Convolutional Neural Networks for Image Classification

Mariajosé Serna Ayala
Universidad de los Andes
m.serna10@uniandes.edu.co

Henry Daniel Torres Acuña
Universidad de los Andes
hd.torres11@uniandes.edu.co

Abstract

The image classification task has been one of the main problems to be addressed in computer vision. Since the appearance of Neural Networks in computer vision with the Convolutional Neural Networks (CNN) which have proven to be the best method to classify images. In this work we train different CNN with different architectures to classify images with different textures form the dataset of the Ponce Group. Our Network didn't classify efectively the Textures dataset, we got an error of 0.794

1.. Introduction

Artificial Neural Networks (ANN) is a computational model, used in machine learning, based on the functioning of the nervous system. This model propose the use of units, neurons, that are connected with others. Therefore the response of a neuron is going to be the summation of the activities of the other neurons connected to it, and the response of an activation function. These ANN are normally used for regression (prediction of a value depending of an input value) and classification (assigning a label to a input value) tasks. During the training phase this model is suppose to learn the weights that minimize the error. This learning is done by Back propagation which by the error in the last layer, the value predicted by the ANN and the real value, the errors in the previous layers can be computed respect to the weights. And therefore these can be modified in order to improve the result.

Convolutional Neural Networks or ConvNets (CNN) are ANN but their functioning are inspired in the visual cortex of the brain. This means that are used for the classification and detection of objects in images. These also are made from units with learnable weights and biases, that are the representation of neurons. These are composed of the same layers of an ANN but have some extra to make easier this task. The layers are: Convolutional layers, pooling layers, normalization layers (ReLu) and fully connected layers.

The convolutional layers correspond to filters with the learnable weights. These filters are convolve with the input image, and a response map is produce. The pooling layers function is the reduce the size of the response maps. This is done by selecting a window of an arbitrary size and passing it through the response map and selecting the maximum value inside that window, and all those maximums are the ones that are going to pass to the next layer. The Normalization or Rectified Linear Units layers are used to avoid having negative values after the convolution with the different filters in the convolutional layers. This is done with the activation function which wherever a negative number occurs, it swap it out for a 0. The fully connected layer have full connections to all units in the previous layer. This means that each response from a map in the previous layer is arranged in a single vector.

During the set of experiments developed in this document the purpose was to classified images in different datasets. For this task of classification is necessary a final layer: Softmax. This is the classifier, and takes the "votes", or values, from the fully connected layer to guess a correct label to the input image. Softmax classifiers give you probabilities for each class label. This is done by a function f that takes an input set of data x (fully connected layer units) and maps them to the output class labels via a simple (linear) dot product of the data x and a weight matrix W.

The aim of this laboratory was to train a CNN based on *VGG Convolutional Neural Networks Practical* from the Oxford Visual Geometry Group, which use functions from the MatConvNet: CNNs for MATLAB library [2], for image classification in the the textures dataset provided by the Ponce Group, Computer Vision and Robotics from Beckman Institute, University of Illinois at Urbana-Champaign [5].

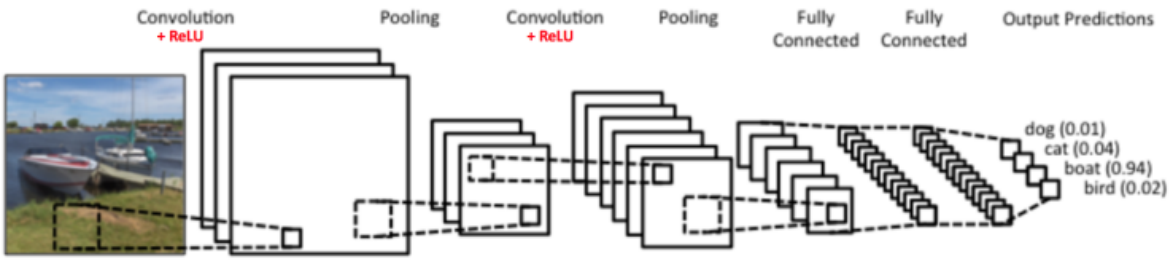

Figure 1: Visualization of all types of layers in a Convolutional Neural Network (CNN): ReLu, max pooling, and fully connected. Retrieved from: `https://www.clarifai.com/technology`

2.. Materials and Methods

The design of a CNN consists of first a change of some hyperparameters with th purpose of getting a lower error in the classification. This design can be in the number and type of layers inside the CNN, you can choose how mane layers are you going to use, of which type and in which order.

