Final Project

Deep Neural Network for Image Classification

Professor: Leonardo Garrido L

Layla Tame A01192934
Jorge Mendez A00816559

December 9, 2019

Introduction

Throughout the last years, artificial intelligence has grown exponentially and with it the branch involving machine and deep learning. Both of these branches have allowed algorithms and products to be created, which facilitate jobs normally done by a human being; and not only that, but they also provide more precise and faster results those obtained by a person. AI has been applied in a variety of industries, from medicine to the manufacturing industry, the implementation of several different algorithms has had amazing results.

Deep learning, a branch of what we commonly know as artificial intelligence, is originally inspired in the real functionality of a brain, which works through neural networks. Its main purpose is to teach a computer to do what a human does naturally: learn through examples. It aims to incorporate learning processes and adaptation into electronic processes, allowing to look for results through the implementation of diverse strategies using neural networks, evolutionary algorithms, amongst others, adapting them according to the necessities of the problem one is looking to optimize or resolve. One of the main purposes of deep learning is to allow classification of data into certain categories with the intent of bettering processes, increasing knowledge in a certain field, amongst others.

For the development of this project, it was chosen to apply deep learning to create an image classification algorithm through the use of deep neural networks. It was done trying to get the best accuracy possible, altering and playing with different input parameters in order to obtain the best result possible. A data set of images was used to train the algorithm and later on put it to the test.

Throughout the document, an explanation to the data set will be presented, as well as the steps taken for the successful development and application of the project. Results obtained will be presented as well.

Data Set

The data set chosen for this project is called Flower Color Images and was obtained from Kaggle. It is a file that already contains clean data respecting images of a variety of flowers, representing different colors, types and sizes. The information is contained in a .h5 file, which is a file that contains a multidimensional array of scientific data and is ready for processing.

A total of 209 images conform the final data set used for the project. Although the number of images used is not very high, it was of great importance to find images with good resolution in order to improve training of the algorithm and therefore the results obtained. The images used were all 128x128 pixels and contained three different layers concerning the different scales of colors (RGB). Two different columns made up the input file, one containing the id of the image in question and another one describing the type of flower, or more specifically, the classification given to each image. Originally, 10 different species of flowers and their labels were found within the data set. The classification was as follows:

- 0 => phlox
- 1 => rose
- 2 => calendula
- 3 => iris
- 4 => leucanthemum maximum
- 5 => bellflower
- 6 => viola
- 7 => rudbeckia laciniata (Goldquelle)
- 8 => peony
- 9 => aquilegia.

For the purpose of this project, the algorithm designed did not have the intention of classifying the flowers into one of the different species' categories available. Instead, it was decided to implement an algorithm that would more specifically, tell you whether an input image is that of a rose or not.

The data set was divided into two different sets: the one used for training and a set used for testing the performance and accuracy of the algorithm. For the train file, 80% of the original file was used while the resting 20% was designated towards the test file. The total amount of images used for training was 169, while 40 were used for testing.

Methodology

An L-layer deep neural network architecture was used for the project. Specifically, a network made up of 4 layers: the input layer, three hidden layers and the output layer. The input layer consists of a flattened vector that contains 49,152 neurons obtained from the 209 images. Each pixel represents a feature of the input layer, which leads to getting 128x128x3 features to input into training the network, therefore obtaining an input vector of size (49152, 1).

The process followed was the one for such architecture, in which the flattened vector was obtained, then multiplying it by the weight matrix W1. Afterwards, the intercept is added and the relu vector is obtained. The process is repeated until the resulting vector W2 is obtained.

The general methodology was followed for the initial creation of the algorithm: initializing the parameters and looping according to the number of iterations. For each iteration forward propagation was executed, the cost function was computed, backward propagation was applied and finally the parameters were updated. After finishing n iterations, the results are used in order to classify whether the image is that of a rose or not. The classification was done based on the final parameters and sigmoid.

