Evaluating Deep Convolutional Neural Networks as Texture Feature Extractors

Leonardo F. S. Scabini[1](✉), Rayner H. M. Condori[2], Lucas C. Ribas[2], and Odemir M. Bruno[1]

[1] São Carlos Institute of Physics, University of São Paulo, São Carlos, SP, Brazil
{scabini,bruno}@ifsc.usp.br

[2] Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, SP, Brazil
{raynerhmc,lucasribas}@usp.br

Abstract. Texture is an important visual property which has been largely employed for image characterization. Recently, Convolutional Networks has been the predominant approach on Computer Vision, and their application on texture analysis shows interesting results. However, their popularity steers around object recognition, and several convolutional architectures have been proposed and trained for this task. Therefore, this works evaluates 17 of the most diffused Deep Convolutional Neural Networks when employed for texture analysis as feature extractors. Image descriptors are obtained through Global Average Pooling over the output of the last convolutional layer of networks with random weights or learned from the ImageNet dataset. The analysis is performed under 6 texture datasets and using 3 different supervised classifiers (KNN, LDA, and SVM). Results using networks with random weights indicates that the architecture alone plays an important role in texture characterization, and it can even provide useful features for classification for some datasets. On the other hand, we found that although ImageNet weights usually provide the best results it can also perform similar to random weights in some cases, indicating that transferring convolutional weights learned on ImageNet may not always be appropriate for texture analysis. When comparing the best models, our results corroborate that DenseNet presents the highest overall performance while keeping a significantly small number of hyperparameters, thus we recommend its use for texture characterization.

Keywords: Deep Convolutional Neural Networks · Texture analysis · Feature extraction

1 Introduction

Texture is a highly discriminating visual characteristic that has been widely studied since the 1960s. In Computer Vision (CV), texture analysis is an active field of research and there is a constant need for new methods for texture characterization in several areas that rely on image recognition systems. Many techniques have then been proposed in the past years [16] composing a wide and heterogeneous literature on texture analysis, however, the recent advances on Deep Learning made Neural Networks the predominant approach on CV. This trend grew mainly after the success obtained by the AlexNet [11] Deep Convolutional Neural Network (DCNN) on the 2012 ImageNet [5] challenge. Several DCNN architectures have then been proposed for object recognition, with varying structural properties such as different convolutional blocks and their combinations. These models with pre-trained weights on ImageNet have then been employed as texture feature extractors [3,30], achieving promising results.

Although it has been shown that transferring the learning of some DCNN pre-trained on the ImageNet dataset into texture analysis usually provides good results, little is known on the impacts of different architectures on performance. Moreover, there are no obvious reasons to believe that the use of ImageNet weights, which have been trained for object recognition, is the best approach for texture characterization. Therefore, this works presents a broad evaluation of 17 DCNN models considering these aspects. Texture descriptors are obtained through the Global Average Pooling (GAP) over the output of the last convolutional layer, which is then fed to a supervised classifier. We analyze the importance of the architecture alone and the gain on using ImageNet weights for texture analysis by comparing the network performance with random weights. Moreover, our experiments include 6 texture datasets with varying properties and 3 classifiers (KNN, LDA, and SVM) in order to measure the capacity of each model. We also analyze the efficiency of the networks for texture analysis in terms of its size (number of hyperparameters) and average performance.

2 Theoretical Background

2.1 Texture Analysis

Texture is a key feature of many types of images and, over the decades, many methods for texture representation (i.e., extraction of features to describe the texture information) have been proposed. In the last century, the research in texture representation mainly focused on two-well established types of approaches: filtering-based and statistical-based [16]. In the filtering-based approaches, the image is convolved with a filter bank. In this category, some important techniques are Gabor filters, wavelet pyramids, and simple linear filters. The statistical-based approaches model the texture using kth order statistics and use probability distributions to characterize the images, such as the Markov Random Field, Fractal models and Gray Level Co-occurrence Matrix (GLCM).

At the end of the last century and in the 2000s, there was the propose of texton-based approaches such as Bag of Textons [13] and Bag of Words [4]. In these approaches, a dictionary of textons is obtained and the images are represented by statistics over the texton dictionary. Furthermore, the need for features invariant to scale, rotation, illumination and viewpoint stimulated the development of local invariant descriptors, such as SIFT [17] and LBP [19]. These local descriptors were extensively used in computer vision, however, more recently, there was a renaissance of deep neural networks approaches from the work of Krizhevsky et al. in 2012 [11]. In this way, actually, the research in textures and image analysis has focused on deep learning approaches. In texture analysis, important advances have been reported such as in [3,30].

