Indian Journal of Science and Technology, Vol 11(31), DOI: 10.17485/ijst/2018/v11i31/130149, August 2018
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Analysis of the Layers in Convolutional Neural Network in the Context of Text Recognition

Mauricio Vladimir Peña[1], Diego Mauricio Rivera[2], Carol Rodríguez[2], Ricardo Ramirez[3] and Victor Grisales[3]

[1]Universidad Libre Colombia, Cl. 8 #580, Bogotá, Colombia; mauriciov.penag@unilibre.edu.co,
[2]Universidad Pedagógica Nacional Colombia. 72 #11-86, Bogotá, Colombia; dmrivera@pedagogica.edu.co, cirodriguezf@pedagogica.edu.co
[3]Universidad Nacional de Colombia, Bogotá, D.C., Cundinamarca, Colombia; reramirezh@unal.edu.co vhgrisalesp@unal.edu.co

Abstract

Objectives: To analyze the layers in Convolutional Neural Network in the context of text recognition looking for interpretations. **Methods/Analysis:** Through the training of a deep Convolutional Neural Network and its application to the recognition of numerical characters from the MNIST dataset, the characteristics of deep architectures are studied and analyzed. Making a detailed study of the behavior of the different weights and their significance through the training of the network using - images, error values and gradient values which characterize each of the layers. **Findings:** After the training it is observed that the convolution layers have a possible interpretation. Results were obtained from the images of the MNIST dataset after going through the convolution layers with images and random filters. However, the most representative results are achieved by viewing a single image using random filters. **Improvement:** Recommendations for design and implementation based on the example and other references are presented.

Keywords: Artificial Neural Network, Convolutional Neural Network, Text recognition

1. Introduction

In recent years, models of Convolutional Neural Networks have applied different types of deep network arrays to make different types of classification using different techniques: Designing and training a complete deep network with a set of specific inputs to the classification, taking an architecture already trained with available standards for different research groups in deep learning and retraining the network according to the new set of images, starting from the assumption that the pre-trained weights are approximate to those that must be obtained with the retraining or final tuning.

Next, related applications on deep convolutional neural network architectures are described: cifarNet, AlexNet and GoogLeNet[1]. on the use of AlexNet[2] it uses as input the set of ImageNet images, they use five convolution layers with activation layers type Rectified Linear Unit (ReLU) and with pooling and a set of 3 completely connected layers being the last of 1000 units according to the number of classes, this arrangement completes a total of 60 million parameters and 650000 neurons, and test errors of the order of 15% are obtained. About the use of VGG16 and VGG19[3] use fixed size RGB images of 224 × 224 as inputs for training and a single preprocessing is the subtraction of the RGB average of each pixel. The inputs are passed through convolution layers where the filters are 3×3 receptive fields, the filter stride equals 1 and the padding to complete the initial resolution is 1 pixel and the max- pooling are developed on 2x2 pixel and 2 stride windows. After the convolution layers come the FC layers, the first two with 4096 channels each, followed by the classification layer with 1000 classes and the activation layer type Softmax. All layers are provided with ReLU type non-linear activation stages. In[3]

*Author for correspondence

they make a comparison of architectures by their depth, starting with 11 layers (8 layers of convolution and 3 FC) until reaching 16 and 19 layers in configurations of groups of convolution layers, separated by layers of max-pooling, thus: 2-2-3-3-3 and 2-2-4-4-4 respectively, connected at the end with the FC layers. These configurations reach several trainable parameters of approximately 138 and 144 million, the test error on the ImageNet data set is approximately 7 - 8%. In relation to GoogleNet, in[4] they construct a new convolutional neural network configuration, using a depth 2 architecture called «Inception», which follows a concatenation of the outputs of a subsampling or pooling operation with operations outputs of convolution with 1x1, 3x3 and 5x5 filters. The results obtained with this type of network are of the order of 6-7% margin of error. On the LeNet-5 convolution network, its initial version was composed of 7 layers with image entries of 32x32 pixels normalized to -0.1 (for white background) and 1.175 (for a black foreground) to reach an average of 0 and variance of 1 and in this way accelerate learning. A simplified version of this type of network designed and calculated in[5], which is made up of two layers, each with a convolution operation, a pooling operation and an activation layer; additional these is a hidden layer of a fully connected neural network and a Softmax type logistic classifier.