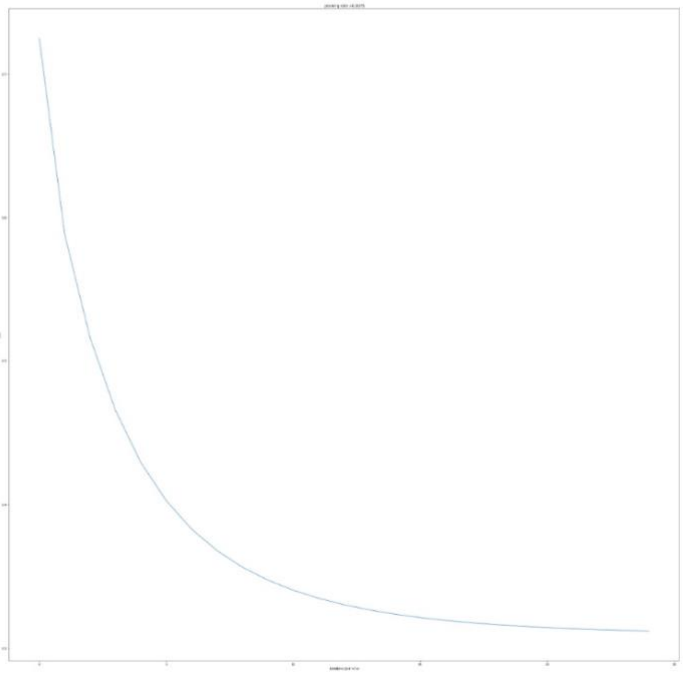
For the first version of the algorithm, predetermined parameters were used taking as a base case those used in class for classifying cat images. Once the algorithm was up and running correctly, the overall accuracy results were tried to be improved by altering and playing with the different parameters.

Four specific parameters were taken into account. The first was the learning rate which is the hyper-parameter that controls how much the weights are being adjusted at the end of each iteration. Then we have the number of total iterations to be done, which is ideally different according to the specific data set in question. The third hyper-parameter considered was the number of layers in the neural network. Finally, the number of neurons in each layer, which is assigned according to the dataset itself.

The first execution was done using the following parameters:

```
In [460]: layers_dims = [49152, 20, 7, 5, 1]
num_iterations = 2500
learning_rate=0.0075
parameters = L_layer_model(train_x, train_y, layers_dims, learning_rate, num_iterations, p
pred_test = predict(test_x, test_y, parameters)
print_mislabeled_images(classes, test_x, test_y, pred_test)
```

```
Cost after iteration 0: 0.724755
Cost after iteration 100: 0.588304
Cost after iteration 200: 0.516240
Cost after iteration 300: 0.465574
Cost after iteration 400: 0.429242
Cost after iteration 500: 0.402668
Cost after iteration 600: 0.382870
Cost after iteration 700: 0.367875
Cost after iteration 800: 0.356350
Cost after iteration 900: 0.347380
Cost after iteration 1000: 0.340321
Cost after iteration 1100: 0.334714
Cost after iteration 1200: 0.330222
Cost after iteration 1300: 0.326599
Cost after iteration 1400: 0.323657
Cost after iteration 1500: 0.321257
Cost after iteration 1600: 0.319289
Cost after iteration 1700: 0.317669
Cost after iteration 1800: 0.316330
Cost after iteration 1900: 0.315220
Cost after iteration 2000: 0.314298
Cost after iteration 2100: 0.313530
Cost after iteration 2200: 0.312888
Cost after iteration 2300: 0.312351
Cost after iteration 2400: 0.311901
```



```
Accuracy: 0.8999999999999999
```

It can be observed how the cost gradually decreases with each iteration, reaching 0.311 after iteration 2,400. The final accuracy obtained was of 89.9%.

In order to improve the accuracy results and therefore achieve a better classification of the imaged used for testing, the hyper-parameters were altered. The methodology used to experiment with them was trial and error. With each execution, one or several parameters where changed in order to observe how increasing or decreasing them affected the algorithm's performance and overall results.

Results

After trying out both architectures seen: 2-layer neural networks and L-layers neural networks, it was decided to continue working with the second one as it seemed to present better results than the first. The dimension of the layers was obtained from the data set and the pixels corresponding for each image.

Different experiments were done varying the different parameters mentioned before. At first, the algorithm was executed using the same original learning rate of 0.0075, but significantly reducing the number of iterations from 3,000 to 100. Doing so made the algorithm run much quicker than it originally did, but there wasn't much information obtained on the cost function as the number of iterations was very low. The cost after iteration 0 obtained was 0.68 and we didn't get a chance to observe how it decreases. However, the final accuracy obtained was fairly good with 85%.