2.2 Deep Convolutional Neural Networks

A convolutional network is a type of neural network that is mainly used for analyzing images, videos and any kind of spatially organized data, and it has been studied since the 1990's [12]. In a general fashion, we can split a convolutional neural network into two well-defined components: (i) the feature extraction part and (ii) the classification part. In most of the DCNN, the latter component is organized into one or more *fully-connected layers*, like a traditional multilayer neural network. On the other hand, the feature extraction part consists of convolutional layers which can vary greatly from one network to another, as various convolutional blocks have been proposed throughout the years. In early models [12] it consisted of a few convolutional and pooling layers, however, since the introduction of the *deep learning* concept new architectures are increasingly grouping layers. Moreover, different structures of convolutional blocks have been proposed such as the Inception modules [27] and Residual connections [8].

The input and output of layers in the feature extraction part of a DCNN consists of a set of $n_{\mathbf{A}^{(l)}}$ stacked 2-D matrices $\mathbf{A}^{(l)}$, that we refer here as "feature maps", where l indicates the layer's depth and $\mathbf{A}_j^{(l)}$ represents the j-th feature map ($1 \leq j \leq n_{\mathbf{A}^{(l)}}$). In the case of convolutional layers, the trainable weights are organized into a set of $n_{\mathbf{F}^{(l)}}$ filters of fixed size. This set, also known as the filter bank ($\mathbf{F}^{(l)}$), is in charge of computing a new set of feature maps $\mathbf{A}^{(l+1)}$ by convolving $\mathbf{A}^{(l)}$ with every filter in $\mathbf{F}^{(l)}$. In addition, there are vastly more types of layers in a DCNN that also yield a feature map, for example, the Dropout layer, the concatenation layer, and batch normalization.

The set of feature maps $\mathbf{A}^{(n)}$ represents the output of the last convolutional layer (n) of a DCNN, and it is usually flattened to fit into a fully-connected layer. Another approach is the use of Global Average Pooling (GAP) [14] to reduce the data before passing it to a classification layer

$$\varphi_j = \frac{\sum \mathbf{A_j}^{(n)}}{n_{\mathbf{A_j}^{(n)}}} \tag{1}$$

In other words, the GAP takes the mean value of every feature map $\mathbf{A}_j^{(n)}$ and concatenates into a vector φ. This vector acts as a final image descriptor, therefore pre-trained DCNN can be used as feature extractors by obtaining φ.

3 Experiments

3.1 Considered Networks

For a broad analysis of DCNN architectures we considered 17 models with varying depth and modules proposed along the past years (between 2012 and 2018), details are given on Table 1. These are well-known models which have been extensively evaluated by the deep learning community recently, and its source code is open access[1]. All networks are applied as feature extractors by removing its last fully-connected layers and

[1] Keras models: https://keras.io/applications/
PyTorch models: https://github.com/Cadene/pretrained-models.pytorch.

taking the GAP from feature maps of the last convolutional layer, composing the feature vector φ. A more elaborated approach for feature map characterization is presented in [3], however, our goal here is to analyze the difference in performance between these architectures rather than maximizing the obtained results. Regarding the weight initialization policy, we considered the standard definition of the framework from which the model is imported, as this is the most common case when applying these networks. In the Keras 2.2.4 library, the 2D convolutional layers employ the Glorot uniform random weight initialization [6], and the bias terms are initialized with 0. In the case of AlexNet, PolyNet, PNASNet5 and all the Squeeze-and-Excitation networks, the PyTorch 1.0.0 default initialization policy is used, that is: the convolutional layer weights and bias are sampled from an uniform distribution $\mathcal{U}(-\sqrt{k}, \sqrt{k})$, where $k = 1/N_{w_c}$. The term N_{w_c} is the number of weights at the convolutional layer c. On the other hand, in the case of the ResNet and DenseNet models, the weights of the convolutional layers are initialized according to the Kaiming normal distribution [7], $\mathcal{N}$ (mean = 0 and std = $\sqrt{2/N_{w_c}}$), the batch normalization weights and bias are respectively initialized with 1s and 0s.

Table 1. Details on each DCNN considered for analysis. The network size is measured by the number of hyperparameters (full includes fully-connected layers and GAP refers to only the feature extraction part). $|\varphi|$ represents the size of the resulting feature vector.

