Convolutional Neural Networks for Image Classification

Laura Bravo
Universidad de los Andes
Cra 1 #18a-12, Bogotá, Colombia
lm.bravo10@uniandes.edu.co

Catalina Gómez
Universidad de los Andes
Cra 1 #18a-12, Bogotá, Colombia
c.gomez10@uniandes.edu.co

Juan Pérez
Universidad de los Andes
Cra 1 #18a-12, Bogotá, Colombia
jc.perez13@uniandes.edu.co

Abstract

Classification of images is one of the fundamental tasks in Computer Vision, which consists of assigning a label to a given image depending on its content. Regarding the present work, images depicting some texture were assigned to a class of images depicting a similar texture. The dataset in which our experiments were run was obtained from the Ponce group, and it presents many orientations for 25 different textures. For the actual classification, we experimented with various architectures for convolutional neural networks by adding and removing convolutional, ReLU and pooling layers by trial and error. The implementation of these structures was MatConvNet, from the VLFeat library. The error of the networks was measured in two ways: by only considering the first prediction of the network, and by considering the first five predictions of the network. While the error of the best performing network was 0.192 when only taking into account the first prediction, that error drops to 0.0172 when considering the first five predictions.

1. Introduction

Classification of images consists on assigning a label to an image that reflects its contents, usually a certain type of object, for instance, a dog. It is one of the fundamental problems in Computer Vision, since it is the basis for *understanding* an environment once individual objects have been detected. It has been widely studied by many research groups across the world, and very important advances have been made in the last years [1].

As many problems in Computer Vision, the traditional approach, given an image, is to extract some features that characterize it and compare them to previously known features of other images whose class is known. According to this information, a label is assigned to the image. Here the usual pipeline is clear: representation and then classification based on it.

More recently, convolutional neural networks have become a very useful tool in Computer Vision, since they aim at merging the two traditional steps into a single one, that is, to not only learn what differentiates classes based on a given representation, but also learn the representation that potentially establishes clear differences while learning how to use this representation. Convolutional neural networks are constructions based on layers. Some layers are composed of fundamental units called neurons, which represent a linear combination of their input and can model convolution, these layers are called, therefore, convolutional layers. Convolutional networks also have other types of layers that perform operations different from convolution such as *ReLU* layers, which apply a ramp function (a non linear operation), *pooling* layers, which perform pooling, *dropout* layers, which exclude the output of neurons from a given layer and, finally, aitloss function layer, which determines the function that will be optimized [2]. It is worthy to note that this approach has had outstanding performances in a wide variety of tasks in recent years, including classification [3].

A rather simple exercise is to classify images depicting textures. When studying texture, one finds that it is not easily described, unlike colors, which can be represented by combinations of numbers. The most widely used method for dealing with texture is that of *textons*, which involves the definition of a bank of filters with which the images are convoluted, and then a classification methodology is applied to the output of these convolutions. The first layers of convolutional neural networks could be regarded, therefore, as an enhanced version of this method, in which the library of filters is not predefined, but learned simultaneously with the classification method for the output of this filtering. Addi-

tionally, these networks provide other ways of processing the data in the image, so that the performance of the whole method has room to improve dramatically when compared to the usual classification based on some representation

Here, the task was to classify these images in one of 25 possible classes with convolutional neural networks. The implementation here used is *MatConvNet* which is a MATLAB toolbox implementing Convolutional Neural Networks (CNNs) for computer vision applications [4]. This being our first time using this strategy, we experimented with architectures that differed in the amount of convolutional, ReLU and pooling layers, while searching to improve the network's performance.

2. Materials and methods

The implemented code was taken from VGG Convolutional Neural Networks Practical that uses VLFeat with its extension for CNNs MatConvNet. The network architecture was designed taking into account the provided function `initializeCharacterCNN`, which creates a structure with one field that correspond to the layers of the network. Each layer was modified according to the size of the images from the dataset (128×128 pixels), and the resizing to 64x64 images. The convolutional layers required the definition of kernel size, filters depth, and number of outputs. The filtering operation reduces the size of the original input, so in order to preserve the original size zero padding must be included. The bias for each output must be added, and the output can be downsampled with the stride option.

