

Multilayer deep feature extraction for visual texture recognition

Lucas O. Lyra^{a,*}, Antonio Elias Fabris^a, Joao B. Florindo^b

^a*Institute of Mathematics and Statistics - University of São Paulo
Rua do Matão, 1010 - Butantã, CEP 05508-090, São Paulo, SP, Brasil*

^b*Institute of Mathematics, Statistics and Scientific Computing - University of Campinas
Rua Sérgio Buarque de Holanda, 651 - Barão Geraldo, CEP 13083-859, Campinas, SP,
Brasil*

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 paper is focused on improving the accuracy of convolutional neural networks in texture classification. This is done by extracting features from multiple convolutional layers of a pretrained neural network and aggregating such features using Fisher vector. The reason for using features from earlier convolutional layers is obtaining information that is less domain specific. We verify the effectiveness of our method on texture classification of benchmark datasets, as well as on a practical task of Brazilian plant species identification. In both scenarios, Fisher vectors calculated on multiple layers outperform state-of-art methods, confirming that early convolutional layers provide important information about the texture image for classification.

Keywords: Texture recognition, Convolutional neural networks, Fisher vector, Image descriptors.

*Corresponding author

Email addresses: lucas.oliveira.lyra@alumni.usp.br (Lucas O. Lyra), fabris@usp.br (Antonio Elias Fabris), jbflorindo@ime.unicamp.br (Joao B. Florindo)

1. Introduction

Texture is one of the most important image attributes in computational vision. It provides information on the spatial arrangement of the pixel intensities in an image. Textures can be useful for recognition of material properties, specially when other image attributes, like shape, are not useful. They play an important role in remote sensing [3], material science [26], medicine [32], agriculture [17] and many other fields.

The problem of recognizing a texture can be divided into two tasks: the first is extracting features from an image; the second one is training a classifier for feature recognition. Given the way those tasks are performed, techniques can be divided into two groups. In the first case, features are extracted by a computer vision method and used by a classical machine learning algorithm for classification. In the other one, both feature extraction and classification are performed by a deep neural network, usually Convolutional Neural Network (CNN). In that case, parameters are learned by an optimization algorithm.

Although CNNs have been quite successful for image classification, textures are still challenging. This is a consequence of the limited availability of data for training in areas of application, such as medicine, for example, and other characteristics such as the high inter-class similarity and the lack of a global viewpoint over the analyzed object. Even if we consider transferring knowledge from large databases, like 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 [11, 41, 22].

When using classical texture features, the performance classification of algorithms rely heavily on how well the extracted features describe the image. Features extracted from the last convolutional layer of CNNs tend to be better than features extracted by classical filter banks [8]. However, given the domain shift mentioned in the previous paragraph, features from the last convolutional layer might be too specific to the training database.

In this context, we propose a method that combines generalist local features with specific ones into a single set of features. In order to evaluate such method, we compute Fisher Vectors on this set of features and classify them using Support Vector Machine (SVM). The major contributions of this paper are the following:

1. Up to our knowledge, this is the first time Fisher Vectors are associated with features extracted from multiple layers of a CNN;
2. We propose the application of normalization on descriptors extracted from fully-connected layers and evaluate the impact of the proposed normalization in classification accuracy;
3. We obtain results competitive with other methods available in literature, establishing, up to our knowledge, new state-of-the-art performance in Flickr Material Database (FMD) [34], Describable Textures Dataset (DTD) [7] and in Brazilian plant species 1200Tex database [5].

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. The code will be available at <https://github.com/lolyra/multilayer>.

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) [23], Local Binary Patterns (LBP) [27] and variants [12, 30] 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 [24, 19, 44, 33], Vector of Locally Aggregated Descriptors (VLAD) [2] and Fisher Vectors (FV) [28, 29].

However, in recent works, 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 pretrained CNNs on large databases, like ImageNet. This is the case of Cimpoi et al. [8]. They proposed a method combining CNNs with traditional encoders that achieved state-of-the-art results. However, its good performance requires the use of multiple scales in the input image, which implies using CNNs several times.

More recently, improvements on the association of CNNs with traditional encoders have been proposed. Song et al. [37] proposed a method that consists of optimizing the Fisher Vector for classification by applying a simple neural network on top of the FV descriptor and training it from scratch. Lin et. al. [21] avoid the use of generative models such as GMM, applying outer product to features extracted from two neural networks, thus obtaining second-order image features that are used to calculate FV or VLAD.

Given the overall good performance of SIFT even when compared to CNNs, a step towards a hybrid model was taken by Jbene et al. [18]. They calculate Fisher Vectors on features extracted by CNN and on features extracted by SIFT, later combining for classification purposes.

In addition, another source of features that can contribute to a good performance for classification are those extracted from other layers of a CNN rather than only the last convolutional layer. Such approach is presented by Chen et al. [6] where encoding is performed by calculating statistical self-similarity using a soft histogram of local differential box-counting dimensions of cross-layer features.

More recent works have focused on alternatives to traditional encoders, like

FV, either to create an end-to-end trainable model or reduce computational costs. Florindo et al. [9] performs aggregation by joining two different fully-connected layers output. The first one calculated over original image and the other one calculated over an entropy measure of the original image. In other work [10], encoding is performed by visibility graphs.

In the scope of end-to-end training, Mao et al. [25] obtain a fully trainable model by performing encoding with an aggregation module, which consists of convolutions and average pooling. In Xu et al. [43], encoding is performed by a local-global hierarchical fractal analysis.

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 Convolutional Neural Networks (CNN), detailing some frequently used layers. In Section 3.2, we present a concise summary of Fisher Vector (FV).

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.

Classical extraction of features is performed by applying convolutional filters to the input image [20]. In this context, 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.

3.1.1. Convolution Layer

A convolutional layer is an appropriate mechanism to reduce the number of parameters and explore relationships between neighbor pixels. It is composed by multiple 2-dimensional kernels $K_c(i, j) = w_{i,j}$. Given a 2-dimensional input $I(i, j)$, each output $S_c(i, j)$ is the convolution of I by K_c , which is given by:

$$S_c(i, j) = \sum_{m=0}^{d-1} \sum_{n=0}^{d-1} I(i \cdot s - m, j \cdot s - n) K_c(m, n). \quad (1)$$

The parameter d is called *kernel size* and s is the *stride*. Note that, generally, kernels have width equals height. Stride is the convolution step size.

