Lab 11 - TextureCNN

Jhony A. Meja
Universidad de Los Andes
Biomedical Engineering Department
ja.mejia12@uniandes.edu.co

Abstract

convolutional Neural Networks (CNN) are useful for studying vision problems as classification tasks. In this paper a texture dataset containing 20000 images (15000 for Train, 2500 for Validation and 2500 for Test) and 25 types of textures was considered. A CNN based on AlexNet was built in order to address the classification problem. 5 convolutional layers and 3 fully connected layers were taken into account for the architecture of the network. Each layer was activated with the ReLU function, and some of them went through pooling and dropout processes. The effect of data transformation such as Normalization and Color Jitter in the CNN was studied. The consideration of normalization showed better results in terms of ACA. Finally, the CNN's proposed architecture efficacy was tested by ablation tests. The first layers focus on reducing the computational cost, while the last ones focus on classifying correctly the previously gathered information. After doing this process the network was redefined to have only 6 layers (4 convolutional and 2 fully-connected). To conclude, CNN is a powerful method for solving classification problems.

1. Introduction

Convolutional networks allow studying the classical vision problem's using neural networks. The idea is to convolve the images with a bank of filters for identifying patterns that will be useful for determining the final task. One of the main advantages of neural networks in general is that they provide both the feature extraction process and the classification/detection/segmentation algorithm.

Roughly speaking, convolutional networks receive an image as input, and in it's first layer a family of filters is applied for detecting general descriptors, like borders, blobs, changes in intensities, lines, curves, etc. The following convolutional layers keep extracting information with higher complexity. The deeper the layer, the most semantic information it has. However, it is impossible to keep all of the information at the same time. That is why processes like strides and pooling are done for reducing the computational cost. Additionally, processes like padding control that not to much information is lost in the process of lowering computational cost.

In general, sub-sampling processes are done as the neural network gets deeper. When the descriptor is concise enough, the data goes into fully-connected layers. This layers gather all the information previously found and starts finding patterns for improving the task being done (classification, detection). Finally, the last layer has a soft-max for giving the final result (category or detection).

2. Materials and Methods

The proposed neural network was mainly based in AlexNet [1]. It was composed by 5 convolutional layers and 3 fully-connected layers. No pre-processing was performed to the images. The raw images of 128x128 were directly used in the neural network.

2.1. Dataset

The dataset consisted of 20000 images of textures (15000 for Train, 2500 for Val and 2500 for Test). Each image was of size 128x128, and had a corresponding label. In total there were 25 classes. The dataset was sub-divided in batches of 100 images. Finally, 20 epochs were considered for optimizing the neural network.

2.2. Convolutional Neural Network Architecture

Convolutional layers

The first convolutional layer gave as output 64 layers, after being filtered with a kernel of 8x8, and having a stride and padding equal to 2. The idea of this first layer is to five general information of each image but without losing important local data. The kernel of 8x8 is big enough to give a general insight of each image. The padding is trying to keep as much information as it can by adding zeros around the image for allowing the convolution process. The stride is lowering the computational cost by not taking into account every pixel. However, the stride is not big enough for losing

possible valuable information. In this first layer a process of ReLU function was applied for inducing non-linearity. Finally, a MaxPooling process was performed with a kernel of 4x4 and a stride of 2. The idea was to lower the computational cost.

All of the convolutional layers were affected by ReLU function for adding non-linearity and fastening up the convolution process. The second convolutional layer gave as output 192 layers. The idea is that this layer would be more descriptive than the first one. A kernel size of 5 and a padding of 2 was performed for obtaining more general information. After that, a pooling process was done with a kernel of 3 and a stride of 2 for keeping the computational cost low.

The third and the fourth convolutional layer only did ReLU as an additional process. Their layer outputs were 384 and 256 respectively. Both were performed with kernels of 3x3. The idea of the third layer is to be as descriptive as it could be. The fourth layer idea is to start to gather common information.

The fifth and last convolutional layer only filtered the images with the same number of filters (256). The idea was to make some final refinement before moving to the fully connected layers. A Max Pooling was also performed for keeping the information dimensions low.