The non-linearity operator of the network is included after the linear filtering as a non-linear activation function (ReLU). Pooling was another operator included in the network, to group nearby features by specifying the size of the window and downsampling the size. A common pooling choice is the max operator. The last layer was a softmax for the classification task, since it allows a soft assignment of the output to various categories, by producing a probability distribution function over the 25 possible categories, whose entropy is to be minimized.

The final design of the network was established by trial and error, and the parameters of each layer were chosen based on dimension adjustments and previous works. Figure 1 represents a schematic of the network's architecture.

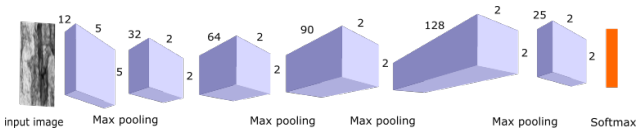

Figure 1. Schematic of our network's architecture. It consists of 6 convolutional layers with their corresponding ReLU layers. After each of these, except between the second and third layers, there is a max pooling layer. At the end of the network there is a softmax layer for the classification.

In order to study the contribution of each layer, we performed ablation tests, that is, we removed some layers and retrained the resulting networks again to evaluate the change in performance due to the removed layers. The learning rate was not changed for these experiments. The first experiment consisted of eliminating all of the *ReLU* layers, thus reducing the complexity of the network by taking out these non-linear operations. The second experiment discarded the middle portion of the architecture, expressly, the second and third convolutional layers as well as the subsequent max pooling and corresponding *ReLU* layers. The final experiment retained only two max pooling layers, after the first and fourth convolutional layers.

3. Results

In particular, we trained our network for 25 epochs with a fixed learning rate of 0.001. The network was evaluated on the train and validation sets of the texture dataset for each epoch. Fig. 2 shows the progression of the error of the loss function and the classification error per epoch of the top detection (only one label per image) and the top 5 detections. As expected, the error is smaller when having 5 predictions per image (0.0172) as opposed to 1 (0.192). This is due to the larger margin of error in the top 5 modality.

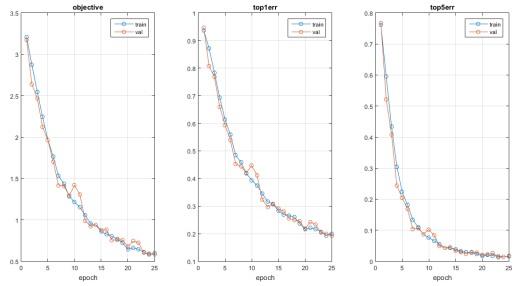

Figure 2. Performance in the classification task for the original network. Left panel: change of the objective function in each epoch (0.581). Center panel: top prediction error 0.192. Right panel: top 5 prediction error 0.0172.

When designing the architecture it was essential to adjust the filter sizes of the convolutional layers, as well as the number of the outputs and inputs in consecutive convolutional layers. If the filters were to large, they would not fit in the image and if the inputs or outputs did not correspond, then, there would not be a proper information flux throughout the layers. Similarly, it was important when defining max pooling layers to understand the dimensions of the image, because the pooling need not exceed the available information.

Another element to take into account was the learning rate. This parameter corresponds to the amount that the network's weights are changed in the learning processes. This

is paramount for finding the optimal weights for the function. If the learning rate is too small, then the network may get trapped in a local minimum of the loss function or may take too long to converge. Likewise, if it is too large, then the network may change the weights so much that it may not learn anything useful. Furthermore, to reduce the probability of overfitting, a jitter function was used in the network's training. This is accomplished by performing small perturbations to the images, rotations and translations, to discourage the network to memorize the training images. Expressly, to avoid learning specific locations within the image and for the network to learn the relationship between the pixels.

