CNN for Texture Image Classification - Lab 11 - Computer Vision

Jos Yesith Juez
University of the Andes
jy.juez@uniandes.edu.co

Nicolas Florez
University of the Andes
n.florez228@uniandes.edu.co

1. Introduction

The CNN are composed of neurons that have learnable weights and Biases. Each neuron receives inputs, performs a point product and then optionally follows a non-linearity [1]. A CNN is comprised of many convolutional layers (one or more), and at the end of each layer performs pooling. The last convolutional layer are joined with fully connected layers [2].

A very powerful tool that allows us to work with convolutional neural networks is the Pytorch library.This is a Python package that provides two high-level features: Tensor computation with strong GPU acceleration and Deep neural networks built on a tape-based autograd system [3].

Alex Krizhebsky developed a very deep convolutional neural network with the following architecture [4]:

- Seven hidden layers without counting some max pooling layers.
- The first layers were convolutional.
- The last two layers were fully connected.
- The activations functions were of type Rectified Linear Units (RELU) in every hidden layer.
- Used dropout to regularize the weights in the fully connected layer.

In this laboratory, a texture image classifier will be developed using a CNN with the architecture of AlexNet as the baseline [5]. The dataset is composed by 20000 images of 128x128 patches (15000 train - 2500 val - 2500 test), each with their corresponding labels. The Database have 25 Class that corresponding each to textures different.

2. Materials and methods

2.1. Description of the Network

A CNN with 5 convolutional layers and 3 fully connected layers was implemented. In Each convolutional layer was applied a nonlinearity (RELU) to the output.The last fully-connected layer is plugged in to a 25-way softmax which create a distribution over the 25 class labels. In order to convolutional layers 1, 2 and 5 a max pooling was applied in the output of each RELU.

The first convolutional layer filters the 128x128x3 input image with 32 kernels of size 11x11x3 with a stride of 4 pixels. The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 96 kernels of size 5x5x32.The third last convolutional layers were connected to one another without to be pooled neither normalized. The third convolutional layer has 192 kernels of size 3x3x96 connected to the (normalized and pooled) outputs of the second convolutional layer. The fourth convolutional layer has 128 kernels of size 3x3x192 , and the fifth convolutional layer has 128 kernels of size 3x3x128. The fully-connected layers have 2048 neurons each.

To reduce overfitting we use a technique called dropout (an efficient way to combining different model predictions [4]), in the input of the two first fully connected layers. This technique zeroes half the entries of a layer randomly. The classification problem was optimized using minibatchs; In total 100 for the training stage and 50 in evaluation stage. The cost function that was used was cross type Entropy.

The number of epochs that were used to carry out training and validation of the network was 50 in total, but a conditional was used in the evaluation stage where after 30 epochs we will ask in each epoch (30 to 50) if we want continue training the network or not. This with the aim of obtaining the highest ACA in the network.If in question one answers '0', the training of the network stops and the ACA that was obtained with that number of times will be saved.

3. Results

The results obtained of the Classifier of Texture Image was:

In the Training stage:

- Accuracy = 97,86 %
- Loss = 0,068
- Epochs Number = 34

In the Validation Stage:

- Accuracy = 86,64 %
- Loss = 0,577
- Epochs Number = 34

An experiment was made: We eliminate the penultimate fully connected layer of CNN implemented and the following results were obtained:

In the Training stage:

- Accuracy = 97,61 %
- Loss = 0,080
- Epochs Number = 33

In the Validation Stage:

- Accuracy = 86,64 %
- Loss = 0,628
- Epochs Number = 33

It is noted that by eliminating the penultimate layer of CNN the Accuracy for both training and testing are very similar.

Then, We eliminated the convolutional layer 3, obtaining the following results:

In the Training stage:

- Accuracy = 96,91 %
- Loss = 0,095
- Epochs Number = 32

In the Validation Stage:

- Accuracy = 85%
- Loss = 0,649
- Epochs Number = 32

For these cases, no significant changes were obtained in the performance of the CNN. This can happen because the response of the filters in the first two layers is already very fine, that is, there are already significant differences in each of the filters and with details very well defined.

4. Discussion

One of the great difficulties in the implementation of the image classifier was the adaptation of the data to Alexnet's network. This was because there was no clear knowledge of how to upload the training and validation images with Pytorch.

Initially we tried to use a cycle where we tried to read each of the images one by one, but there were many problems to obtain all the data with each of its corresponding labels. At the end we used a package that included the Torchvision library: "datasets.ImageFolder", which allowed us to load all the images correctly.

Another difficulty was in the adaptation of the kernels and the inputs/outputs of each of the convolutional layers. This is because the baseline of AlexNet was designed for images of 256x256 patches, and the images that were used were 128x128 patches.

Finally, in order to obtain the highest performance it was necessary to use a conditional that would allow us to control the number of times to train the network, since in some cases if it exceeded a number of times the Accuracy began to decrease.

5. Conclusions

It is observed that when using a CNN a very high performance is obtained, without having to use a representation of the image before performing the classification algorithm. To obtain this performance, we must take into account optimization methods (dropout and minibatch).It is important to have into account the number of epochs for network training, since for a certain epoch the maximum Accuracy will be obtained, decreasing then in the following.

References

[1] CS231n Convolutional Neural Networks for Visual Recognition. Available in: http://cs231n.github.io/convolutional-networks/

[2] Convolutional Neural Network. UFLDL Tutorial: Available in: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[3] Pytorch. Available in: http://pytorch.org/about/

[4] P. Arbelez, "Computer Vision: Recognition 05", University of the Andes, 2018

[5] Pytorch/Vision Github. Available in: https://github.com/pytorch/vision/blob/master/torchvision/models/alexnet.py