Network	Weight initialization	Size (in millions)		$\lvert\varphi\rvert$	Framework
		Full	GAP		
AlexNet [11]	$\mathcal{U}(-\sqrt{k}, \sqrt{k}), k = 1/N_{w_c}$	61.1008	2.4697	256	Pytorch 1.0.0
VGG16 [25]	Glorot uniform	138.3575	14.7147	512	Keras 2.2.4
VGG19 [25]	Glorot uniform	143.6672	20.0244	512	Keras 2.2.4
InceptionV3 [27]	Glorot uniform	23.8518	21.8028	2048	Keras 2.2.4
ResNet18 [8]	Kaiming normal	11.6895	11.1765	512	Pytorch 1.0.0
ResNet50 [8]	Kaiming normal	25.5570	23.5080	2048	Pytorch 1.0.0
ResNet152 [8]	Kaiming normal	60.1928	58.1438	2048	Pytorch 1.0.0
InceptionResNetV2 [26]	Glorot uniform	55.8737	54.3367	1536	Keras 2.2.4
PolyNet [31]	$\mathcal{U}(-\sqrt{k}, \sqrt{k}), k = 1/N_{w_c}$	95.3666	93.3176	2048	Pytorch 1.0.0
DenseNet121 [10]	Kaiming normal	7.9789	6.9539	1024	Pytorch 1.0.0
DenseNet201 [10]	Kaiming normal	20.0139	18.0929	1920	Pytorch 1.0.0
PNASNet5 (large) [15]	$\mathcal{U}(-\sqrt{k}, \sqrt{k}), k = 1/N_{w_c}$	86.0577	81.7367	4320	Pytorch 1.0.0
SENet154 [9]	$\mathcal{U}(-\sqrt{k}, \sqrt{k}), k = 1/N_{w_c}$	115.0890	113.0400	2048	Pytorch 1.0.0
SEResNet50 [9]	$\mathcal{U}(-\sqrt{k}, \sqrt{k}), k = 1/N_{w_c}$	28.0880	26.0390	2048	Pytorch 1.0.0
SEResNet152 [9]	$\mathcal{U}(-\sqrt{k}, \sqrt{k}), k = 1/N_{w_c}$	66.8218	64.7728	2048	Pytorch 1.0.0
SEResNeXt50 [9,29]	$\mathcal{U}(-\sqrt{k}, \sqrt{k}), k = 1/N_{w_c}$	27.5599	25.5109	2048	Pytorch 1.0.0
SEResNeXt101 [9,29]	$\mathcal{U}(-\sqrt{k}, \sqrt{k}), k = 1/N_{w_c}$	48.9554	46.9064	2048	Pytorch 1.0.0

3.2 Experimental Protocol

Six color texture datasets with varying properties are used in our experiments. We considered well-known traditional datasets and also recent and more complicated ones:

(1) Vistex [20] (864 images, 54 classes); (2) USPtex [2] (2292 images, 191 classes); (3) Outex13 [18] (test suite Outex_TC_00013, 1360 images, 68 classes); (4) CUReT [28] (5612 images, 61 classes); (5) MBT [1] (2464 images, 154 classes, clipping of original images as in [22]); and (6) FMD [24] (1000 images, 10 classes). To process the datasets with the studied DCNN we resize the images to the corresponding network input, except for the FMD dataset where images are not square, thus we use the original images to preserve its proportions (the DCNN will produce valid feature maps for GAP). Regarding the gray-level images present in the FMD dataset, we transform them in RGB by replicating its values for all channels.

We considered different classifiers in our experiments: (1) a K-Nearest-Neighbors (KNN) approach with $k = 1$; (2) Linear Discriminant Analysis (LDA) using a least squares solution; (3) The Support Vector Machine (SVM) using a linear kernel and penalty parameter $C = 1$. The classification experiments are performed with 10 random splits into half for training and a half for the test in a stratified fashion. In the case of the network random weight initialization, 10 different networks are initialized and the classification is done using the same 10 random splits for each one, yielding a total of 100 train and test procedures. The performance is measured by the mean and standard deviation of the test accuracy over the iterations.

