Convolutional Neural Networks for Texture Image Classification

Cristina Isabel González Osorio
ci.gonzalez10@uniandes.edu.co

Raúl Santiago Molina Rodríguez
rs.molina10@uniandes.edu.co

Abstract

In the present laboratory report a convolutional neuronal network is trained for the classification of texture images in 25 categories. The architecture of the final network corresponds to 4 convolutional layers and 2 fully-conected layers. In addition, different data augmentation techniques are used. Finally, based on the implementation and training of the network with different parameters, a maximum accuracy of 98% was obtained in the training set and 92.5% in the validation set. In accordance with the above, it is possible to affirm that the model does not over-fit the training data.

1. Introduction

Conventional neural networks consist of arrays of layers of neurons that perform a linear combination of the neurons of the previous layer along with a bias. In this sense, from the backpropagation algorithm, it is possible to estimate the appropriate weights to reduce the error of the network in a specific task. In a convolutional layer, unlike conventional neuronal layers, the neurons are connected to a small region of the anterior layer. According to the above, the output of a convolutional layer corresponds to a volume.

The implementation of convolutional networks for the classification of images is recurrent in the literature. Within this training framework, it is important to select the appropriate parameters to optimize the performance of the network in the training subset without overfitting the model to the data of the same. Thus, it is expected to obtain the smallest difference between the training error and the test error in the same way as in the accuracy of the same. In this way, we must have a generalized distribution and a large amount of data to obtain a sufficiently general.

2. Materials and methods

Texton analysis was made using the *Texture Database* from the Ponce Research Group [1]. This database contains 40 samples of 25 different texture classes, each sample is a grayscale JPEG image of 640x480 pixels. From each image in the set 128x128 patches were randomly sampled. In this sense, the resulting database is composed of 15000 training images (600 images per class), 2500 for validation and testing (100 images per class).

2.1. Convolutional Neural Network

For the network used in this practice, each convolutional layer was followed by MaxPooling of a $2x2$ with a $2x2$ stride, a ReLu and BatchNorm layers, dropout ($p = 0.5$) was used between the last and second-to-last fully connected layers. In order to keep a low number of layers, thus less parameters and less prone to overfitting, the last convolutional layer has a 5X5 stride (every other layer has 1x1 stride). Given that texture images repeat a pattern across the images, it was assumed that activation maps across the image must be similar (not much information is lost increasing the stride). Kernel size, and stride, were fixed across all iterations of the network; the number of channels and fully connected neurons did vary, although it reached high accuracy in the training and validation sets (¿90%), multiple combination of meta-parameters (learning rate, momentum, batch-size, data augmentation) were tried in order to improve accuracy in the test set. An overview of the architecture used is presented in Fig. 7.

Multiple data-augmentation techniques were used, color jitter and image rotation, given that the texture in the image is the same irrespective of the image angle. Both techniques improved our generalization error in the test and validation sets. Different optimizers were used too: Adam and RMSprop, although there was no significant difference between the two.

3. Results and Discussion

The results obtained for the training and validation sets during the training stage are presented below:

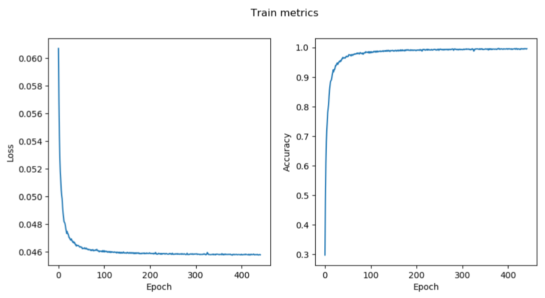

Figure 1: Loss and accuracy results for the training set.

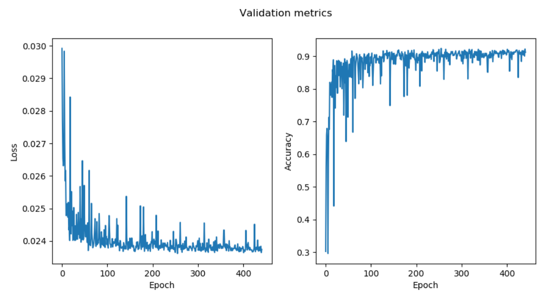

Figure 2: Loss and accuracy results for the validation set.

In order to understand the importance of each of the layers used in the network and the parameters associated with these, the network was trained by eliminating some of the layers. Thus, the figures 3, 4, 5 y 6 show the results of training and validation by eliminating the first and fourth convolutional layers:

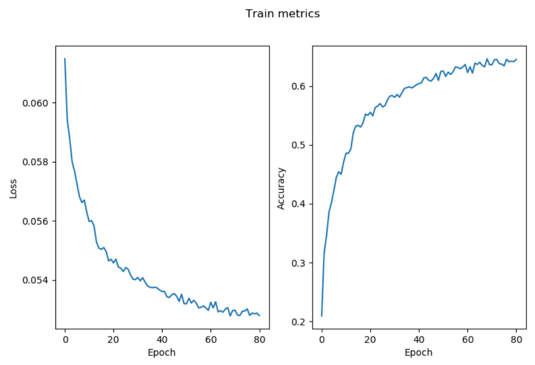

Figure 3: Loss and accuracy results for the training set by eliminating the first convolutional layer.

