Convolutional Neural Networks for Texture Image Classification

Maria Camila Escobar
Universidad de los Andes
Bogotá D.C, Colombia
mc.escobar11@uniandes.edu.co

Laura Gongas
Universidad de los Andes
Bogotá D.C, Colombia
l.gongas10@uniandes.edu.co

***Abstract*- Convolutional neural networks (CNN) are a growing set of algorithms in the machine learning field. Nowadays, with the use of CNNs it is possible to obtain features of thousands of images and correctly classify them. Here we develop a neural network as a texture classifier including convolutional and fully connected layers and we include mechanisms to reduce model overfitting such as Pooling, dropout and jitter. Finally, it was possible to achieve an ACA in the test dataset of 82.9% and there were additional experiments to determine the relevance of including jitter and ablation tests to evaluate the importance of each layer in the network.**

1. Introduction

In machine learning neural networks are a set of algorithms designed with the objective of recognizing patterns in large amounts of data. Neural networks are inspired by the human brain in the way that each neuron has an input and calculates an output based on parameter estimation. The final result of the neural network will depend of the outputs given by the neurons in each of the layers. The basis of NN is the algorithm of retropropagation, in which the model estimates a loss function, usually mean square error, between the results given and the groundtruth and uses this information to recalculate the weights and bias given to each neuron of the layers [4].

Convolutional neural networks (CNNs) have the same underlying principle as the other neural networks previously described. Each neuron has a weight and a bias that must be adjusted to achieve the best possible model performance. However, convolutional neural networks are adapted to use complete images as inputs, therefore creating neurons that can calculate outputs with three dimensions. Additionally, neurons in a CNN layer will only be connected to a particular region of the layer before it, instead of being connected to the whole set of neurons in the previous layer [3].

2. Materials and methods

Our network consisted in three convolutional layers, each one followed by a ReLU and a maximum pooling layer. Also, we used a dropout layer after pooling 2 and 3, and batch normalization for all of the convolutional layers. At the end of the network two fully connected layers were implemented with a dropout in between.

The ReLU layers or activation layers were added to apply nonlinearity to the system. Mainly they were used to speed up the convergence of the gradient descent. The pooling layers were also used to reduce computation costs because they reduce spatial size and therefore the number of parameters. Additionally, a decrease in parameters avoids overfitting too. Dropout layers, as their names indicate, drop out a random set of activations in a layer which reduces overfitting.

Batch normalization makes reference to the normalization of activations of the previous layer for each batch. This method makes network training faster because it converges faster. Consequently, a higher learning rate may be used [1]. Fully connected layers are at the end of the network because it allows the decision of classification to be taken based on the whole image. This is due to the connection that every neuron has with the following layer which permits a flow of information between the location of the pixels or the input dimension and the output class.

Even though the initial network worked pretty well, we varied the learning rate and the amount of epochs along with ablation and jitter. The learning rate was finally fixed to 0.001, the batch size to 30 and the amount of epochs to 450.

3. Results

The following tables show the results in train and validation sets by implementing jitter and ablation. Also, the performance of our neuronal network was compared in some classes of the datasets with the results obtained by implementing textons. Finally, a graph is presented with the ACA in train and validation sets for each epoch.

Table 1. Transformations done to the train dataset.

Transformations of train dataset	Best ACA	
	Train	Validation
Random horizontal flip	79.28	84.12
Random horizontal and vertical flip, and color jitter	79.01	83.80

Table 2. Ablation tests results by removing convolutional and fully connected layers of the network.

Layer Removed	Best ACA	
	Train	Validation
Convolutional 1	53.47	64.60
Convolutional 2	67.08	59.24
Convolutional 3	55.03	57.40
Fully connected 1	82.89	84.76
Fully connected 2	84.34	85.24

Table 3. Comparison of lab5 results with current performance

	lab5	now
ACA	42.8%	82.9%
Upholstery	100%	96%
Marble	0	76%
Wallpaper	0	88%
Plaid	0	96%

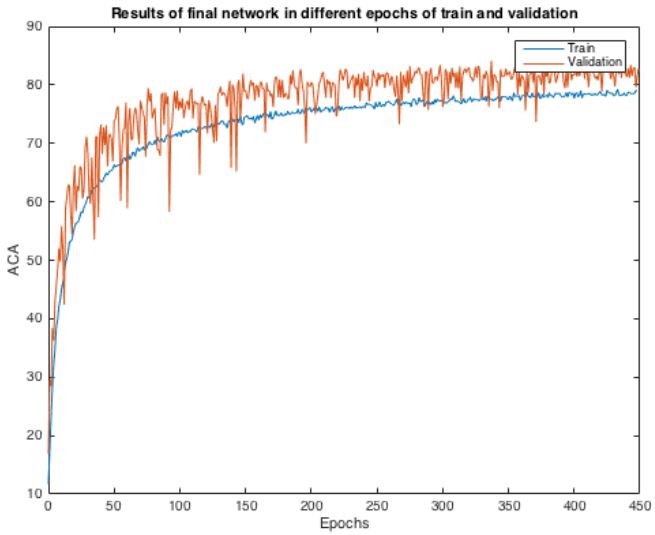

Figure 1. Results of final network in different epochs of train and validation datasets.