4 Results Analysis

4.1 Contribution of the Architecture Versus Learned Weights

Intriguing results [21] have shown that neural networks, convolutional or not, with completely random weights can achieve interesting performance. This is, in fact, similar to randomized neural networks [23], where a hidden layer with random weights has the purpose of projecting non-linearly the input data in another dimensional space where it is more likely that the feature vectors are linearly separable. Therefore, on these networks, the performance relies on the training of the output layer and the shape of the hidden layer rather than on its weights, as they are random. In the case of DCNN, the feature extraction part plays a similar role than the hidden layer of the randomized neural network, projecting the data. The discussion presented in [21] corroborates that the convolutional architecture alone is of high importance for object recognition, where they propose a fast method for architecture selection based on the performance on random weights. In this context, we verify the contribution of the architecture alone and the learned weights for texture analysis by analyzing the performance of DCNN using random weights or pre-trained on the ImageNet dataset. Figure 1 shows the results obtained with the simplest classifier, KNN, that we choose in this experiment in order to highlight the quality of the features itself so that the result does not rely mostly on the classification technique. We can verify the contribution of the learned weights over the random ones by the distance of the points to the dotted line, which represents $x = y$ (where random weight equals the accuracy of the ImageNet weights).

It is possible to notice different cases regarding random and ImageNet weights, where performance varies both according to the dataset or the DCNN model. First, the standard deviation on random weights indicates that different initialization has a relatively small impact in comparison to the architecture performance itself. Regarding the

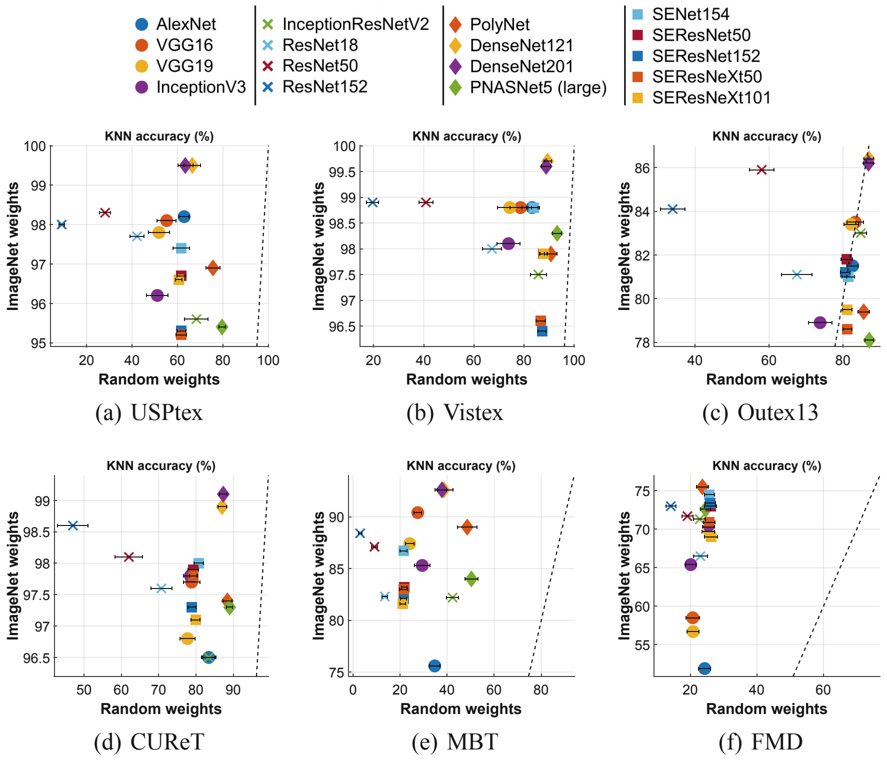

Fig. 1. Correlation between the accuracy using DCNN with random weights or pre-trained on ImageNet. The doted line represents the diagonal of the plan, i.e. $x = y$. The small black horizontal lines represents the standard deviation over 10 random networks.

variation between datasets, we observe in USPtex, Vistex, and CUReT a similar behavior, where the learned weights provide a small but relevant improvement. Nonetheless, in the Outex13 dataset, most of the networks have similar performance either with random or ImageNet weights, some of them, in fact, performs better with random weights. This raises questions concerning the applicability of the ImageNet weights for color texture characterization. On the other hand, in the MBT and FMD datasets, the gain from the learned weights is significantly higher for all networks. In terms of texture classification, these two datasets present different properties in comparison to the others. MBT, for instance, have complex intraband and interband spatial variations, forming unusual spectral texture patterns. It seems that these DCNN architectures alone are not capable of effectively capturing these spectral variations, while learned weights play a significant role in texture characterization. In the FMD dataset, the gain provided by the learned weights is the higher, where most networks with random weights perform poorly (around 20% and 30% of accuracy). FMD is a *in-the-wild* dataset, with texture images from various different objects in the same class, which explains why ImageNet weights are important here, as they are learned in a wide object recognition scenario.