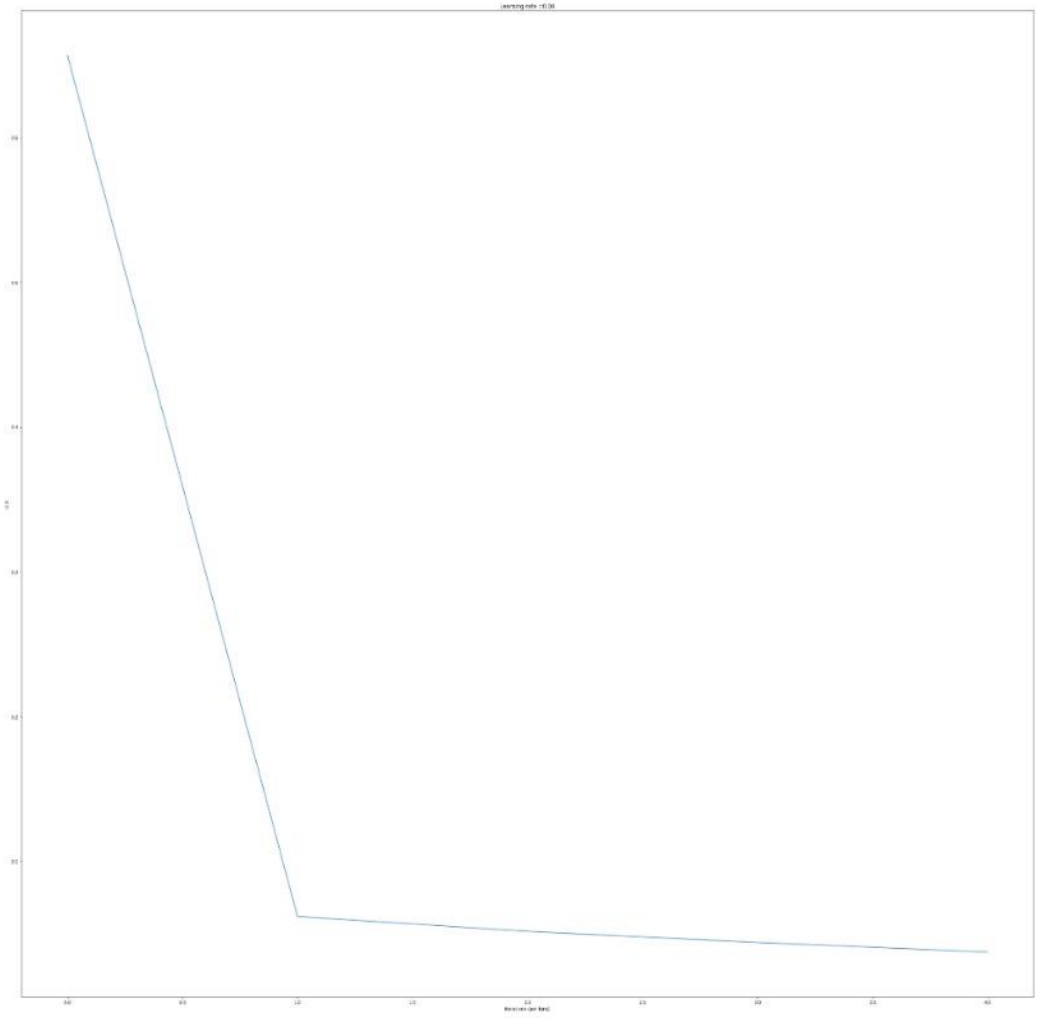
The second significant results gotten were with an experiment where it was decided to reduce the learning rate while increasing the number of neurons used for input. A total of 500 iterations were executed, the cost after iteration 400 was of 0.037 and it could clearly be observed how it decreased after each iteration, starting at 0.65. The final accuracy result was of 88.7%.

EXPERIMENT 1

More neurons Less learning rate

In [454]:
```
layers_dims = [49152, 20, 15, 25, 1]
num_iterations = 500
learning_rate=0.04
parameters = L_layer_model(train_x, train_y, layers_dims, learning_rate, num_iterations, p
pred_test = predict(test_x, test_y, parameters)
print_mislabeled_images(classes, test_x, test_y, pred_test)
```