2. Materials and Methods

In the machine learning algorithms, it is essential to obtain a representation of the data that extracts the desirable characteristics according to a defined task, so it is necessary to pre-process the input information with specific knowledge, in such a way that the algorithm is limited to a certain type of data. However, in artificial intelligence it is desired that algorithms learn the world as it is. In deep learning it is possible to obtain representations of the environment that can go from the general to the particular as the level of depth increases. Deep architectures could be a series of non-linear machine learning transformations whose outputs feed the next layer to a final layer. These deep learning algorithms find their best representative in neural network arrays that can be presented in multiple layers and are called Deep Neural Networks[6].

2.1 Software

For the execution of the mathematical calculation an initial analysis of the type of software to be used for training and the application of the network was proposed. Among the software packages most recognized and recommended by the scientific community are Caffe, Torch, TensorFlow and Theano. Table 1 shows 4 of the main packages used in deep learning networks, these have support for their use in CUDA cores typical of the NVIDIA brand graphics cards, used to work in parallel tasks on different platforms and with libraries destined to convolutional neural networks with the advantage of being open source.

Theano software was chosen because the documentation and the community efficiently support the package, allowing going from the most basic of neural networks to deep neural networks in a guided way. Theano software is a symbolic calculation library focused on operations that are immersed in calculations with neural networks. Among its outstanding capabilities are the backward differentiations or propagation of the error and the use of parallelization on graphic cards[7].

2.1.1 Dataset

The scientific community has created datasets of standard images with which different types of deep learning

Table 1. Comparative table of software tools for deep networks

Software	Creator	Interface	Pre-trained models	RBM/ DBN
Caffe	U Berkeley and community	C++, command prompt, Matlab	yes	no
TensorFlow	Google	C/C++ y Python	no	yes
Torch	Ronan Collabert y others	C/C++ y Lua	yes	yes
Theano	U. Montreal	Python	yes, with Lasagnne	yes

networks have been measured and compared. MNIST[8] is a database of handwritten digits that has 60000 training examples and 10000 standard and dimensioned test samples. Cifar10 and Cifar100 are datasets of color images with 60000 training examples and 10,000 test examples with 10 and 100 classes respectively, with image size of 32×32[9], ImageNet is a dataset of images organized according to the hierarchy given by WordNet (English lexicon database of names, verbs, adjectives and adverbs grouped into knowledge synonyms that express concepts) which has an average of 1000 images per set that represents a concept[10].

2.1.2 Training

The training process was executed twice, using as compute devices a CPU (Processor i7-5930K CPU @ 3.50GHz x 12) and a GPU (GeForce GTX 980 Ti / PCIe / SSE2). The total simulation time was approximately 24 hours and 18 minutes respectively with the same number of iterations and equal values of performance measures. Then, the results of the training, validation and test errors are observed at the end of the 250 epoch of training period of the network:

Training @ iter= 24900, epoch 250, minibatch 100/100,

Training error 0.000000 % validation error 0.930000%

Test error 0.900000 % Best validation score of 0.920000%

Obtained at iteration 22100, with test performance 0.900000%

The code for file CNN_MLP.py ran for 17.27m

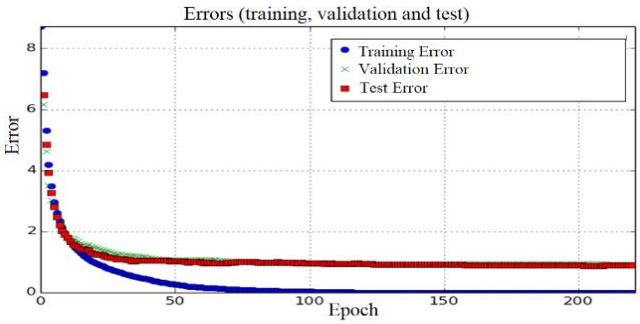


Figure 1. Training, validation and test errors.

The behaviors of the different types of error are presented in Figure 1. It is observed that after 50 epochs the errors reach a stable value close to zero, the training error reaches a value of zero, but the validation and test errors they remain at 0.9 and do not decrease to zero. However, in Figure 2 it is observed that the weights of the different layers continue to vary and even the weights of the convolution layer 1 still have the potential to continue decreasing the test and validation error. Figure 2 shows that the second convolution layer is the most susceptible to learning, followed by the third layer, that is, the hidden layer FC. The first and last layer does not have significant changes in the gradient, as do the gradient values for the bias.