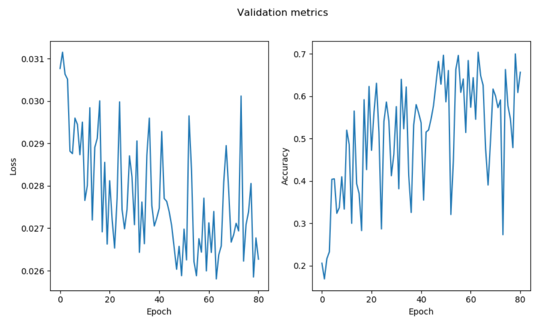

Figure 4: Loss and accuracy results for the validation set by eliminating the first convolutional layer.

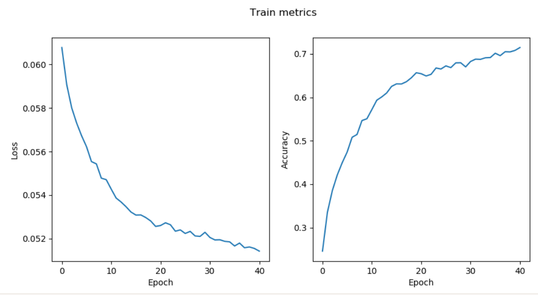

Figure 5: Loss and accuracy results for the training set by eliminating the fourth convolutional layer.

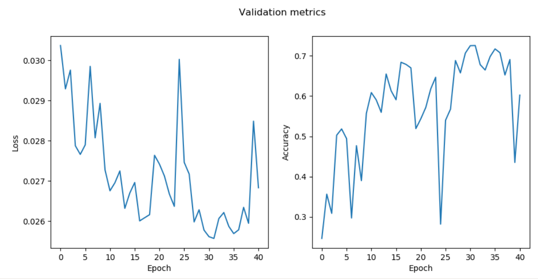

Figure 6: Loss and accuracy results for the validation set by eliminating the fourth convolutional layer.

In agreement with the obtained results, it is possible to affirm that when eliminating the convolutional layers the value of precision in which the performance of the network saturates considerably. This can be understood in terms of the decrease of the parameters of the network, and in turn, the complexity of the model.

4. Conclusions

Convolutional Neural Networks are capable of achieving high accuracy without employing much image preprocessing. Although they do not require much previous knowledge of the problem in order to be useful, finding the right kernel size, number of fully connected neurons or the training meta-parameters is a process of trial and error that takes a significant amount of time.

Bigger and deeper networks are able to fit the training data perfectly but might not generalize the same way, finding the right setting for the network's training might be impractical as each extra layer (or a greater kernel size) adds exponentially more parameters and thus its training consumes more resources and time.

References

[1] 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.

Anexos

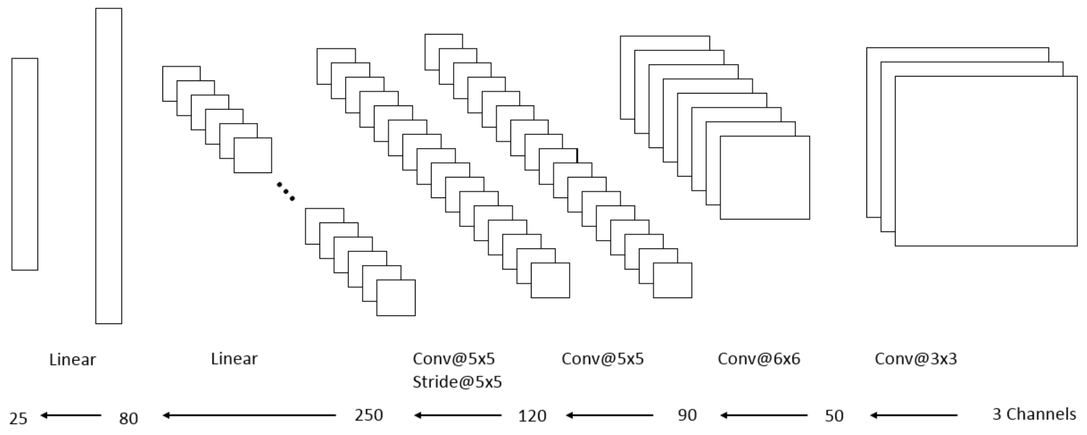

Figure 7: Architecture of the CNN for texture classification