In conclusion, the architecture alone may be responsible for most of the network performance in some cases, however, in most cases, the use of learned weights is important for achieving higher performance. On the other hand, the results on the Outex13 dataset indicates that, for texture analysis, the current models need either modification in its architecture and/or new training procedures beyond ImageNet, for the learning of more unusual color texture patterns.

It is also possible to notice that some networks have higher random weight performance than the other networks in most cases, but in other hand are overcome when using the ImageNet weights. This happens with the PNASNet5 and PolyNet networks, except for the FMD dataset. In fact, in the FMD dataset deeper networks achieve the highest performance. The ResNet networks, although performing among the best in some datasets using the ImageNet weights, have a considerably low performance with random weights, which seems to get worse as we increase its depth. Concerning the best networks in most cases, we can notice the DenseNet model, with both depths we tested (121 and 201 layers), performing above the others in all cases, except on the FMD dataset, where the highest result is achieved by PolyNet.

4.2 Performance Under Different Classifiers

We include the DCNN performance under two additional classifiers, LDA and SVM, results using the ImageNet weights are shown in Table 2. Both these classifiers perform better than KNN, where LDA is slightly superior overall (except for the FMD dataset). Regarding the best networks, we can see a similar pattern as observed with the KNN classifier, where the DenseNet model overcomes the other networks in most cases. In the USPtex dataset, the highest results are obtained by both DenseNet models (121 and 201 layers deep), with $99.8\% \pm 0.1$ of accuracy using either LDA or SVM. For the Vistex dataset, DenseNet201 achieves $100\% \pm 0.1$ with LDA and $99.9\% \pm 0.2$ with SVM, however various networks performs above 99.7% on this dataset, as ResNet50, PolyNet, PNASNET5, SENet154 and SEResNet152. For Outex13 results are significantly lower than for the former 2 datasets, and DenseNet201 present the higher performance ($91.8\% \pm 0.5$ with LDA and $90.8\% \pm 0.7$ with SVM). DenseNet201 also present the highest results for the CUReT ($99.4\% \pm 0.2$ with LDA and $99.8\% \pm 0.1$ with SVM) and MBT datasets ($97.6\% \pm 0.3$ with LDA and $97.2\% \pm 0.3$ with SVM). The results obtained on FMD vary if compared to the other datasets, where DenseNet is overcome by other networks such as PolyNet, SENet154, and SEResNet50. The highest results are obtained by PolyNet using LDA, with $87.3\% \pm 0.9$, and SENet154 using SVM, with $86.4\% \pm 0.8$. However, the standard deviation of their results on this dataset puts these networks in a similar performance range.

4.3 Efficiency Comparison

For a final comparison, we considered the efficiency of the network, defined here as the correlation between size (number of hyperparameters) and performance (mean accuracy for the six datasets). The number of hyperparameters of a network is directly related to the number of operations performed throughout its layers, which is a good measure for the cost of computing the image descriptor. Figure 2 shows this analysis with the three

Table 2. Results of each DCNN using the LDA and SVM classifiers, representing the mean accuracy and the standard deviation for 10 random data splits into half for train and half for test. Best results in each dataset are highlighted in bold type.