Our initial architecture was a CNN with 5 layers : A convolutional layer, a max pooling layer, another convolutional layer, with anothers respective max pooling, and the solfmax classifier. Between the parameters that can be modified, others are the learning rate, which specifies how much does a weight change during the backpropagation algorithm. And, the number of epochs, this is the number of times that the CNN is going to applied the backpropagation algorithm in order to modified the weights.

3.. Results

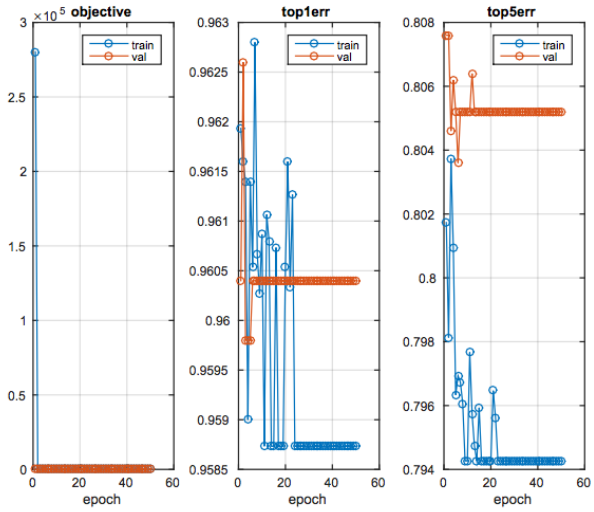

Figure 2: Results for a 5 layer CNN with a learning rate of= and 60 epochs.

4.. Discusion

The learning effectiveness of a convolutional neural network mostly depends on the depth of the network and the number of epochs during the train phase. Regarding this, one of the main challenges we faces during the process of building the neural network was designing the architecture itself. The try-and-fail approach is a long and inefficient process that requires a lot of time and computational resources. Since these resources were generous but restricted to a frame time, the intelligent use of the cluster's machines was another big challenge. It was also possible to work on our personal computers, but computational power isn't enough and the process of installing all the libraries and software required is really difficult and not always worked entirely. Finally, as the results show we didn't

make the CNN to work properly but we did try a bunch of different configurations. These was also a problem because testing and change in the architecture requires to re=initiate all the process from the beginning and wait for it to finish, which can take a really long time.

During the design process we try adding and deleting different layer types at various stages of the network. Since our CNN didn't worked we couldn't see major changes from these process. However, theoretically, deleting layers closer to the final will have a more significant impact on the performance of the network than doing changes in the early ones. Arguably this could be explained taking in count that the final layers have more specialized type of filters to recognize the class of the given image and making changes of deleting those will lead to an incomplete network unable to tell apart the characteristics of a new image.

5.. Conclusions

The design, train and test of a working CNN is a time-consuming process that not always gives the best results. Regarding this, the tunning of the principal parameters requires a lot of time and computational resources. The main parameters we found were the size of the input image (changed with the (jitter()) function), the learning rate, the number of epochs, the architecture itself and the the input size of each layer. Finally, since our network didn't classify quite well the textures we need further efforts in the design of our architecture.

References

[1] R. Szeliski. Computer Vision: Algorithms and Applications. Chapter 14:Recognition. 2011

[2] A. Vedaldi and K. Lenc, Proc. "MatConvNet - Convolutional Neural Networks for MATLAB". of the ACM Int. Conf. on Multimedia, 2015.

[3] A. Vedaldi and B. Fulkerson, "VLFeat : An Open and Portable Library of Computer Vision Algorithms",2008, http://www.vlfeat.org/

[4] A. Vedaldi, A. Zisserman . VGG Practical. Robots.ox.ac.uk. Retrieved 10 May 2017, from http://www.robots.ox.ac.uk/~vgg/practicals/cnn/index.html#part-4-learning-a-character-cnn

[5] S. Lazebnik, C. Schmid, and J. Ponce. A Sparse Texture Representation Using Local Affine Regions. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 8, pp. 1265-1278, August 2005.

[6] CS231n Convolutional Neural Networks for Visual Recognition. Retrieved 11 May 2017, from http://cs231n.github.io/convolutional-networks/#fc

[7] A. Brohrer. How do Convolutional Neural Networks work?. Brohrer.github.io. Retrieved 10 May 2017, from http://brohrer.github.io/how_convolutional_neural_networks_work.html