Let W and H be, respectively, the width and height of I . As i, j are not defined outside the set $[0, W - 1] \times [0, H - 1]$, S has dimensions smaller than I . In order to increase output size, a parameter p , called *padding*, can be introduced. In such a case, we define $I(i, j) = 0$ for $i \in [-p, 0] \cup [W, W + p]$ and $j \in [-p, 0] \cup [H, H + p]$. Thus, the output dimension D_{out} is given by

$$D_{out} = \frac{D_{in} - d + 2p}{2} - 1, \quad (2)$$

where D_{in} can be either W or H . An activation function $f : \mathbb{R} \rightarrow \mathbb{R}$ is usually applied to S in order to introduce non-linearity to the objective function estimator. One of the most popular activation function is called *rectifier linear unit* and is given by

$$f(x) = \max(0, x). \quad (3)$$

3.1.2. Pooling Layer

A pooling layer is used to reduce the number of parameters to be learned and the computational cost of the network. This is performed by reducing the input dimensions and helps preventing overfitting. In this layer, the input I is reduced by merging a set of $m \times n$ pixels into a single one. In general, pixels are combined by retrieving their maximum value. Thus the output S is given by

$$S(i, j) = \max_{0 \leq k \leq m} (\max_{0 \leq l \leq n} (I(i \cdot m + k, j \cdot n + l))). \quad (4)$$

3.1.3. Dropout Layer

Dropout is a regularization technique used to avoid overfitting. It was introduced by Srivastava, et al. in [38] and consists of avoiding the update of randomly selected neurons during one epoch. In the introductory paper, it is shown that the use of this technique has increased the accuracy of supervised learning tasks in areas such as computer vision, voice recognition and computational biology.

3.1.4. Normalization Layer

In neural network training, updates to the weights in early layers can change data distribution significantly in later layers. This phenomenon is called *internal covariate shift* and can make the training process very slow. In order to avoid it, a normalization layer is introduced. Using this layer, input data distribution can be imposed to have mean 0 and variance 1. Normalization can not only speed up the training process, but also act as a regularization layer, dismissing the need of a Dropout layer.

As noted in [15], normalizing all data can be costly and a better approach would be batch normalization of the data. Thus let m be the batch size and d the dimension of the input. Let x_j^i denote the j -th coordinate of the i -th input data from a batch. The normalization is given by

$$\hat{x}_j^i = \frac{x_j^i - \mu_j}{\sqrt{\sigma_j + \epsilon}}, \quad (5)$$

where μ_j and σ_j denote, respectively, the mean and variance of the j -th components of the batch and ϵ is a positive constant to assure numerical stability.

3.1.5. Fully-Connected Layer

A fully-connected (FC) layer explores relations among all the components of the input data. In such layer, the multi-dimensional data from the previous one is rearranged into a one-dimensional vector V . The layer's output S is also a one-dimensional vector and is given by

$$s_j = \sum_{i=0}^{d-1} w_{i,j} v_i, \quad (6)$$

where s_j is the j -th component of S and v_i is the i -th component of V , d is the number of components of V and w 's are the weights of the layer.

The FC layer is generally placed on top of the network to accomplish the classification task. Softmax function is normally used as an activation function after the last layer. The objective function guiding the optimization of the network is called *loss function*. Some commonly used loss functions in classification task are *cross-entropy* and *Hinge loss*.

3.2. Fisher Vector

Let $X = \{x_d, d = 1 \cdots D\}$ denote a sample of D observations, $x_d \in \mathbb{R}^N$. 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

$$G_\lambda^X = \nabla_\lambda \log u_\lambda(X). \quad (7)$$

The gradient vector given by Equation (7) can be classified using any classification algorithm. In [16], the Fisher information matrix F_λ is suggested for this purpose:

$$F_\lambda = E_X[G_\lambda^X G_\lambda^{X'}]. \quad (8)$$

From this observation, a Fisher Kernel (FK) to measure similarity between two samples X and Y was proposed. Such kernel is defined by:

$$K_{FK}(X, Y) = G_\lambda^{X'} F_\lambda^{-1} G_\lambda^Y. \quad (9)$$

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:

$$K_{FK}(X, Y) = \mathcal{G}_\lambda^{X'} \mathcal{G}_\lambda^Y \quad (10)$$

where

$$\mathcal{G}_\lambda^X = L_\lambda \nabla_\lambda \log u_\lambda(X). \quad (11)$$

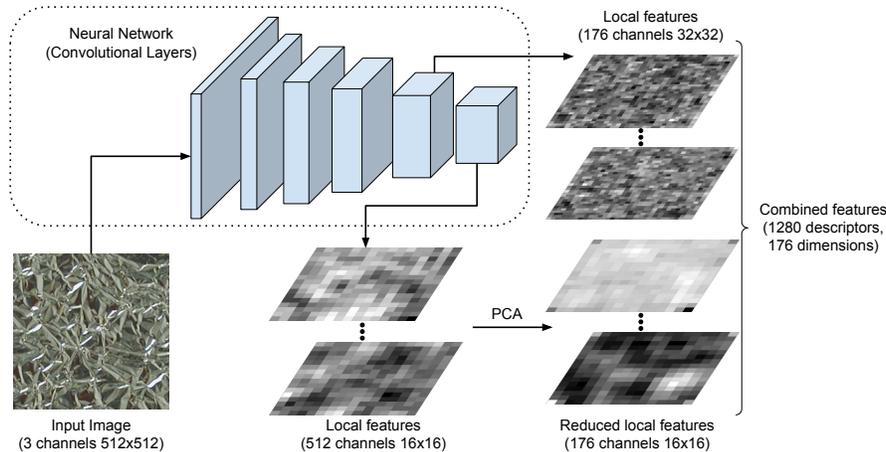


Figure 1: Feature extraction with the proposed method. From left to right we have the input texture, convolutional layers and local features extracted from the last two layers.

