Convolutional Neural Networks for Texture Image Classification

Nicolás Morales Velandia
Universidad de los Andes
Carrera 1 # 19-27
n.morales10@uniandes.edu.co

Oscar Francisco Trujillo Puentes
Universidad de los Andes
Carrera 1 # 19-27
of.trujillo10@uniandes.edu.co

Abstract

There are many methods to classify images. In this way, convolutional neural networks were used in this laboratory to classify images using the texture feature. To fulfill this objective, a CNN was designed with random weights and predetermined weights of AlexNet, obtaining an ACA of 9.6 %. Within the characteristics of the network, it was possible to observe the importance of a ReLu layer, when it is removed the error increases and the ACA falls, the convolutional layers are the most important to extract the characteristics of the same image. Finally, the tool with which the network was developed was pytorch with the help of its documentation.

1. Introduction

In recent years, learning reusable feature representation from large unlabeled datasets has been an area of active research. Additional to this, supervised learning with convolutional networks (CNNs) has huge adoption in computer vision applications. In the context of computer vision, one can leverage the practically unlimited amount of unlabeled images and video to learn good intermediate representation, which can then be used on a variety of supervised learning tasks such as image classification.

This CNN are architectures inspired by biological processes such as the visual system and brain processing. In the last years due to the multiple advantages and benefits that they bring to the different fields of artificial vision, their uses have grown exponentially, given their ability to extract features, data processing and adaptability to different databases. The CNNs from its first implementations demonstrate an excellent performance in results and processing time.

The networks work through an automatic learning, where the acoustic characteristics are extracted, and it is not necessary to define and manual them, which reduces the human failures or complex medelos in the optimal recognition, which is carried out through iterations in try and failure. Based on this, the main objective of this laboratory is to make use of convolutional neural networks for the classification of images based on their texture. [2]

For this, we will proceed to design and train a CNN from an initialization of random weights and pre-trained weights. All this is based on pytorch and the associated GPU. The database used was from the Ponce Group, University of Illinois. The texture database feature 25 texture classes, 40 images samples each. In addition, the entire image are in gray-scale JPG format. [1]

2. Methodology

2.1. Materials

For this laboratory the texture database, were used the **texture database** from The Ponce Group (Computer Vision and Robotics of Beckman Institute, University of Illinois at Urbana-Champaign). The dataset with 20000 images, divided in 15000 train, 2500 validation and 2500 test. It has 25 texture's classes, three classes of bark, three of wood, brick, glass and others; each image was a path of specific material, that each material has a specific texture for each one. All the images are in 128 x 128 pixels, gray-scale and JPG format. [1]

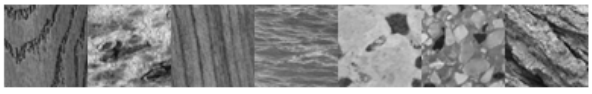

Figure 1: Examples of Texture Database

On other hand, Pytorch is an open source machine learning library for Python for deep learning framework, based on Torch, used for applications such as natural language processing. It is primarily developed by Facebook's artificial-intelligence research group, and probabilistic programming. This allows the use of Tensors and Dynamic neural networks in Python with strong GPU acceleration. [3]

2.2. Methods

In the case of the neural network, an own architecture was created, based on the AlexNet model, starting by creating a 5 layer neural network:

- Convolution
- Convolution
- ReLu
- MaxPooling
- FullyConnected

From this network, changes were made. First, the ReLU was changed by a convolutional layer. Second, two convolutional layers were added after Maxpooling and a new Maxpooling layer was added after the new convolutional layers. Finally, based on AlexNet, a ReLu layer was applied after each convolutional layer.

In the case of reading the data, training the network and testing it in the validation network, the documentation available on the Pytorch website is taken. Within this, the best train model is taken and evaluated. When having a low result, the AlexNet weights were used, in addition, the learning rate and the momentum were reduced in order to increase the acurracy.

3. Results

Next, the results obtained by applying the designed neural network are presented.

