Convolutional neural network to classify images based on texture

Johana M. Ramirez Borda
Los Andes University
jm.ramirez11@uniandes.edu.co

Francisco A. Rozo Forero
Los Andes University
fa.rozo1843@uniandes.edu.co

Abstract

Image texture can be described as the pattern of variations in the gray levels of an image and provides information about smoothness, coarseness, bumpiness and also regularity. In literature there can be found several approaches used to analyze, classify or segmentate images based on this type of information (Texture). Some methods started using Minimum Distance Classifiers, others using Bag of Words, Fisher vectors and Deep learning. In this project, a convolutional neural network (CNN) was crated and trained with 15.000 train images provided by [7] and extracted from [3] in order to classify 2.500 test images belonging to 25 categories. In this project a neural network was implemented by using convolution, max-pooling, rectified linear unit and softmax operators.

1. Introduction

Texture analysis is a common task in computer vision and refers to the characterization of regions in an image by their texture content. In this sense, the texture is related to roughness, bumpiness, smoothness... etc, which quantitatively refers to variations in the gray levels of an image. This analysis is used in numerous applications such as medical image processing, automated inspection, remote sensing and texture segmentation. Even more, texture analysis is used when trying to identify objects in an image that are more characterized by their texture than by intensity [4].

One recent approach to texture analysis is based on the use of Deep Neural Networks, which have shown the ability to learn data representations in both supervised and unsupervised settings [1]. However, texture classification is not an easy task due to large intraclass variation and low interclass distinction. Studies have shown that although methods such as bag of words (BoW) had good results, Fisher vectors (FV), which is based on Gaussian mixture models is able to provide more discriminative power for images with low inter-class distinction [6]. Moreover, in literature there can be found many methodologies based on neural networks, for example, Leung and Peterson [5] used feed-forward neural networks with 1 to 2 hidden layers for the classification and segmentation of textred images. Also, Kulkarni and Byars [5] suggested an artificial NN model to extract frecuency-domain features that can recognize textures.

Considering this, the objective of the present project is to design and train a CNN (convolutional neural network) from a random weight initialization

2. Materials and methods

2.1. Database

The dataset used in this project was provided by [7] and contains randomly sampled 128x128 patches from each image found in the database of the Ponce group [3]. The provided .zip contains one folder for each test, train and validation sets. Within the test one, there are 2.500 images, all labeled as ”0”. The train folder contains 15.000 images divided into 25 categories, 600 images each, and all the images belonging to one category are within a sub-folder named with the corresponding category label. The validation set is similar to the train but it contains 2.500 images. Finally, there is also a .text file that contains the name corresponding to each label, for example, ”bark2 T03, wood1 T05”, etc.

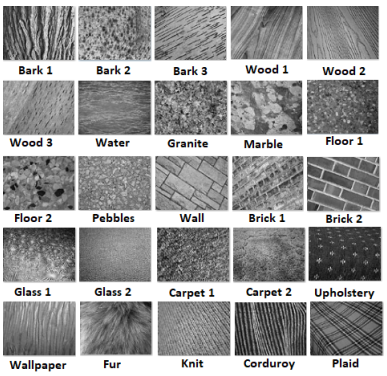

Figure 1. Example of one random image within each category of the database.

As it can be seen, there are categories such as bark, wood, water, floor, carpets,glass,wallpaper, etc .Also, each category within the database has images taken from different angles and distances.

2.2. CNN creation and training

When creating the neural network, and based on the practical retrieved from [2], we tried to implement four main building blocks:

- Convolution: implementation of a bank of filters to the input images preserving the resolution of the feature map:

$$y_{i'j'k'} = \sum_{ijk} w_{ijkk'} x_{i+i',j+j',k} \quad (1)$$

  Which operate on a tensor *x* with *K* channels and *K'* filters.

- Pooling: coalescence of nearby feature values into one. In the implemented neural network it was used max-pooling trough the max operator:

$$y_{ijk} = max(y_{i'j'k} : i \leq i' < i + p, j \leq j' < j + p) \quad (2)$$

- Rectified Linear Unit (ReLU): activation function that takes the positive part of it input argument. Is defined by:

$$y_{ijk} = max(0, x_{ijk}) \quad (3)$$

- Softmax: this function squashes each one of the outputs of the last layer to be between 0 and 1. Moreover, it also divides each output such that the total sum of the outputs is equal to 1. This is necessary since the output of this function is equivalent to a probability distribution and it gives information about the probability of an image to correspond to certain class. It is defined by:

$$y_{ijk'} = \frac{e^{x_{ijk'}}}{\sum_k e^{x_{ijk}}} \quad (4)$$

3. Results and discussion

Layers inside neural networks are represented as matrices in $\mathbb{R}^n$. Hence, the way they connect is according to their dimensions. This means, that the outputs of each layer should coincide with the dimensions of the inputs from the next layer. Each layer is separated by *weights*, which are the ones that operate with the inputs to obtain the outputs. These operations can be linear and non-linear such as dot product and sigmoid function respectively.

As images in the database might not be "realistic", some process should be done to reduce overfitting and make the model wider. In order to solve this, another model is trained with jitter. Jittering randomly adds data to random images to simulate noise. Also, it randomly shifts some images to avoid uniformity. In consequence, jittering should improve the results of the classifier than if it is not used. However, this method could work when detecting specific object, as it adds profundity and changes spatial information. Yet, when the database are more realistic, jittering can become redundant and be an unnecessary use of training time and processing.

When modeling and training neural networks there is no limitation at all. Infinity models can be made by modifying number of neurons, layers, activation function, input data, output data etc. Then, the wide options to build a classifier make it a limitation, as many models can be trained and there will always be a better combination of parameters to obtain a better performance. In consequence, hardware, time and creativity are the main limits in neural networks.

References

[1] S. Basu, S. Mukhopadhyay, M. Karki, R. DiBiano, S. Ganguly, R. Nemani, and S. Gayaka. Deep neural networks for texture classificationa theoretical analysis. *Neural Networks*, 97:173–182, 2018.

[2] O. V. G. Group. Vgg convolutional neural networks practical, 2017.

[3] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[4] MathWorks. Texture analysis, 2017.

[5] A. K. Muhamad and F. Deravi. Neural networks for the classification of image texture. *Engineering Applications of Artificial Intelligence*, 7(4):381–393, 1994.

[6] Y. Song, Q. Li, D. Feng, J. J. Zou, and W. Cai. Texture image classification with discriminative neural networks. *Computational Visual Media*, 2(4):367–377, 2016.

[7] A. F. R. Vergara. Convolutional neural networks for texture image classification, apr 2018.