4. Discussion

We did not have to make significant changes to our original network in order to obtain acceptable results. However, initially we had only 50 epochs and as it is shown in Figure 1, the maximum is reached after approximately 300 epochs. We also had a very high learning rate so the training did not converge. To solve this problem, we lowered the rate until we reached the highest possible value for convergence. Another challenge was to choose kernels and strides in the convolutional layers. Nonetheless, we concluded that a big stride does not ignore as much information in our dataset because textures are repeating patterns so data should be similar through the majority of the image.

The first optimization parameter we wanted to tune were the transformations done to the train dataset. Initially, it was decided that the train dataset should have a random horizontal flip transformation. This was made to ensure that the data would not get over-fitted by the model and it would be more representative of the full textures it was meant to portray. However, when adding more transformations to the dataset, such as vertical flip and color jitter, the model gave a lower ACA value than using only random horizontal flip (table 1). A plausible explanation for this decrease in ACA could be that, since this transformations are performed only on a random sample of train images, the transformations created an inconsistency on the patches that belonged to the same class. Applying more than one transformation to the train patches could result in the loss of important data to determine the category of the texture.

Moving on to the performance of each layer of our neural network we performed an ablation test. Which consists in deleting each layer and observing the change in performance to determine whether the information stored in that layer is relevant or not for the model. The model, including all three convolutional layers and two fully connected layers had a performance in the train dataset of 79.28% and in the validation dataset of 84.12%. It is possible to identify that removing any of the first three convolutional layers will result in a lower performing model. This is due to the fact that, the first layers of a neural network contain generalized features and characteristics that are useful for discriminating between classes.

The further the model goes in the convolutional layers the more specific the filters will be. Therefore, removing the third convolutional layer will result in the worst performance,since it will not have as much specific information of the texture patches, as seen in table 2. Regarding the performance of higher layers, it is possible to observe that the ACA improves when removing one of these layers. Since fully connected layers give an overview of the spatial pattern of the image, the results given in table 2 suggest that it is more relevant in classification to analyze the activation of initial filters but not the spatial information regarding activations [2]. This results are consistent with the task, since textures do not need spatial recognition but rather an effec-

tive pattern discrimination.

In table 3 it is possible to compare the results obtained with our previously developed classification method and our current method since they were both tested on the same texture dataset. Our best CNN yields an overall ACA in the test dataset of 82.9%, while the best configuration of Random Forest classifier, achieved in our previous method could only obtain an ACA of 42.8%. Additionally, the best category for classification in our previous method was upholstery, with an accuracy of 100% and the worst categories were marble, wallpaper and plaid with zero images correctly classified. If we compare this with the results given by our current method it is possible to observe that upholstery still holds a relatively high ACA value and the other categories that had a 0 ACA improved to values up to 96%. This analysis confirms the fact that our CNN will learn more intricate patterns of each category that were not easily identifiable by our previous method, therefore making it possible to classify textures that were previously considered too complex.

However, it is not possible to directly compare the influence of the patch size in the results since our first method was developed extracting random pixels from the whole image, not obtaining random patches of a fixed size. Therefore, it can also be thought that the improvement in ACA with our new method can be due to the subsampling of the image into patches rather than random pixels. The use of patches ensures the preservation of local patterns within the texture.

Figure 1 shows that the methods employed to reduce overfitting worked. First of all, we can see there is no overfitting because the results of validation were higher than the ones of train. Second, it is possible to observe that the system reached a plateau so the results do not decrease as the epochs increase.

5. Conclusions

In conclusion, our method had a performance of 82.9% in the test dataset. Compared with the previous method we had created using texton histograms and a Random Forest classifier, our new method has an increase in ACA of 40% demonstrating the effectiveness of implementing neural networks for classifying tasks. Additionally, it is possible to confirm that our method did not go through overfitting since the results in validation are higher than in the train dataset and the ACA of the results reach a plateau phase when adding more epochs. Finally, to improve the results given by our model it would be recommended to implement training with different sized patches.

References

[1] Glossary of deep learning: Batch normalisation – deeper learning – medium. *Medium*.

[2] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *European conference on computer vision*, pages 329–344. Springer, 2014.

[3] A. Karpathy. Cs231n convolutional neural networks for visual recognition, 2018.

[4] C. V. Nicholson. Introduction to deep neural networks (deep learning) - deeplearning4j: Open-source, distributed deep learning for the jvm, 2018.