Convolutional Neural Networks For Texture Image Classification

Federico Ulloa
Universidad De Los Andes
Cra. 1 18a-12, Bogotá, Colombia
f.ulloa10@uniandes.edu.co

Esteban Vargas
Universidad De Los Andes
Cra. 1 18a-12, Bogotá, Colombia
e.vargas11@uniandes.edu.co

Abstract

Image classification is one of the main areas in computer vision. In this laboratory, we performed texture image classification. In a previous laboratory, a texton dictionary and SVM approach was made for this task. In this work, we used a convolutional neural network (CNN) for this same purpose. Similar results were obtained with the CNN, but in a way bigger dataset, so we concluded that the performance is better with the convolutional neural network.

1. Introduction

Convolutional neural networks (CNN) are one kind of neural networks. They are also composed by neurons with weights and biases that con be learned over iterations. They receive a single vector (or image) as an input and this is transformed through the hidden layers. The last layer (output layer) is a fully-connected layer, which gives a vector of scores for every category. For example, if a CNN is being used for image classification as in this laboratory, given an input image, the output layer will give a score (probability) of the image belonging to each of the categories, classifying it in the category with the highest score [1].

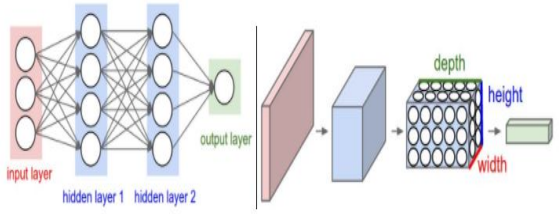

Figure 1: Example of a 3 layer CNN architecture [2]

As shown in Figure 1, neurons are arranged in a three dimensional way. In this example, the height and depth represent the M x N dimensions of the image, while the width would be the channels of the color space.

To build this CNNs, several types of layers exist, apart from the input layer and the output layer (fully-connected or FC). Firstly, a convolutional layer (CONV) computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. A RELU layer is the one that converts all the negative values to zero. A POOL layer performs downsampling of the image[2].

This CNNs are very useful in the computer vision world. In this laboratory, a CNN was used to classify texture images. Previous methods used for this purpose include creating a texton dictionary and using a classifier, as an SVM for instance, to perform this task. As this was the strategy for this same purpose in a previous laboratory, we will compare the results of both methods to determine which gave better results.

2. Materials ans methods

2.1. Dataset

The dataset used in this laboratory is composed of 20000 images, with a size of 128x128 in JPG format. This database was built by taking random patches from the ponce group texture database [3]. The data is divided in train, test and validation sets. The data has 25 classes, 600 of each in the training set, 100 of each in the validation set, and 2500 unlabeled images in the test set. Figure 2 shows a sample of the texture images in the database.

2.2. Convolutional Neural Network

We designed a CNN for the image classification. It is composed of 3 convolutional layers with 192 filters, 300 filters and 400 filters respectively. Then, they are followed by 2 fully connected layers with a softmax.

Batch normalization was also included to reduce the number of epochs in training. The net took 229 epochs to train and each epoch took 0.55 seconds. In other words, the training took approximately 2 hours.

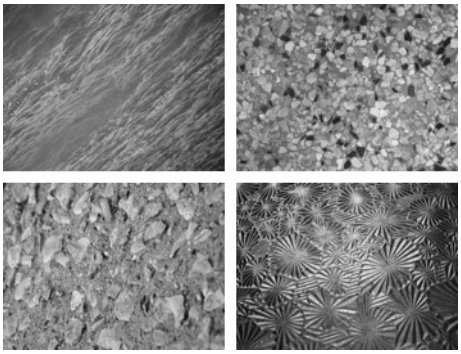

Figure 2: Results using textons and CNNs

3. Results

After training the CNN, the network obtained an ACA of 75.39% in the training set and 71.52% in the validation set.

		ACA
Textons	Training	100%
	Validation	71.20%
CNN	Training	75.39%
	Validation	71.52%

Figure 3: Results using textons and CNNs

Figure 3 shows the results obtained with the texton dictionary and with our CNN. As seen, better results are obtained with textons in the training set, and a similar result for both in the validation set.

It is important to know that the texton results were obtained with a dataset of just 20 images per class, contrary to the 20000 images used in this work database.

Hence, we can conclude that CNN has an overall better performance, with a higher ACA than textons in the validation set and more efficient computationally.

While designing the architecture, we tried to train a CNN from scratch, obtained poor results. Then, we decided to re-train the network used in the previous laboratory challenge. Regarding the architecture, the convolutional layers obtain features with all the filters used, the batch normalization helps the network to learn faster.

Finally, some improvements can be made to improve the CNN. Performing fine tuning in a pre-trained net would make it converge faster. For this, it would be needed to choose which layers to re-trained. Also, using jitter helps as a data augmentation, as it increases the training data. This would be useful as it would improve the performance and reduce overfit.

4. Conclusions

By using a CNN, the performance obtained is better than the one using texton dictionary. Using CNNs is challenging in the design process, as it is a more trial-and-error process, but it normally results in a better performance.

To improve the method, performing fine tuning and using jitter could result in a net with a higher ACA.

References

[1] "An Intuitive Explanation of Convolutional Neural Networks", the data science blog. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. [Accessed: 25- Apr- 2018].

[2] "CS231n Convolutional Neural Networks for Visual Recognition", Cs231n.github.io. [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed: 25- Apr- 2018].

[3] "Ponce Research Group: Datasets", Www-cvr.ai.uiuc.edu, 2018. [Online]. Available: http://www-cvr.ai.uiuc.edu/ponce$_grp/data/.[Accessed : 25 - Apr - 2018]$.