# CNN	First Loss	Last Loss	ACA
1	3.1345	0.4658	4.75
2	3.4665	0.8523	3.56
3	2.8163	0.0317	9.68
4	2.7863	0.0289	9.56

Table 1: CNNs with random weights

# CNN	First Loss	Last Loss	ACA
1	3.4120	0.3122	5.65
2	3.3586	0.7869	4.53
3	2.7156	0.4257	9.88
4	2.8476	0.4786	9.68

Table 2: CNNs with AlexNet weights

Within the results obtained, the neural network with random weights takes between 4 and 5 minutes, in the case of using the AlexNet weights, it takes between 42 to 45 minutes.

4. Discussion

Initially, you can see the difference in the processing times, where when assigning AlexNet weights, the network will take longer, this is because the network must learn and recalculate the same weights. The error gradients are recalculated with respect to all weights in the network and the filter / weight values and parameter values are updated to minimize the output error. Allowing to know what the network has learned to correctly classify this particular image by adjusting its weights / filters so that the exit error is reduced.

In rows 3 and 4 of both tables, a Jitter was applied which allowed to change the brightness, saturation, contrast and tone, you can see the effect where you have lower errors, however, the validation does not take values of ACA very high On the other hand, when implementing Jitter in the development of the network, the performance of the classification increases if the images come to present large differences in contrast or saturation.

For the study of the effect of different layers on the neural network, a procedure is performed in which layers were removed and new ones were added. They were specifically removed, a layer of ReLU, the removal of this gender layer that the system would increase its error and the ACA would remain lower than the model tested before. It can be noticed that networks do not learn, since the error does not fall. However, when using the AlexNet model, where a ReLu was applied at the end of each convolutional layer, no significant or relevant difference was noted in the case of the ACA, the error decreased, but this could be associated with overfitting.

When training the network with 24 periods, it was observed that the error associated with both training and validation decreased as times increased. Result that is reflected in a better performance for the classification using the network with more eras. However, in the designed network it is possible to reduce the error in a considerable amount within 24 periods.

Finally, in the design of the architecture, the most relevant thing is to properly associate the outputs and inputs of each layer. These values have to be correctly established, since the network does not train if they do not match. One of the most relevant parameters are the sizes of filters, if these are not taken properly we have a filter larger than the image or that has no great effect on it, for that reason it was necessary to test the sizes several times, until finding suitable values for the network to work. On the other hand, it was observed that the order in which the convolutional, ReLU and pool layers are placed significantly affects the result. Where there are sequences in which the network did not train.

5. Conclusions

In conclusion, as can be seen in the results, it was not possible to build a suitable classifier for textures, using CNN. This is observed because the ACA obtained from the test database is very low, less than 10% for the 4 designed CNNs. However, as can be seen in Table 1 and Table 2, the loss of the model after finishing the 24 periods decreases for all CNNs. Which can be related that the classification has a overfitting in the data.

Additionally, because the ACA opted is low, it can be observed that the network is not trained with the data. In this way, this may mean that the network has a large number of hidden units in the fully connected layer. Which may explain the overfitting in the data. For these reasons, reducing the number of hidden units in the fully connected layers can reduce overfitting. Also, increasing the number of periods in the model to reduce the lost even more in the model, due it is observed that increasing the number of periods, decreasing the loss in the model. Finally, use a different ReLu ratio during the training of the model, because this parameter have a significantly effect in the performance in the model. Need to find a Relu ration that better adapted to the database used.

6. Code

The way to test the network, is to download the code redes_oscarṗy, finetuningṗy and train_testṗy. The codes with the name of finetuning for pre-trained weights or train_test for new weights are run

References

[1] Ponce research group: Datasets. =http://www-cvr.ai.uiuc.edu/ponce$_grp/data/$, 2006.

[2] 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.

[3] S. Yegulalp. Facebook brings gpu-powered machine learning to python. =https://www.infoworld.com/article/3159120/artificial-intelligence/facebook-brings-gpu-powered-machine-learning-to-python.html, 2018.