The vector \mathcal{G}_λ^X is called *Fisher Vector* (FV). We have that FV \mathcal{G}_λ^X and G_λ^X have the same dimensionality [31]. 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.

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 Subsection 4.1, we show the proposed strategy to build feature vectors. In Subsection 4.2, we show the classification process using such vectors.

4.1. Feature Extraction

In the first stage of our methodology, we are interested in using Fisher vectors to describe information extracted from multiple layers of a convolutional neural network. Initially, we take a CNN architecture pretrained on ImageNet

and use it as a feature extractor. We present the texture image as input to the pretrained CNN and collect the outputs of the last and penultimate convolutional layers. Both layers contain feature information about the image, the last layer presenting more high-level information than the previous one.

Definition 1. Let the set $X_n = \{x_t, t = 1 \cdots T_n \mid x_t \in \mathbb{R}^{D_n}\}$ denote the output of the n -th convolutional layer, where $T_n = W_n \times H_n$ is the resolution of each channel and D_n is the number of channels. We call $x \in X_n$ a local feature and X_n a set of local features extracted from the n -th convolutional layer.

Let N denote the number of convolutional layers in a CNN. Our method takes the local feature sets X_{N-1} and X_N . We are interested in creating a single set X of local features, but we usually have $T_{N-1} > T_N$ and $D_{N-1} \leq D_N$. Thus, in order to combine X_{N-1} and X_N , we apply Principal Component Analysis (PCA) [1] to each element $x \in X_N$, so that the element with reduced dimension x' is such that $x' \in \mathbb{R}^{D_{N-1}}$. We end up with a set $X = \{x_t, t = 1 \cdots T \mid x_t \in \mathbb{R}^D\}$, where $D = D_1$ and $T = T_1 + T_2$. The proposed schema for feature extraction is exemplified in Figure 1, where the neural network architecture used is EfficientNet-B5 [39].

Once we have a set of local features, we calculate the Fisher vector. In order to do so, we assume that the local features x_l are generated independently by the distribution u_λ . Thus Equation (11) becomes:

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

We choose u_λ to be a Gaussian Mixture Model (GMM) composed by K Gaussian distributions, that is,

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

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

Let $\gamma_t(i)$ denote the probability of an observation x_t to be generated by the Gaussian u_i :

$$\gamma_i(x_t) = \frac{w_i u_i(x_t)}{\sum_{j=1}^K w_j u_j(x_t)}. \quad (14)$$

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 Equation (11) as:

$$\mathcal{G}_{w_i^d}^X = \frac{1}{T\sqrt{w_i}} \sum_{t=1}^T (\gamma_i(x_t) - w_i), \quad (15)$$

$$\mathcal{G}_{\mu_i^d}^X = \frac{1}{T\sqrt{w_i}} \sum_{t=1}^T \gamma_i(x_t) \left(\frac{x_t^d - \mu_i^d}{\sigma_i^d} \right), \quad (16)$$

$$\mathcal{G}_{\sigma_i^d}^X = \frac{1}{T\sqrt{2w_i}} \sum_{t=1}^T \gamma_i(x_t) \left[\frac{(x_t^d - \mu_i^d)^2}{(\sigma_i^d)^2} - 1 \right]. \quad (17)$$

4.2. Classification

In our second stage, we extract information from fully-connected layers by removing the classification layer of the CNN. We are left with a feature vector that we call FC.

Before doing classification, we perform a transformation over both FV and FC features. Let \mathbf{x} be a feature vector and let x denote an element of \mathbf{x} . We apply power and L_2 normalization to \mathbf{x} , which can be written as:

$$x \leftarrow \text{sign}(x) \sqrt{|x|}, \quad (18)$$

$$x \leftarrow \frac{x}{\|\mathbf{x}\|}. \quad (19)$$

These transformations were proposed in [29] as a way to improve classification with Fisher vectors. We noticed that those transformations are also beneficial for classification in the case of FC. Once we have normalized feature vectors, we perform classification with Support Vector Machine (SVM), using a modified version of Bhattacharyya coefficient given in Definition 2 as kernel.

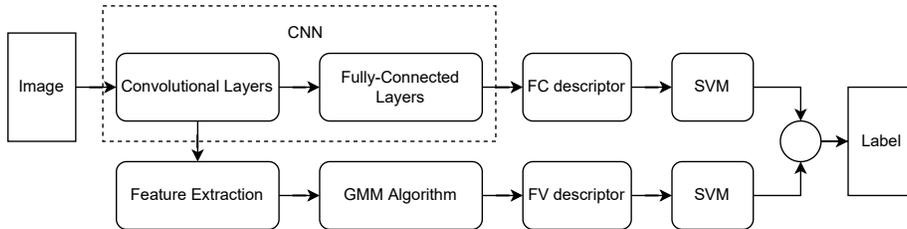


Figure 2: Summary of our proposed method for classifying a given image. Normalization is applied to FC and FV descriptors before classification with SVM.

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

$$K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^N \text{sign}(x_i y_i) \sqrt{|x_i y_i|}. \quad (20)$$

Note that the modified Bhattacharyya coefficient can be rewritten as

$$K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y}), \quad (21)$$

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

$$\phi(\mathbf{x})_i = \text{sign}(x_i) \sqrt{|x_i|}. \quad (22)$$

Thus, we apply the transformation given by Equation (22) to the normalized feature vectors and proceed to classification with a linear SVM.

Finally, we combine classification with SVM trained on FC and FV data by applying soft assignment. Let f_{FC} denote the decision function of SVM trained on FC and f_{FV} the decision function of SVM trained on FV. Given FC \mathbf{x} and FV \mathbf{y} calculated on the same sample, we assign a class c to the sample by:

$$c = \text{argmax}(f_{\text{FC}}(\mathbf{x}) + f_{\text{FV}}(\mathbf{y})). \quad (23)$$

The proposed method for classification is summarized in Figure 2.