Network	USPtex	Vistex	Outex13	CUReT	MBT	FMD
LDA						
Alexnet	98.7 ± 0.2	99.4 ± 0.3	87.2 ± 1.0	90.5 ± 0.7	88.2 ± 0.9	70.0 ± 1.4
VGG16	98.7 ± 0.2	99.0 ± 0.4	86.6 ± 0.5	93.6 ± 0.5	93.4 ± 0.7	74.8 ± 1.3
VGG19	98.5 ± 0.3	98.8 ± 0.4	86.3 ± 0.9	93.2 ± 0.6	92.6 ± 0.7	72.5 ± 1.8
InceptionV3	98.4 ± 0.3	98.8 ± 0.5	86.1 ± 0.8	97.4 ± 0.2	92.2 ± 0.5	80.6 ± 1.2
InceptionResNetV2	97.7 ± 0.4	98.8 ± 0.3	88.2 ± 0.6	96.4 ± 0.5	91.2 ± 0.7	82.6 ± 2.2
ResNet18	98.8 ± 0.3	99.3 ± 0.3	86.9 ± 0.7	95.7 ± 0.4	91.3 ± 0.8	82.8 ± 0.4
ResNet50	99.4 ± 0.1	99.7 ± 0.3	90.5 ± 0.5	98.6 ± 0.2	94.1 ± 0.8	84.3 ± 1.5
ResNet152	99.6 ± 0.2	99.5 ± 0.3	89.4 ± 0.8	98.9 ± 0.2	94.8 ± 0.6	86.1 ± 0.7
PolyNet	99.3 ± 0.2	99.6 ± 0.4	88.5 ± 0.7	98.5 ± 0.2	95.4 ± 0.5	**87.3 ± 0.9**
DenseNet121	**99.8 ± 0.1**	99.9 ± 0.2	91.1 ± 0.8	98.9 ± 0.2	97.4 ± 0.3	85.9 ± 0.8
DenseNet201	**99.8 ± 0.1**	**100.0 ± 0.1**	**91.8 ± 0.5**	**99.4 ± 0.2**	**97.6 ± 0.3**	86.8 ± 1.1
PNASNet5(large)	98.7 ± 0.4	99.7 ± 0.3	86.9 ± 0.9	98.8 ± 0.3	93.5 ± 1.0	83.6 ± 1.6
SENet154	99.5 ± 0.2	99.7 ± 0.3	87.7 ± 0.7	98.1 ± 0.3	93.4 ± 0.4	87.1 ± 1.2
SEResNet50	99.5 ± 0.2	99.6 ± 0.3	88.7 ± 0.6	97.8 ± 0.2	93.2 ± 0.7	87.1 ± 0.7
SEResNet152	98.9 ± 0.2	99.8 ± 0.3	90.1 ± 0.9	96.9 ± 0.3	92.1 ± 0.4	86.0 ± 0.8
SEResNeXt50	99.2 ± 0.3	99.5 ± 0.3	86.2 ± 0.9	97.4 ± 0.3	92.0 ± 0.8	86.0 ± 0.9
SEResNeXt101	99.3 ± 0.3	99.5 ± 0.4	86.9 ± 0.7	96.9 ± 0.3	91.9 ± 0.5	86.2 ± 1.0
SVM						
Alexnet	98.9 ± 0.3	99.4 ± 0.4	86.8 ± 0.8	98.3 ± 0.2	86.6 ± 0.6	66.7 ± 2.3
VGG16	99.0 ± 0.2	99.4 ± 0.4	87.7 ± 0.7	98.6 ± 0.1	94.5 ± 0.7	72.5 ± 1.9
VGG19	98.5 ± 0.3	99.3 ± 0.3	87.5 ± 0.7	98.1 ± 0.3	93.3 ± 0.6	69.6 ± 2.4
InceptionV3	98.0 ± 0.3	99.0 ± 0.4	85.4 ± 1.3	99.0 ± 0.2	91.6 ± 0.4	80.7 ± 1.0
InceptionResNetV2	97.6 ± 0.2	98.7 ± 0.3	87.8 ± 1.0	98.6 ± 0.2	89.2 ± 0.5	81.5 ± 1.6
ResNet18	99.1 ± 0.2	98.9 ± 0.5	87.3 ± 0.6	98.9 ± 0.2	90.9 ± 0.8	80.7 ± 1.0
ResNet50	99.2 ± 0.4	99.5 ± 0.4	89.5 ± 0.8	99.3 ± 0.1	92.9 ± 0.6	82.8 ± 1.2
ResNet152	99.1 ± 0.3	99.5 ± 0.4	88.3 ± 0.8	99.4 ± 0.2	93.8 ± 0.6	84.6 ± 1.2
PolyNet	98.7 ± 0.3	99.7 ± 0.4	87.3 ± 0.6	99.2 ± 0.1	94.5 ± 0.6	86.1 ± 0.9
DenseNet121	**99.8 ± 0.1**	99.8 ± 0.2	90.3 ± 0.8	99.7 ± 0.1	96.9 ± 0.5	84.3 ± 1.2
DenseNet201	**99.8 ± 0.1**	**99.9 ± 0.2**	**90.8 ± 0.7**	**99.8 ± 0.1**	**97.2 ± 0.3**	85.2 ± 1.1
PNASNet5 (large)	98.0 ± 0.4	99.3 ± 0.4	85.0 ± 1.0	99.0 ± 0.2	91.5 ± 0.7	84.3 ± 1.1
SENet154	99.1 ± 0.2	99.6 ± 0.3	87.0 ± 0.6	99.3 ± 0.1	92.8 ± 0.7	**86.4 ± 0.8**
SEResNet50	99.0 ± 0.4	99.4 ± 0.5	87.5 ± 0.7	99.3 ± 0.1	92.3 ± 0.4	85.6 ± 1.0
SEResNet152	98.2 ± 0.3	99.4 ± 0.3	88.6 ± 0.7	99.1 ± 0.2	91.3 ± 0.6	84.8 ± 0.7
SEResNeXt50	98.7 ± 0.4	99.2 ± 0.5	85.6 ± 0.4	99.2 ± 0.2	91.5 ± 0.6	83.9 ± 0.9
SEResNeXt101	98.7 ± 0.4	99.1 ± 0.5	86.0 ± 0.9	99.0 ± 0.2	90.7 ± 0.7	84.7 ± 1.0

