribas, W. Gonçalves, B. De Baets, O. Bruno, RADAM: Texture recognition through randomized aggregated encoding of deep activation maps, 2023, arXiv preprint arXiv:2303.04554.

[39] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, S. Xie, A convnet for the 2020s, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 11976–11986.

[40] J.B. Florindo, Y.-S. Lee, K. Jun, G. Jeon, M.K. Albertini, VisGraphNet: A complex network interpretation of convolutional neural features, Inform. Sci. 543 (2021) 296–308.

[41] J. Florindo, K. Metze, Using non-additive entropy to enhance convolutional neural features for texture recognition, Entropy 23 (2021) 1259, http://dx.doi.org/10.3390/e23101259.

[42] J.B. Florindo, E.E. Laureano, BoFF: A bag of fuzzy deep features for texture recognition, Expert Syst. Appl. 219 (2023) 119627.

[43] T. Leung, J. Malik, Representing and recognizing the visual appearance of materials using three-dimensional textons, Int. J. Comput. Vis. 43 (1) (2001) 29–44.

[44] T. Jaakkola, D. Haussler, Exploiting generative models in discriminative classifiers, Adv. Neural Inf. Process. Syst. 11 (1998).

[45] J. Sánchez, F. Perronnin, T. Mensink, J. Verbeek, Image classification with the fisher vector: Theory and practice, Int. J. Comput. Vis. 105 (3) (2013) 222–245.

[46] B. Caputo, E. Hayman, P. Mallikarjuna, Class-specific material categorisation, in: Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Vol. 2, 2005, pp. 1597–1604, http://dx.doi.org/10.1109/ICCV.2005.54.

[47] Y. Xu, H. Ji, C. Fermüller, Viewpoint invariant texture description using fractal analysis, Int. J. Comput. Vis. 83 (1) (2009) 85–100, http://dx.doi.org/10.1007/s11263-009-0220-6.

[48] J. Xue, H. Zhang, K. Dana, K. Nishino, Differential angular imaging for material recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 764–773.

[49] N. Halko, P.-G. Martinsson, J.A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.

[50] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[51] Ç. Arı, S. Aksoy, O. Arıkan, Maximum likelihood estimation of Gaussian mixture models using stochastic search, Pattern Recognit. 45 (7) (2012) 2804–2816.

[52] S. Balakrishnan, M.J. Wainwright, B. Yu, Statistical guarantees for the EM algorithm: From population to sample-based analysis, Ann. Statist. 45 (1) (2017) 77–120, http://dx.doi.org/10.1214/16-AOS1435.

[53] I. Radosavovic, R.P. Kosaraju, R. Girshick, K. He, P. Dollár, Designing network design spaces, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10428–10436.

[54] W. Zhai, Y. Cao, Z.-J. Zha, H. Xie, F. Wu, Deep structure-revealed network for texture recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11010–11019.

[55] B. Mamidibathula, S. Amirneni, S.S. Sistla, N. Patnam, Texture classification using capsule networks, in: Pattern Recognition and Image Analysis: 9th Iberian Conference, IbPRIA 2019, Madrid, Spain, July 1–4, 2019, Proceedings, Part I 9, Springer, 2019, pp. 589–599.

[56] P.M. Silva, J.B. Florindo, Fractal measures of image local features: an application to texture recognition, Multimedia Tools Appl. 80 (9) (2021) 14213–14229.

[57] R.C. Pinto, P.M. Engel, A fast incremental gaussian mixture model, PLoS One 10 (10) (2015) e0139931.

[58] P. Laskov, C. Gehl, S. Krüger, K.-R. Müller, K.P. Bennett, E. Parrado-Hernández, Incremental support vector learning: Analysis, implementation and applications, J. Mach. Learn. Res. 7 (9) (2006).