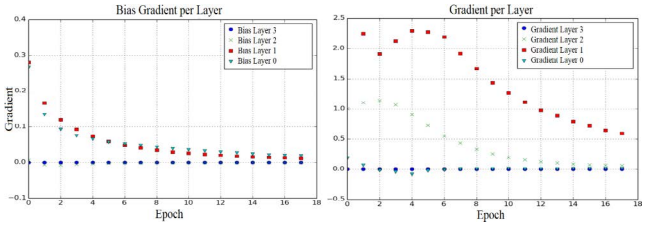


Figure 2. Gradient behavior per layer, left gradient of bias, right gradient of weights.

3. Results and Discussion

3.1 Interpretation of Intermediate Layers

After the training it is observed that the convolution layers have a possible interpretation. For this exercise, results were obtained from the images of the MNIST dataset after going through the convolution layers with images and random filters. However, the most representative results are achieved by viewing a single image using random filters as presented below.

First Layer: The «number four» written by hand is one of the numbers with more features (edges, corners, and crosses, horizontal, vertical and diagonal lines). When operating the convolution layer on this 28 × 28 image, 12 × 12 images are obtained. In Figure 3 it is observed that this first layer makes different edge detection for each filter and additionally reduces the resolution of the image.

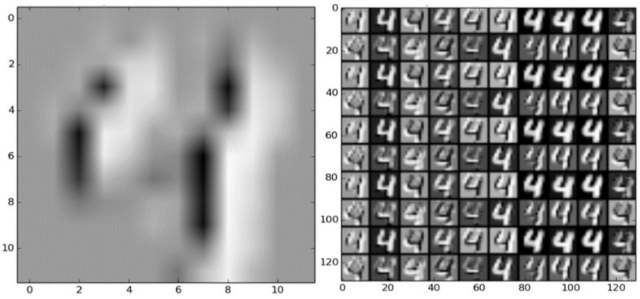
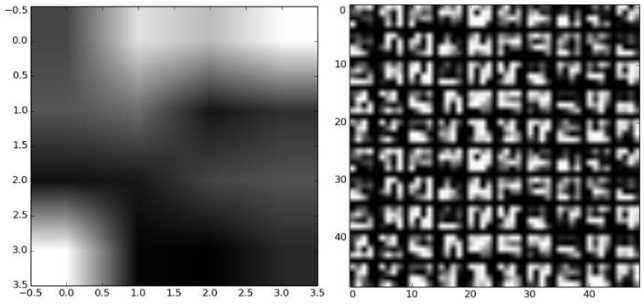


Figure 3. Images obtained after the first convolution layer. Left: simple image, right: mosaic with different and random filters.

Second Layer: As the results of the first layer pass through the second convolution, it can be seen in Figure 4a that the image is a partial representation of the input. From Figure 4b it is evident that each image represents a part rather than a characteristic, and they differ because the filters have elements of different values.



a b

Figure 4. Images obtained after the second convolution layer. Left: simple image, right: mosaic with different and random filters.

Third Layer (Hidden Layer): In this layer the information that reaches each neuron of the input of 28 × 28 = 784 pixels is only 4 × 4 = 16 pixels that represents a section or characteristic of the original image but receives another 999 images that have a different representation of the entry. In Figure 5 several possible representations of the neuron for the entrance to this layer are observed. However, there is no evidence of a pattern that allows to assert that this layer has a physical sense or characteristic of the original image. Finally, 10 entries are randomly selected, and the convolutional neural network is executed, the results are 100% effective in the classification process of the images of the MNIST database.

From the previous section it is possible to suppose ideas that enrich the knowledge of neuronal networks of convolution but that are not conclusive, and it is necessary to deepen their study. The random initialization of the weights of the first layer is sufficient to obtain important characteristics of the entrance, one could train for 10 or 20 times or not to train this layer, the gradient shows that the weights obtained imperceptible variations. This first layer acts as an extractor of features of the input image even without training it. The second layer, on the other hand, has the highest learning rates and this may be due to the large number of characteristics extracted from it, the goodness of the Backpropagation algorithm, since there is no attenuation of the error in this layer, as in the case of the hidden layer, the gradient in the last layer is close to zero, which may be because the outputs are only 10 possible classifications taking into account a much greater number of characteristics of the image.