Fully-connected Layers

The first fully-connected layer received a vector of 6x6x256, turned into vector (9216). Firstly, a Dropout process was done for preventing over-fitting from the training data. After that, a linear layer was applied giving an output of 4096 responses. The idea was to keep gathering common patterns. After that, a ReLU function was applied for keeping the non-linearity and keeping the process fast.

The second fully-connected layer was anticipated by another process of Dropout for preventing over-fitting again. After that, a ReLU function was applied for the previously mentioned reasons.

The final fully-connected layer did a soft-max for giving the final predicted category (one of the 25 labels).

2.3. Initial normalization

A process of initial normalization was done for keeping the images in a common comparison. Also, the idea was to expand a little bit the dynamic range of the original data. This consideration significantly improved the results obtained.

2.4. Color Jitter

Color Jitter was performed as an alternative for data augmentation. The idea was to change randomly features like brightness, contrast, saturation and hue of an image. The values chosen were 0.5, 0.5, 0.5 and 0.25 respectively. This process was performed both with and without the initial normalization.

2.5. Ablation Tests

Each layer was removed and its effect was seen. Changes in the input-output received by each layer were performed for maintaining the structure of the network. The rest of the parameters as filter size, padding and stride were left as default. As it was stated before, each layer had some related operations, so if the layer was removed, so did its corresponding operations.

3. Results

3.1. Data Transformations

Initially, the network was designed without Initial normalization and Color jitter. This initial performance showed an Accuracy of 73.51% for Train and 66.88% for Validation. There was not perceived a phenomenon of over-fitting. The Accuracy of Train and Validation were similar through the epochs, which suggested that the method is robust.

Taking into account the Initial normalization the performance increased to 94.92% for Train and to 87.76% for Validation. However, this consideration showed and increased tendency for over-fitting. The Validation set reached it's maximum value and the started to decrease until 85.36%. The over-fitting could be explained taking into account that after normalization the data look alike into each other, and in some point the network starts seeing differences were they aren't. To sum up, the consideration of an initial Normalization showed that it could lead to better results but having the risk of over-fitting.

Taking into account both Initial normalization and Color Jitter showed better results than the original network, but worse that only Initial normalization. The best ACA obtained was 91.17% for Train and 81.58%. The increase between Train ad Test was almost constant, which suggests that this method is more robust than using Initial normalization alone. Additionally, a plateau phase was observed near the best ACA values, which supports the hypothesis that this methods provides more robust results. The robustness of this method makes sense. First, some randomness is added to the dataset by using Jitter, and then, that randomness is somehow controlled by the normalization process. It changes the data but not to much for producing misclassifications.

Color Jitter only showed better results that the original raw data, but worse than the other two methods. The best ACA obtained for Train was 77.42% and 74.76% for Validation. Jitter wasn't very useful because there was enough raw training data (15000). It is logical to think that the number of data is significant enough. Adding more variation to

the data isn't useful when there is a lot of data. Maybe Color Jitter could be important when there is less data available.

A summary of the best performances of each Data Transformation method is shown in Table 3.

Transformation Method	Train ACA	Val ACA
None	73.51%	66.88%
Normalization Only	94.92%	87.76%
Normalization + Jitter	91.17%	81.58%
Jitter Only	77.42%	74.76%

Table 1. Best performances in Train and Validation sets for each Transformation method considered.

As seen in the previous table, the best results are obtained considering only Normalization. However, it is important to say that this method might not be fully robust, and there is a risk of suffering over-fitting.

3.2. Ablation Tests

Ablation Tests were performed considering the best method (Initial Normalization only). The results of eliminating one layer can be seen in Table 2.

Missing Layer	# Parameters	Train ACA	Val ACA
-	5'7095,321	94.92%	87.76%
1	1'026'722,969	NA	NA
2	224'117,721	99.12%	87.28%
3	55'989,017	96.33%	88.64%
4	56'505,241	96.68%	89.20%
5	224'277,401	97.39%	88.28%
6	40'314,009	97.22%	88.68%
7	40'314,009	97.41%	88.00%
8	40'314,009	93.91%	86.72%

Table 2. Ablation Tests for each layer performed in the best method obtained.