5. Experiments

In this section we describe how we evaluate our proposed methodology. We start by evaluating the effects of hyperparameters on classification accuracy.

Our base model uses the EfficientNet-B5 architecture [39] with pre-trained ImageNet weights, input image resolution of 512×512 and 64 Gaussian distributions to model u_λ . Any hyperparameter change keeps the remaining parameters constant.

Fisher Vector accuracy can be affected by the number of Gaussian distributions that we use to model u_λ . Thus, using our base model, we tested the effect of this variation by reducing the number of Gaussian distributions. Another hyperparameter of our model is the input image resolution. We tested its effect on accuracy by downsampling the image in our base model. Also, we change the CNN architecture to see how our method behaves on other architectures.

Afterwards, we evaluated the effects of normalization of FC by comparing it with a model without normalization. Finally, we compare our base model with alternative state-of-art approaches. 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.

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

KTH-TIPS2-b [4], here referred to as KTH-TIPS, 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 of 200x200 size per material per sample. In each round, we use 1 sample for training and 3 samples for testing.

FMD [34] consists of 10 classes containing 100 images each. Each image has a size of 512x384. We run 10 training/testing rounds, each randomly selecting half of the database for training and using the other half for testing.

DTD [7] 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 [42] consists of 25 classes containing 40 images each. All images have a dimension of 1280x960. We evaluated our model 10 times in this dataset, each time randomly choosing 20 images from each class for training and the remainder for testing, following the same protocol as FMD.

UIUC [19], as UMD, consists of 1000 images evenly divided in 25 classes. Each image has resolution of 640x480. In order to evaluate our method in this dataset, we use the same protocol applied to FMD.

1200Tex [5] 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.

6. Results and Discussion

In this section we present the results obtained from the experiments described in Section 5. We show how they accomplished to verify the effectiveness of the proposed methodology in texture classification. 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*. In our tests, we call this hyperparameter *number of kernels*. We used 16, 32, 48 and 64 kernels in benchmark tests. The results are show in Figure 3. We observed very little variation of accuracy in the case of FMD and UMD, but a significant increase in accuracy as we increase the number of kernels for KTH-TIPS and DTD. Thus, using 64 kernels seems to be a good choice for all the databases tested. An improvement with increasing kernels is expected, as the greater the number of Gaussian distributions, the better it can model the underlying distribution that generates the local features. However, the number of Gaussian distributions should not be very large given the limited availability of data to train the GMM algorithm and computational costs.

The second hyperparameter of our model is the resolution of the input image. This resolution is directly proportional to the number of local features, which

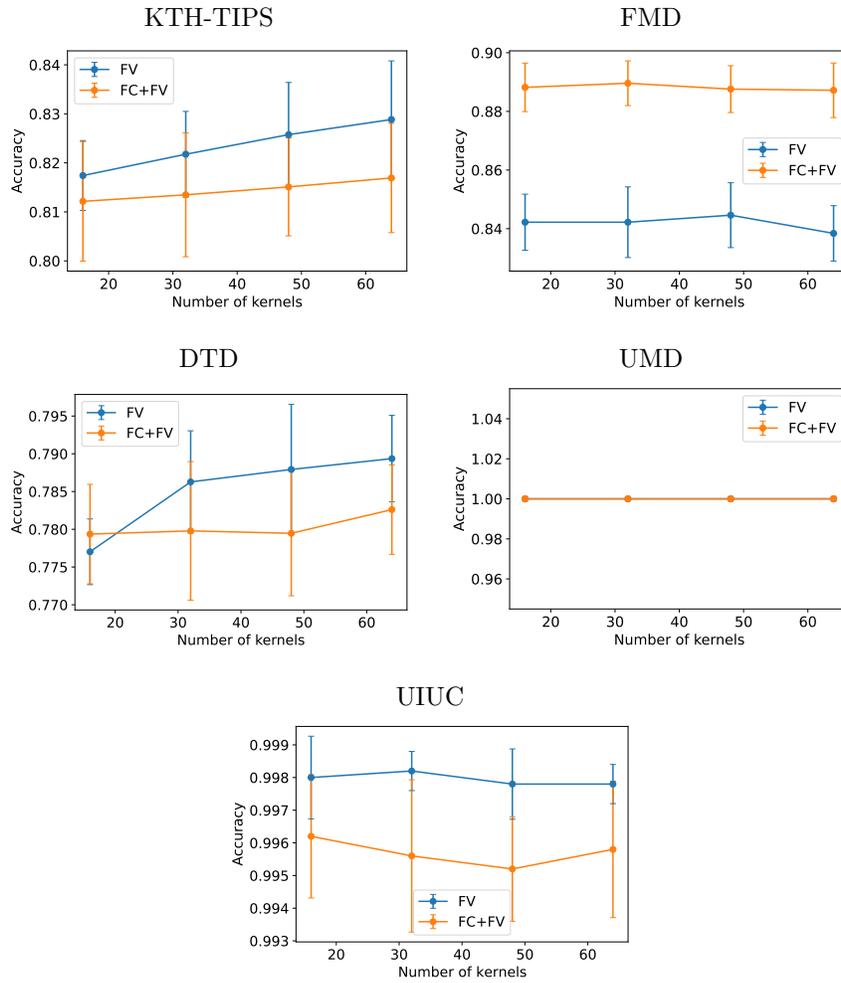


Figure 3: Variation of accuracy of our method according to the number of Gaussian distributions (kernels) used in GMM. Line colors describe which information was used for training the classifier. Error bars indicate the standard deviation of classification accuracy.

is linked to the performance of the GMM algorithm. We evaluated our model on the benchmark databases starting with 224×224 , the size used by the CNN architecture for training. We increase the resolution linearly up to 512×512 to show its effect on GMM algorithm. The results of this variation are shown in Figure 4. As expected, the increase in image resolution improved accuracy across all databases. This improvement is not only due to the number of local features, but also to how specific a local feature is. If the resolution is too small, information from small regions in an image may be lost. A condition for the use of generative models to be beneficial for accuracy is that local features must describe small regions rather than large ones.