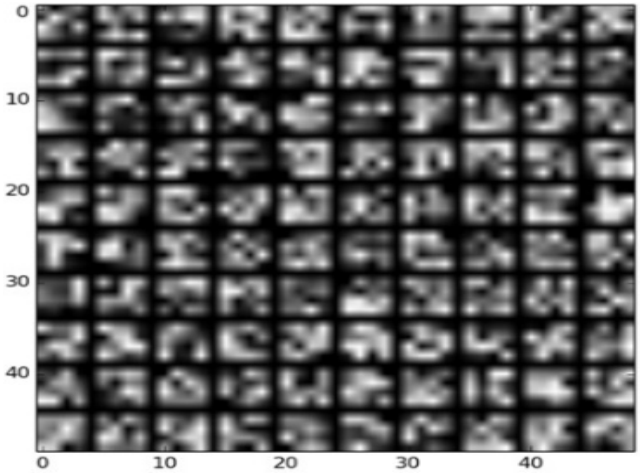


Figure 5. Mosaic of possible representations of a neuron of the hidden layer for an input image (number "four" image).

Some authors assure that in the algorithm of «Backpropagation» the layers of lower level are less trained by the attenuation of the error, the present work does not allow to make this analysis of forceful way. The intermediate layers of convolution and complete connection present learning rates that suggest that these are the layers that learn the task of extracting characteristics and classification. From the analysis of convolution layers (convolution, pooling, activation) it is possible to affirm that pure convolution has the effect of extracting different types of characteristics locally depending on the filter used to form a new image that highlights the type of feature extracted from the input. The pooling layer compresses the image creating a compact representation of the characteristic given by the type of filter and the activation can highlight the characteristic depending on the activation function. From the second convolution layer it must be affirmed that the resulting images are local views of the original image, in the first layer the general forms are extracted accentuating characteristics such as edges and types of lines. In the second layer each filter can detect sub characteristics of the images resulting from the first layer. When analyzing the representation of the hidden layer it was not possible to find something that indicates that each neuron has a specific task, the original size of the image is an impediment due to its low size; addition-

ally the convolution and pooling filters reduce more than 80% the resolution of the image in the hidden layer.

4. Conclusions

The role of convolution layers strictly speaking is to detect relationships of existing characteristics in previous layers, while the role of the subsampling layer or pooling is to make the semantic union of similar characteristics in a characteristic. It is important for the understanding of concepts of neural networks to use specialized software and to make different types of visualizations for each layer. This would allow deepening and building new architectures specific to certain tasks.

As future work it is possible to implement the concepts of GPU, ReLU activation, and Dropout regularization, also perform the analysis of convolution layer using the concept of Autoencoder to verify the interpretation of convolution layers and complete connection, perhaps allow to pretrain layers and obtain results in less time and less computational cost. Additionally, it is possible to use larger images and natural environments in convolutional neural networks at different network depths.

5. References

1. Shin HC, Roth HR, Gao M, Lu L, Xu Z, Nogues I, Yao J, Mollura D, Summers RM. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning, IEEE Transactions on Medical Imaging. 2016; 35(5):1285–98. https://doi.org/10.1109/TMI.2016.2528162. PMid: 26886976, PMCid: PMC4890616
2. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, 2012. p. 1097–105.
3. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014.
4. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR), 2015. p. 1-9. https://doi.org/10.1109/CVPR.2015.7298594.
5. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition, Proceeding of the IEEE. 1998; 86(11):2278–324. https://doi.org/10.1109/5.726791.
6. Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives, IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013; 35(8):1798–828. https://doi.org/10.1109/TPAMI.2013.50. PMid: 23787338.
7. Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. aarXiv:1605.02688. 2016.
8. LeCun Y, Cortes C. MNIST handwritten digit database. AT&T Labs, 2010. p. 2.
9. Krizhevsky A. Learning multiple layers of features from tiny images, 2009. p. 1–60.
10. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks, Proceedings of the 25th International Conference on Neural Information Processing Systems. 2012; 1:1097–105.
11. LeCun Y, Bengio Y, Hinton G. Deep learning, Nature. 2015; 521(7553):436–44. https://doi.org/10.1038/nature14539. PMid: 26017442.