The results of the ablation tests are described below. The architecture of the modified network without non-linearity is shown in Fig. 3. The convolutional and max pooling layers did not change. This network was trained and the results in the classification task are shown in Fig. 4. The objective function tends to decrease until epoch 15, the next values are not plotted in the graph since it was displayed as `NaN`. Similarly, the top 1 error and top 5 error decreased as the epoch number increased until it reached epoch 15. After this epoch, both errors significantly increased up to 96%, which means that the classification accuracy decreased up to 4% that corresponds to the random classification probability.

layer	0	1	2	3	4	5	6	7	8	9	10	11
type	input	conv	mpool	conv	conv	mpool	conv	mpool	conv	mpool	conv	softmx1
name	n/a	layer1	layer2	layer3	layer4	layer5	layer6	layer7	layer8	layer9	layer10	layer11
support	n/a	5	2	2	2	2	2	2	2	2	2	1
filt dim	n/a	1	n/a	12	32	n/a	64	n/a	90	n/a	128	n/a
filt dilat	n/a	1	n/a	1	1	n/a	1	n/a	1	n/a	1	n/a
num filts	n/a	12	n/a	32	64	n/a	90	n/a	128	n/a	25	n/a
stride	n/a	1	2	1	1	2	1	2	1	2	1	1
pad	n/a	0	0	0	0	0	0	0	0	0	0	0
rf size	n/a	5	6	8	10	12	16	20	28	36	52	52
rf offset	n/a	3	3.5	4.5	5.5	6.5	8.5	10.5	14.5	18.5	26.5	26.5
rf stride	n/a	1	2	2	2	4	4	8	8	16	16	16
data size	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN
data depth	NaN	12	12	32	64	64	90	90	128	128	25	1
data num	165	165	165	165	165	165	165	165	165	165	165	1
data mem	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
param mem	n/a	1KB	0B	6KB	32KB	0B	90KB	0B	181KB	0B	50KB	0B

parameter memory|361KB (9.2e+04 parameters)|
data memory| NaN (for batch size 165)|

Figure 3. Network architecture without non-linearity layers.

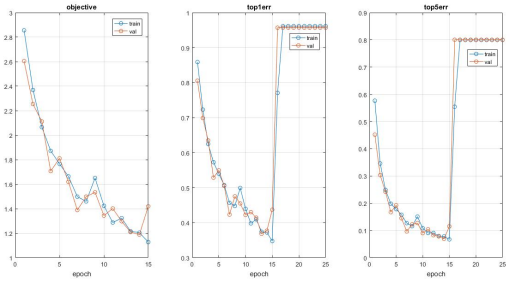

Figure 4. Performance in the classification task for the network without ReLU function. Left panel: change of the objective function in each epoch. Center panel: top prediction error. Right panel: top 5 prediction errors.

The design of the network for the second experiment and the performance results are shown in Figs. 5 and 6, respectively. Since two intermediate convolutional layers were removed, additional changes were done in order to keep coherence between the input and output sizes, and in the kernel dimensions of the remaining convolutional layers. Since less convolutions were performed, the kernel dimension was increased for the first convolutional layer to overcome mismatch in dimensions. Additionally, the max pooling parameters were changed to adjust the input dimensions for the following layer. By removing the two convolutional layers, the objective function is not minimized in each epoch, for example, after the fifth epoch, it starts to increase and decrease in some occasions. The final top prediction error was 67.8% and less than 30% for the top 5 prediction error. The former corresponds to an accuracy of 32.2%, which is greater than chance, but the performance decreased when compared to the original network.