classifiers (KNN, LDA, and SVM), where network size refers to the feature extraction part only. The observed behavior for the three classifiers is similar where, overall, there is a positive correlation between the increase in size and performance. However, the DenseNet model seems to escape this rule, presenting the highest mean performance together with a relatively small size. It has a size around the smallest networks analyzed while performing significantly better. After DenseNet, the ResNet50 and SEResNet50 models present the highest efficiency, with performance similar to big networks while keeping a relatively small size.

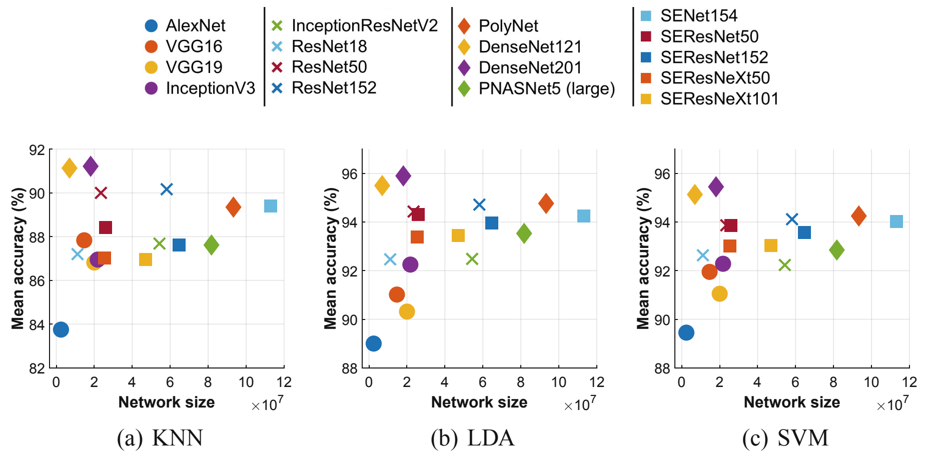

Fig. 2. The relation between the mean accuracy over the 6 datasets and the network size, i.e. the number of hyperparameters of the part used as feature extractor (no fully connected layers). Results are shown for 3 different classifiers (KNN, LDA and SVM).

5 Conclusion

On this work we performed a broad analysis of different DCNN models on texture analysis, specifically as texture feature extractors coupled with a supervised classifier. Our experiments include 17 DCNN, 6 texture datasets and 3 different classifiers (KNN, LDA, and SVM). The results indicate various interesting properties of these networks and datasets. First, we observed that the DCNN architecture alone is of great importance for texture characterization, as results with random weights point out, and learned weights make a complementary contribution for performance. However, weights learned on ImageNet may not be always appropriate for texture analysis, as results in the Outex13 dataset indicates, where random weights outperformed ImageNet weights for some networks. Regarding the performance of the models, our results indicate that the DenseNet architecture, both with 121 or 201 layers deep, is highly recommendable for texture analysis as it achieves the highest results in all datasets, except FMD. Moreover, our efficiency analysis indicates a positive correlation between the increase in network size and performance, but DenseNet escapes this rule as it has a significantly small size while keeping higher performance. The pattern of performance difference between networks is similar regardless of the chosen classifier. However, the LDA and SVM classifiers perform better than KNN, and LDA is slightly superior overall, except for the CUReT dataset where SVM is better. As future works for a better understanding of DCNN architectures, it is possible to explore the impact of different weight initialization techniques, as we considered here only the default from the frameworks. Moreover, it is possible to explore the use of more sophisticated feature map characterization techniques besides from GAP, in order to obtain better texture descriptors from each network.