The model ran out of memory when removing the first layer. This is obvious. The first layer has the function of extracting very superficial features and reducing the features dimensions at the same time. The number of parameters is 200 times less when considering this first layer. This suggests that the first layer is accomplishing its function.

The function of the second layer is similar to the first one. However, the model still runs because most of the hard work is done by the first layer. Anyway, the number of parameters is 50 times less when considering this layer. Without this layer the ACA Train improved because the model learned forcefully that data. The generalization of the model didn't improve because the Val ACA was practically the same. To sum up, the second layer fulfills the function of reducing the computational cost without losing important information valuable for producing a general/robust model.

The third layer still has sub-sampling functions, but it also has more importance on detecting fine details. The number of parameters without this layer is 10 times more. Although the results obtained in terms of ACA were better without this layer, the time it took and the computational cost was significantly higher. Also, the improvement was only of 1%, so it is considered that removing this layer is not worth it.

The fourth and fifth layer have the function of extracting fine details. Their goal is to be descriptive enough for roughly identifying an entity. The fifth layer also has the function of subsampling previous to the fully connected layers. The fourth layer had no effect in the number of parameters, neither on the ACA. This layer should be deleted, as it is doing practically the same that the fifth one (it actually has the same kernel size and padding). The fifth layer lowers the computational cost (50 times less parameters) but has no effect in the ACA. In this way, the fifth layer accomplishes its function of lowering computational cost and should be kept.

The sixth layer is the first fully connected layer. It's function is to reduce the dimensionality by fusing together characteristics that look alike. The presence of this layer reduces the number of parameters by a factor of 8. This layer fulfills the requirement of reducing the computational cost without significantly damaging the ACA.

The seventh layer should have a greater impact on the ACA. Otherwise, it is doing the same as the sixth one. The later is the case. The performance was similar to the original one but without reducing the computational cost. This layer is doing practically the same than the sixth one. Again, the idea of 'checking' does not work. Not even the dropout had a significant effect. The seventh layer should be removed.

Finally, the eight layer showed fluctuations in the ACA between epochs. This suggests that the last downsampling is needed to provide robustness to the model.

To sum up, the layers that were thought to 'look over again' didn't have any effect, so they should be removed. For a layer to be important it must improve the ACA or reduce the computational cost. If a layer is not doing any of those, the most probable thing is that it is not making anything but consuming computational resources.

Finally, without the non-productive layers the neural network has the following performance:

# of Layers	#. of Params	Train ACA	Val ACA
6	3'9723,929	99.34%	90.20%

Table 3. Best performance in Train and Validation sets after removing the non-productive layers (layers 4 and 7), found by ablation tests.

After removing unnecessary layers the neural network was faster and needed less epochs to show better results in

terms of ACA.

4. Conclusions

Convolutional Neural Networks are a powerful method for addressing a classification problem. It can even be more effective than textons in a dataset designed for classifying textures.

Data Transformations such as Normalization and Color Jitter can be useful for improving performance and robustness. More specifically, Normalization provides better results in terms of ACA, while the combination of Normalization and Color Jitter provides a more robust neural network.

Data augmentation techniques like Color Jitter are not very useful when a considerable amount of data is already supplied as raw data. However, these processes can be useful for databases with less data.

The first layers of a convolutional network play a crucial role in both extracting global information and reducing the feature's extraction dimensions. The goal is to find an optimum balance between extracting valuable information and saving GPU space. Without any of the first layers the computational cost rises dramatically (200, 50 or 10 times more parameters).

The last layers of a convolutional network usually correspond to fully-connected layers. This layers are responsible of gathering information together for being able to classify correctly. These layers affect directly the ACA obtained, but its performance also depends of the information obtained in the previous layers.

Unnecessary middle layers can be identified by ablation tests. A layer should be removed if it is not contributing to none the ACA neither the computational cost.

References

[1] Krizhevsky, A., Sutskever, I., Hinton, G. *ImageNet Classication with Deep Convolutional Neural Networks*. University of Toronto. 2012.