In our third experiment, we show how our method behaves in different network architectures. We chose architectures that result in local features with similar dimensions. We tested our method in

- EfficientNet-B5, where local feature dimension $D = 176$;
- EfficientNetV2-s [40], where $D = 160$;
- ResNet34 [13], where $D = 256$.

As shown in Table 1, the best accuracy for KTH-TIPS2-b was achieved in EfficientNet-B5 while EfficientNetV2-s achieved better accuracy in FMD and DTD. Very deep ResNet, VGG [36], DenseNet [14] would increase greatly local feature dimensions and consequently the computational cost of GMM algorithm. For example, in DenseNet-161 local feature dimension is $D = 2048$.

Moreover, we verify the impact of the power and L_2 normalization applied to FC. All FC features used for evaluation are extracted from the architecture EfficientNet-B5. In Figure 5 we show the impact of the normalization of the distribution of FC elements for DTD database. For the other databases, the effect is similar. The normalization affects the format of the distribution and increases data sparsity. In Table 2 we show the effect of normalization on accuracy considering exclusively FC classification. In general, it helped increasing accuracy mean or reducing standard deviation. Most notorious result can be

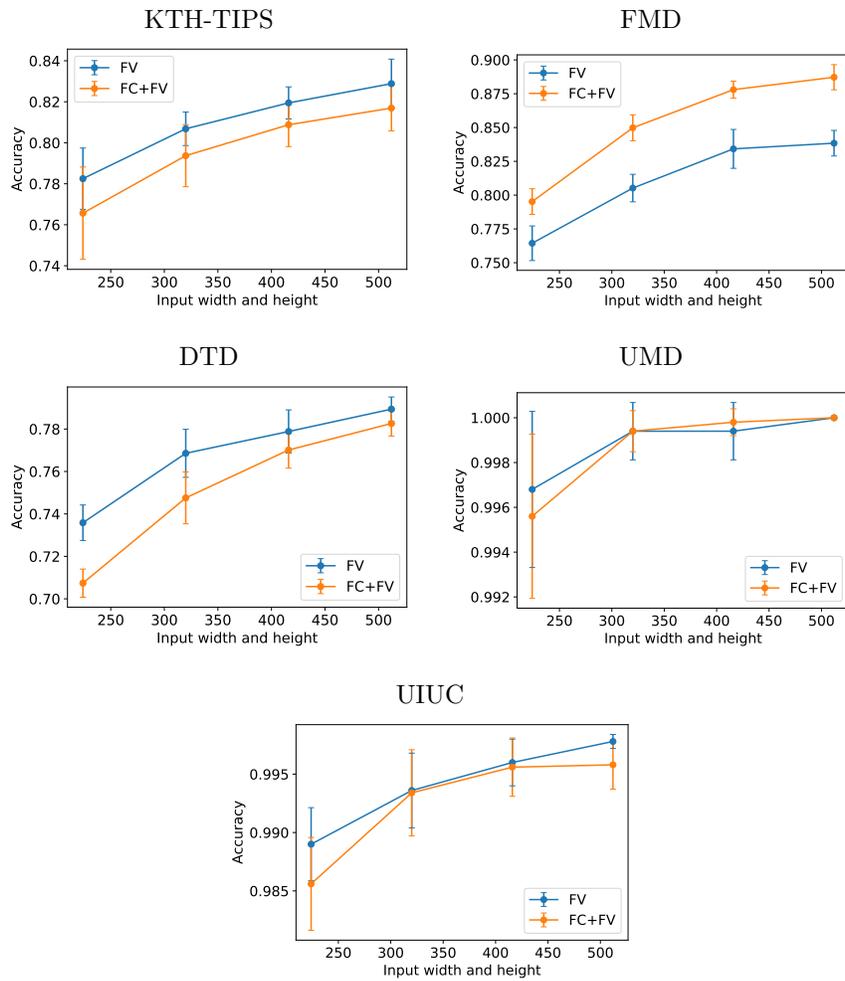


Figure 4: Accuracy of our method according to the resolution of the image used as input to the CNN. Line colors describe the feature vectors used for training the classifier. Error bars indicate the standard deviation of classification accuracy.

Table 1: Behavior of our proposed method in different CNN architectures. Column “Method” describes the feature vectors that were used to train the classifier.

Dataset	Method	EfficientNet-B5	EfficientNetV2-s	ResNet34
KTH-TIPS	FV	82.9 ± 1.2	80.6 ± 1.3	79.2 ± 1.7
	FV+FC	81.7 ± 1.1	79.5 ± 1.5	77.3 ± 3.7
FMD	FV	83.8 ± 0.9	85.7 ± 1.2	82.2 ± 1.4
	FV+FC	88.7 ± 0.9	88.9 ± 1.0	83.5 ± 1.2
DTD	FV	78.9 ± 0.6	79.3 ± 0.6	76.2 ± 0.3
	FV+FC	78.3 ± 0.6	77.8 ± 1.1	71.3 ± 0.7
UMD	FV	100 ± 0.0	100 ± 0.0	99.9 ± 0.1
	FV+FC	100 ± 0.0	100 ± 0.0	99.9 ± 0.1
UIUC	FV	99.8 ± 0.1	99.8 ± 0.1	99.8 ± 0.1
	FV+FC	99.6 ± 0.2	99.7 ± 0.2	99.7 ± 0.3

seen in DTD database, where both effects are present, while normalization had no impact in UMD.

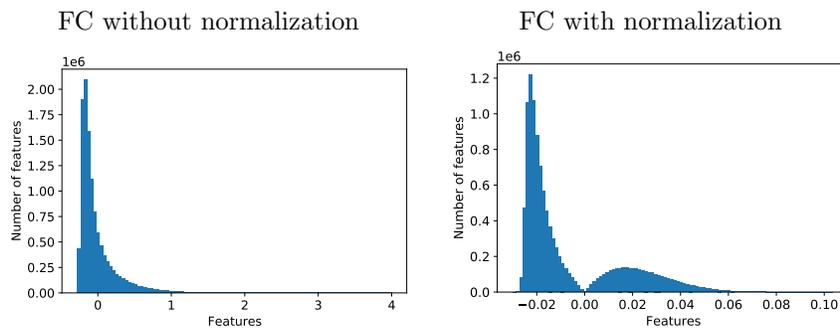


Figure 5: Histograms of FC calculated for DTD database before and after applying normalization given by Equations (18) and (19).