Acknowledgements. L. F. S. Scabini acknowledges support from CNPq (grant #142438/2018-9) and the São Carlos Institute of Physics (CAPES funding). R. H. M. Condori acknowledges support from FONDECYT, an initiative of the National Council of Science, Technology and Technological Innovation-CONCYTEC (Peru). L. C. Ribas gratefully acknowledges the financial support grant #s 2016/23763-8 and 2019/03277-0, São Paulo Research Foundation (FAPESP). O. M. Bruno acknowledges support from CNPq (grants #307797/2014-7 and #484312/2013-8) and FAPESP (grants #14/08026-1 and #16/18809-9). The authors are also grateful to the NVIDIA GPU Grant Program for the donation of the Quadro P6000 and the Titan Xp GPUs used on this research.

References

1. Abdelmounaime, S., Dong-Chen, H.: New Brodatz-based image databases for grayscale color and multiband texture analysis. ISRN Mach. Vis. **2013**, 14 (2013)
2. Backes, A.R., Casanova, D., Bruno, O.M.: Color texture analysis based on fractal descriptors. Pattern Recogn. **45**(5), 1984–1992 (2012)
3. Cimpoi, M., Maji, S., Kokkinos, I., Vedaldi, A.: Deep filter banks for texture recognition, description, and segmentation. Int. J. Comput. Vis. **118**(1), 65–94 (2016)
4. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision, ECCV, Prague, vol. 1, pp. 1–2 (2004)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 248–255. IEEE (2009)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010)
7. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV 2015, pp. 1026–1034. IEEE Computer Society, Washington, DC (2015)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
9. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)
10. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
12. LeCun, Y., et al.: Handwritten digit recognition with a back-propagation network. In: Advances in Neural Information Processing Systems, pp. 396–404 (1990)
13. Leung, T., Malik, J.: Representing and recognizing the visual appearance of materials using three-dimensional textons. Int. J. Comput. Vis. **43**(1), 29–44 (2001)
14. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
15. Liu, C., et al.: Progressive neural architecture search. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11205, pp. 19–35. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01246-5_2

16. Liu, L., Chen, J., Fieguth, P., Zhao, G., Chellappa, R., Pietikäinen, M.: From BoW to CNN: two decades of texture representation for texture classification. Int. J. Comput. Vis. **127**(1), 74–109 (2019)
17. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
18. Ojala, T., Maenpaa, T., Pietikainen, M., Viertola, J., Kyllonen, J., Huovinen, S.: Outex-new framework for empirical evaluation of texture analysis algorithms. In: Proceedings of the 16th International Conference on Pattern Recognition, vol. 1, pp. 701–706. IEEE (2002)
19. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
20. Pickard, R., Graszyk, C., Mann, S., Wachman, J., Pickard, L., Campbell, L.: VisTex Database. MIT Media Lab, Cambridge (1995)
21. Saxe, A.M., Koh, P.W., Chen, Z., Bhand, M., Suresh, B., Ng, A.Y.: On random weights and unsupervised feature learning. In: ICML, vol. 2, p. 6 (2011)
22. Scabini, L.F., Condori, R.H., Gonçalves, W.N., Bruno, O.M.: Multilayer complex network descriptors for color-texture characterization. Inf. Sci. **491**, 30–47 (2019)
23. Schmidt, W.F., Kraaijveld, M.A., Duin, R.P.W.: Feedforward neural networks with random weights. In: Proceedings of the 11th IAPR International Conference on Pattern Recognition, Conference B: Pattern Recognition Methodology and Systems, vol. II, pp. 1–4 (1992)
24. Sharan, L., Rosenholtz, R., Adelson, E.: Material perception: what can you see in a brief glance? J. Vis. **9**, 784 (2009)
25. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
26. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
27. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016
28. Varma, M., Zisserman, A.: A statistical approach to texture classification from single images. Int. J. Comput. Vis. **62**(1), 61–81 (2005)
29. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1492–1500 (2017)
30. Zhang, H., Xue, J., Dana, K.: Deep TEN: texture encoding network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 708–717 (2017)
31. Zhang, X., Li, Z., Change Loy, C., Lin, D.: PolyNet: a pursuit of structural diversity in very deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 718–726 (2017)