layer	0	1	2	3	4	5	6	7	8	9	10	11
type	input	conv	relu	mpool	conv	relu	mpool	conv	relu	mpool	conv	softmx1
name	n/a	layer1	layer2	layer3	layer4	layer5	layer6	layer7	layer8	layer9	layer10	layer11
support	n/a	11	1	2	5	1	2	2	1	4	2	1
filt dim	n/a	1	n/a	n/a	12	n/a	n/a	32	n/a	n/a	64	n/a
filt dilat	n/a	1	n/a	n/a	1	n/a	n/a	1	n/a	n/a	1	n/a
num filts	n/a	12	n/a	n/a	32	n/a	n/a	64	n/a	n/a	25	n/a
stride	n/a	1	1	2	1	1	2	1	1	4	1	1
pad	n/a	0	0	0	0	0	0	0	0	0	0	0
rf size	n/a	11	11	12	20	20	22	26	26	38	54	54
rf offset	n/a	6	6	6.5	10.5	10.5	11.5	13.5	13.5	19.5	27.5	27.5
rf stride	n/a	1	1	2	2	2	4	4	4	16	16	16
data size	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN
data depth	NaN	12	12	12	32	32	32	64	64	64	25	1
data num	165	165	165	165	165	165	165	165	165	165	165	1
data mem	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
param mem	n/a	6KB	0B	0B	38KB	0B	0B	32KB	0B	0B	25KB	0B

parameter memory|101KB (2.6e+04 parameters)|
data memory| NaN (for batch size 165)|

Figure 5. Network architecture without two intermediate convolutional layers and subsequent max pooling.

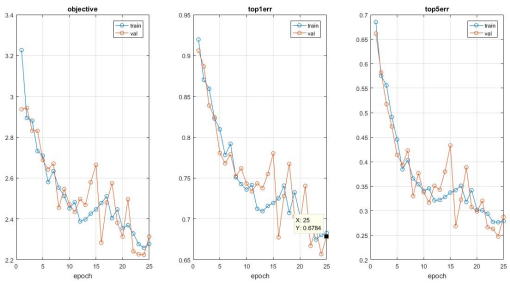

Figure 6. Performance in the classification task for the network without 2 intermediate convolutional layers. Left panel: change of the objective function in each epoch. Center panel: top prediction error. Right panel: top 5 prediction errors.

The final experiment aimed to evaluate the effect of the max. pooling layers, so we preserved the max. pooling layer after the first and fourth convolutional layers (the original network had four max pooling layers). The number of convolutional layers did not change, as well as the ReLU layers after each convolution. The detailed architecture of

this network is shown in Fig. 7 and the performance results in Fig. 8. In this case, the objective function behaves similar to the original network, but it does not reach the minimum objective of the original network. As expected, the performance was better for the training set. When comparing the top prediction error for the validation set, it was reduced up to 26.1% and less than 10% for the top 5 prediction errors. The former prediction error corresponds to an accuracy of 73.9%, which is slightly lower than the performance of the original network (80.8%). On the other hand, it is important to notice that the top 5 prediction errors reached a plateau after the epoch 15, meaning that the performance did not increase even if more epochs were used to trained the network.

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
type	input	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	softmxl
name	n/a	layer1	layer2	layer3	layer4	layer5	layer6	layer7	layer8	layer9	layer10	layer11	layer12	layer13	layer14
support	n/a	5	1	4	2	1	2	1	2	1	4	2	1	2	1
filt dim	n/a	1	n/a	n/a	12	n/a	32	n/a	64	n/a	n/a	90	n/a	128	n/a
filt dilat	n/a	1	n/a	n/a	1	n/a	1	n/a	1	n/a	n/a	1	n/a	1	n/a
num filts	n/a	12	n/a	n/a	32	n/a	64	n/a	90	n/a	n/a	128	n/a	25	n/a
stride	n/a	1	1	4	1	1	1	1	1	1	4	1	1	1	1
pad	n/a	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rf size	n/a	5	5	8	12	12	16	16	20	20	32	48	48	64	64
rf offset	n/a	3	3	4.5	6.5	6.5	8.5	8.5	10.5	10.5	16.5	24.5	24.5	32.5	32.5
rf stride	n/a	1	1	4	4	4	4	4	4	4	16	16	16	16	16
data size	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN	NaNxNaN
ata depth	NaN	12	12	12	32	32	64	64	90	90	90	128	128	25	1
data num	165	165	165	165	165	165	165	165	165	165	165	165	165	165	1
data mem	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
aram mem	n/a	1KB	0B	0B	6KB	0B	32KB	0B	90KB	0B	0B	181KB	0B	50KB	0B

rameter memory|361KB (9.2e+04 parameters)|
data memory| NaN (for batch size 165)|

Figure 7. Network architecture with two max pooling layers.