For all the following results, the architecture used is the EfficientNet-B5, the input image resolution is 512×512 and the number of kernels is 64. In Figure 6 we detail how our method behaves in the benchmark databases by showing how much confusion is presented in each database.

Table 2: Normalization impact on accuracy for benchmark databases. In this experiment, SVM was used to classify FC, in the first case with no transformation applied, in the second case, applying the normalization proposed in Section 4.2.

Database	Without Normalization	With Normalization
KTH-TIPS	78.8 ± 1.9	79.1 ± 1.9
FMD	86.6 ± 1.2	86.8 ± 0.9
DTD	72.9 ± 0.8	73.3 ± 0.7
UMD	100 ± 0.0	100 ± 0.0
UIUC	99.3 ± 0.3	99.2 ± 0.2

In KTH-TIPS, most noticeable problems are the classification of examples from class 5 (cotton) and class 11 (wool). In the case of cotton, its mostly confused with class 8 (linen), although there are certain confusion also with classes 3 (corduroy), 10 (wood) and 11. In the case of wool, its mostly confused with linen, but there is also confusion with classes 1 (aluminium foil) and 3. Interestingly, most part of the confusion is among textile textures, which are indeed challenging to classify, given that they can have a similar pattern.

In FMD, our model had most problems distinguishing class 5 (metal) from other classes, confusing it with classes 3 (glass), 7 (plastic), 8 (stone) and 10 (wood). The presence of confusion in this case could be explained by the fact that objects made out from these materials can present a similar shape or color to metallic objects.

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. These samples are mostly mistaken by classes 38 (stained) and 43 (veined). The confusion between blotchy and stained was expected, as images from both classes are very similar. Also, the edges between botched and non-blotched regions in an image can be mistakenly interpreted as veins, what could explain confusion with class 43. No significant confusion can be observed in UIUC and UMD databases.

In Table 3, we list the accuracy of several methods in the literature of texture

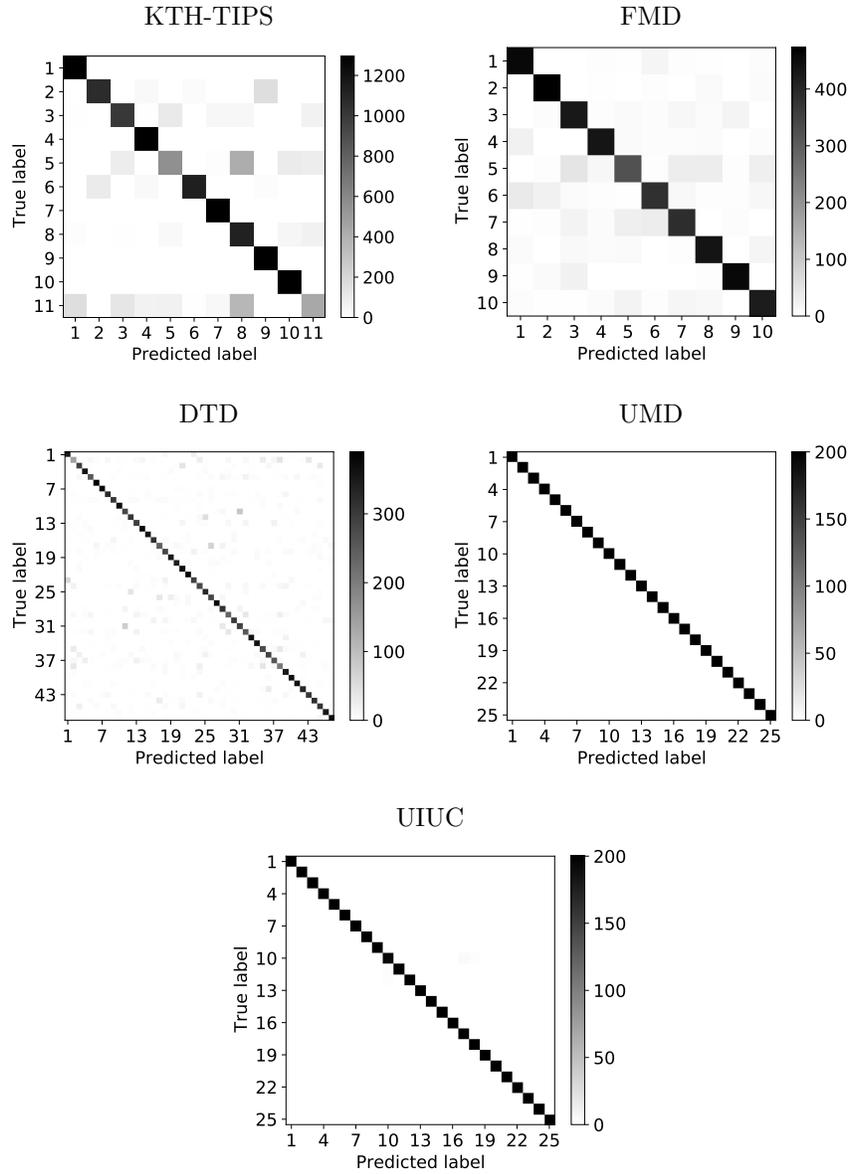


Figure 6: Confusion matrices computed with our classifier trained with FV, which encodes information from multiple convolutional layers. These matrices show the number of images that were assigned to certain class. No confusion means all images are labeled their true class, i.e., all diagonal elements are black and the remaining elements are white.

recognition compared with the proposed approach. Our proposed method using only FV outperforms other modern deep learning approaches in KTH-TIPS, DTD and UMD. The accuracy we achieved in DTD is, as far as we know, the best result available in the literature. Furthermore, our proposed method combining FC and FV is able to, to the best of our knowledge, achieve state-of-art performance in FMD database.

Table 3: Accuracy comparison with other methods in literature. Our proposed method is named here Multilayer-FV and Multilayer-FV+FC. All results shown are obtained directly from the original paper of each method. Non-published results are represented by dashes.

Method	KTH-TIPS	FMD	DTD	UMD	UIUC
FV-VGGVD [8]	81.8	79.8	72.3	99.9	99.9
SIFT-FV [8]	81.5	82.2	75.5	99.9	99.9
LFV [37]	82.6	82.1	73.8	-	-
VisGraphNet [10]	75.7	77.3	-	98.1	97.6
Non-Add Entropy [9]	-	77.7	-	98.8	98.5
Xception + SIFT-FV [18]	-	86.1	75.4	-	-
Residual Pooling [25]	-	85.7	76.6	-	-
FENet [43]	-	86.7	74.2	-	-
CLASSNet [6]	-	86.2	74.0	-	-
Multilayer-FV	82.9	83.8	78.9	100.0	99.8
Multilayer-FV+FC	81.7	88.7	78.3	100.0	99.6

Finally, we apply our model to the classification task of Brazilian plant species. We first evaluate the impact of hyperparameter change on the database. In Figure 7, we show that the increase in number of kernels affects negatively the accuracy of classification. This is probably caused by the low availability of data in order to train the GMM algorithm for a greater number of Gaussian distributions. We can also see that increasing the resolution of the input image 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 16 Gaussian distributions to model u_λ .

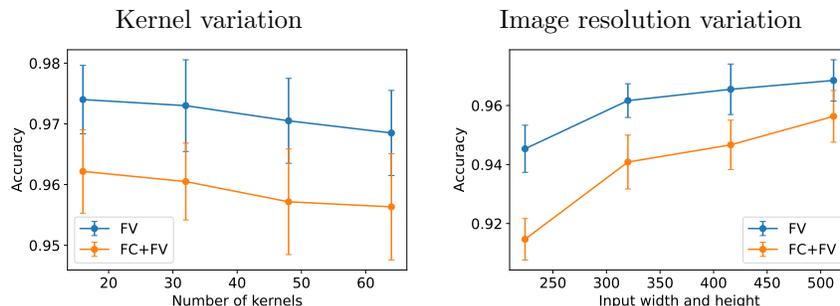


Figure 7: Accuracy of our proposed method for different number of kernels (Gaussian distributions in GMM) and different resolutions of the image provided to the CNN.

This is done because, as shown in Figure 7, our method behaves better with few distributions in this case. In Figure 8 we note that there is not much confusion when classifying the plant species. The classes that our model has most problems classifying are 8, wrong labelling around 14.5% of samples, and 5 and 6, in both around 8.5% 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 labelled 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 being interpreted as dotted regions. Class 5, which is mostly smooth with few grained areas, is confused with class 19, which is mostly grained. This confusion can be generated when focus is given to grained areas in images from class 5.

In Table 4 we list the accuracy of the best previous results on 1200Tex database that we found in literature, in comparison with our proposal. Here, the usage of our methodology made a huge difference in accuracy, scoring a result 9% better than the second best method (Non-Add Entropy). In fact, up to our knowledge, this result sets a new state-of-the-art accuracy on this database.

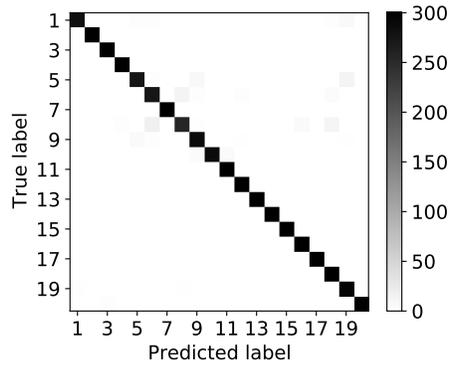


Figure 8: Confusion matrix for our method in 1200Tex. This matrix was computed using a classifier trained exclusively with information from FV descriptors.

Table 4: Comparison of accuracy in 1200Tex database with other methods in literature. Our method is presented as Multilayer-FV. 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 [8]	86.0 [35]
FV-VGGVD [8]	87.1 [9]
Fractal [35]	86.3
VisGraphNet [10]	87.4
Non-Add Entropy [9]	88.5
Multilayer-FV	97.4

7. Conclusions

In this work, we proposed and investigated the use of local features extracted from multiple convolutional layers and how this improves classification using Fisher Vector. More precisely, we computed the Fisher Vector on local features extracted from the last two convolutional layers and used it as texture descriptor.

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. This good performance can be explained by some points. One of them is the use of a mixture of more generalist features extracted from earlier convolutional layers and more domain-specific features extracted from later layers. The second factor is the adjustment of the input image to a resolution higher than the CNN standard, which affects both the number of local features and the specificity of a local feature to a given area of the input image. A last point is the use of normalization in the FC descriptor, which resulted in improvement of the method by combining the FV and FC descriptors.

The results expressed here also suggest that a combination of outputs from previous layers might be beneficial for classification accuracy. Also, different ways of combining local features from multiple layers may help to preserve better information from later layers and is a topic for future investigation.

Acknowledgements

L. O. L. gratefully acknowledges the financial support of Coordination for the Improvement of Higher Education Personnel, Brazil (CAPES) (Grant #1796018). J. B. F. gratefully acknowledges the financial support of São Paulo Research Foundation (FAPESP) (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] Abdi, H., Williams, L.J., 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics* 2, 433–459.
- [2] Amato, G., Bolettieri, P., Falchi, F., Gennaro, C., 2013. Large scale image retrieval using vector of locally aggregated descriptors, in: *International Conference on Similarity Search and Applications*, Springer. pp. 245–256.
- [3] Ansari, R.A., Buddhiraju, K.M., Malhotra, R., 2020. Urban change detection analysis utilizing multiresolution texture features from polarimetric sar images. *Remote Sensing Applications: Society and Environment* 20, 100418.
- [4] Caputo, B., Hayman, E., Mallikarjuna, P., 2005. Class-specific material categorisation, in: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, IEEE. pp. 1597–1604.
- [5] Casanova, D., de Mesquita Sá Junior, J.J., Bruno, O.M., 2009. Plant leaf identification using Gabor wavelets. *International Journal of Imaging Systems and Technology* 19, 236–243.
- [6] Chen, Z., Li, F., Quan, Y., Xu, Y., Ji, H., 2021. Deep texture recognition via exploiting cross-layer statistical self-similarity, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5231–5240.
- [7] Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A., 2014. Describing textures in the wild, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613.
- [8] Cimpoi, M., Maji, S., Kokkinos, I., Vedaldi, A., 2016. Deep filter banks for texture recognition, description, and segmentation. *International Journal of Computer Vision* 118, 65–94.