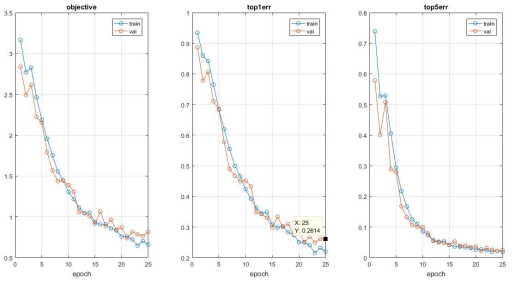

Figure 8. Results with two max pooling layers. Left panel: change of the objective function in each epoch. Center panel: top prediction error. Right panel: top 5 prediction error.

From the ablation tests results, we were able to identify the layers that improved the performance of the network. In particular, including the ReLu functions allows the output of the network not to be a linear combination of the inputs. When these non-linearity layers were removed, the performance was the same as random classification, so the potential of the network is lost. The effect of removing convolutional layers also affects the classification accuracy because these layers determine the feature representation of the input images, but this representation is not well understood [6]. However, the experiments of [6] showed that deeper convolutional layers could provide more specific and discriminative information for assigning a specific label.

Finally, the classification accuracy did not decreased much when removing two max pooling layers. This could be due to the fact that another two max pooling layers remained at the architecture. Removing these extra layers changed the dimensions of the remaining pooling, that is the way in which spatial information is added. Since the dimension was increased (4x4 windows), more pixels were merged into the same output. When more subsampling is performed, it results in a bigger receptive field seen by each neuron in the original image.

4. Conclusions

The performance of the network is greatly affected when the ReLU layers are not used, which is the expected output of such process. This is because the networks would only be left with convolutional and max pooling layers, whose outputs are just responses to filters and not any other posterior processing. After several epochs training networks with this architecture, the performance drops approximately to what random classification would produce. Removing intermediate convolutional layers significantly affected the classification accuracy.

A jitter function appears to be useful when trying to reduce overfitting of the whole network, since it makes the net's output less dependent of the output of each single neuron, and more dependent of the general output. The error, when taking into account the top 5 of the best performing network, was 0.0172, while that error was 0.192 when considering only the first prediction. With the original network architecture the classification accuracy (80.8%) was better than the first approach used to classify textures based on textons histograms and k-means algorithm.

References

[1] "UC Berkeley Computer Vision Group - Recognition", Www2.eecs.berkeley.edu, 2017. [Online]. Available: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/shape/. [Accessed: 06- Apr- 2017].

[2] A. Vedaldi and A. Zisserman, "VGG Practical", Robots.ox.ac.uk, 2017. [Online]. Available: http://www.robots.ox.ac.uk/%7Evgg/practicals/cnn/index.html. [Accessed: 11- May- 2017].

[3] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012.

[4] "MatConvNet", Vlfeat.org, 2017. [Online]. Available: http://www.vlfeat.org/matconvnet/. [Accessed: 11- May- 2017].

[5] "Ponce Research Group: Datasets", Www-cvr.ai.uiuc.edu, 2017. [Online]. Available: http://www-cvr.ai.uiuc.edu/ponce_grp/data/. [Accessed: 02- Mar- 2017].

[6] P. Pulkit, R. Girshick and J. Malik. "Analyzing the Performance of Multilayer Neural Networks for Object Recognition", European Conference on Computer Vision 2014.