- [9] Florindo, J., Metze, K., 2021. Using non-additive entropy to enhance convolutional neural features for texture recognition. *Entropy* 23, 1259.
- [10] Florindo, J.B., Lee, Y.S., Jun, K., Jeon, G., Albertini, M.K., 2021. Visgraphnet: A complex network interpretation of convolutional neural features. *Information Sciences* 543, 296–308.
- [11] Gibert, D., Mateu, C., Planes, J., Vicens, R., 2018. Classification of malware by using structural entropy on convolutional neural networks, in: *Proceedings of the AAAI Conference on Artificial Intelligence*.
- [12] Hafiane, A., Palaniappan, K., Seetharaman, G., 2015. Joint adaptive median binary patterns for texture classification. *Pattern Recognition* 48, 2609–2620.
- [13] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.
- [14] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.
- [15] Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* .
- [16] Jaakkola, T., Haussler, D., 1998. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems* 11.
- [17] Jana, S., Basak, S., Parekh, R., 2017. Automatic fruit recognition from natural images using color and texture features, in: *2017 Devices for Integrated Circuit (DevIC)*, IEEE. pp. 620–624.
- [18] Jbene, M., El Maliani, A.D., El Hassouni, M., 2019. Fusion of convolutional neural network and statistical features for texture classification, in:

2019 International Conference on Wireless Networks and Mobile Communications (WINCOM), IEEE. pp. 1–4.

- [19] Lazebnik, S., Schmid, C., Ponce, J., 2005. A sparse texture representation using local affine regions. *IEEE transactions on pattern analysis and machine intelligence* 27, 1265–1278.
- [20] Leung, T., Malik, J., 2001. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International journal of computer vision* 43, 29–44.
- [21] Lin, T.Y., RoyChowdhury, A., Maji, S., 2017. Bilinear convolutional neural networks for fine-grained visual recognition. *IEEE transactions on pattern analysis and machine intelligence* 40, 1309–1322.
- [22] Liu, P., Zhang, H., Lian, W., Zuo, W., 2019. Multi-level wavelet convolutional neural networks. *IEEE Access* 7, 74973–74985.
- [23] Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 91–110.
- [24] Malik, J., Belongie, S., Leung, T., Shi, J., 2001. Contour and texture analysis for image segmentation. *International journal of computer vision* 43, 7–27.
- [25] Mao, S., Rajan, D., Chia, L.T., 2021. Deep residual pooling network for texture recognition. *Pattern Recognition* 112, 107817.
- [26] Nurzynska, K., Iwaszenko, S., 2020. Application of texture features and machine learning methods to grain segmentation in rock material images. *Image Analysis & Stereology* 39, 73–90.
- [27] Ojala, T., Pietikainen, M., Maenpaa, T., 2002. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence* 24, 971–987.

- [28] Perronnin, F., Dance, C., 2007. Fisher kernels on visual vocabularies for image categorization, in: 2007 IEEE conference on computer vision and pattern recognition, IEEE. pp. 1–8.
- [29] Perronnin, F., Sánchez, J., Mensink, T., 2010. Improving the fisher kernel for large-scale image classification, in: European conference on computer vision, Springer. pp. 143–156.
- [30] Ruichek, Y., et al., 2018. Local concave-and-convex micro-structure patterns for texture classification. *Pattern Recognition* 76, 303–322.
- [31] Sánchez, J., Perronnin, F., Mensink, T., Verbeek, J., 2013. Image classification with the fisher vector: Theory and practice. *International journal of computer vision* 105, 222–245.
- [32] Scalco, E., Rizzo, G., 2017. Texture analysis of medical images for radiotherapy applications. *The British journal of radiology* 90, 20160642.
- [33] Sharan, L., Liu, C., Rosenholtz, R., Adelson, E.H., 2013. Recognizing materials using perceptually inspired features. *International journal of computer vision* 103, 348–371.
- [34] Sharan, L., Rosenholtz, R., Adelson, E., 2009. Material perception: What can you see in a brief glance? *Journal of Vision* 9, 784–784.
- [35] Silva, P.M., Florindo, J.B., 2021. Fractal measures of image local features: an application to texture recognition. *Multimedia Tools and Applications* 80, 14213–14229.
- [36] Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .
- [37] Song, Y., Zhang, F., Li, Q., Huang, H., O’Donnell, L.J., Cai, W., 2017. Locally-transferred fisher vectors for texture classification, in: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4912–4920.

- [38] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1929–1958.
- [39] Tan, M., Le, Q., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks, in: *International Conference on Machine Learning*, PMLR. pp. 6105–6114.
- [40] Tan, M., Le, Q., 2021. Efficientnetv2: Smaller models and faster training, in: *International Conference on Machine Learning*, PMLR. pp. 10096–10106.
- [41] Wan, W., Chen, J., Li, T., Huang, Y., Tian, J., Yu, C., Xue, Y., 2019. Information entropy based feature pooling for convolutional neural networks, in: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3405–3414.
- [42] Xu, Y., Ji, H., Fermüller, C., 2009. Viewpoint invariant texture description using fractal analysis. *International Journal of Computer Vision* 83, 85–100.
- [43] Xu, Y., Li, F., Chen, Z., Liang, J., Quan, Y., 2021. Encoding spatial distribution of convolutional features for texture representation. *Advances in Neural Information Processing Systems* 34.
- [44] Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C., 2007. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision* 73, 213–238.