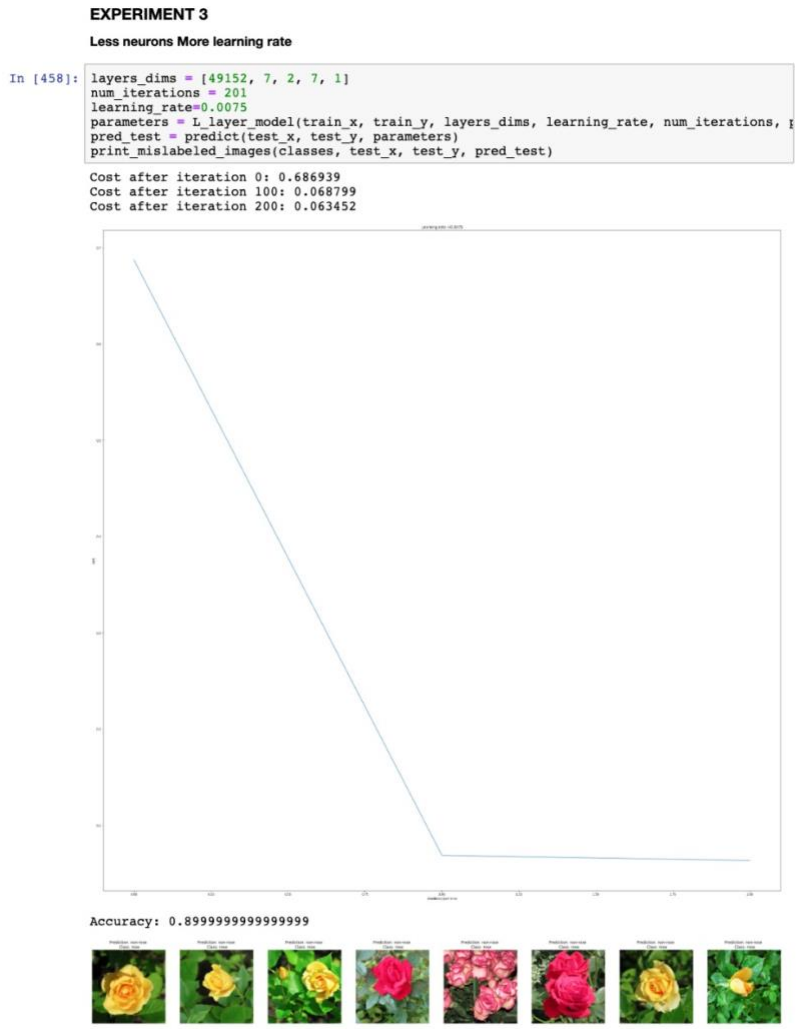
```
Cost after iteration 0: 0.656766
Cost after iteration 100: 0.061817
Cost after iteration 200: 0.051568
Cost after iteration 300: 0.043902
Cost after iteration 400: 0.037272
```



```
Accuracy: 0.8875
```

Final Results

Finally, a third significant alteration to the hyper-parameters was done that presented with good results. For the case of this experiment, the learning rate was increased while decreasing the number of neuros, doing it the other way around than on the previous experiment. This set of data and definition of parameters were the ones that presented the best and final results. Using a number of iterations of 201 and a learning rate of 0.0075, it was observed that the cost after each iteration was effectively decreasing, achieving a cost of 0.063 after iteration #200. The final accuracy obtained was of 89.99%, meaning that almost 9 out every 10 images introduced into the algorithm were correctly classified as being or not roses. The following results can be observed in the following image:



Conclusions

Although deep learning through the implementation of neural networks have gone a long way and represent a very real and efficient solution to a series of problems, it is important to achieve the correct implementation of the algorithms in order to obtain successful results. It is not enough to know only the bases behind the implementation of different algorithms, but it is of sum importance to know what is going on behind each function, to understand the purpose of each one of the parameters, in order to optimize the function correctly.

It can be observed through the results of this project and the use of the floral data set, that results can be greatly improved through the experimentation of different input data, different parameters, etc. It is important to note that improving an algorithm does not necessarily mean only improving the final outcome of accuracy, but also creating a code that is executed easily. In this case, we can observe that the initial parameters used to execute and train the neural network gave up the same results than the final parameters used after experimenting and doing several executions of the code. However, even though the final accuracy results were the same for these two scenarios, the execution time was much shorter for the one that used 201 number of iterations instead of 2,500.

When working with data sets containing big amounts of information and trying to get accurate results, it is very important to work with an algorithm that portrays success

rates in its results but that runs as fast as possible to ensure a proficient implementation of the same.

Video

https://drive.google.com/drive/folders/1onMO4NiJltbXxGjl9JBDg-8JHvLWpkfV?usp=sharing

References

- Belitskaya, O. (2017). Flower Color Images Data Set. Retrieved from: https://www.kaggle.com/olgabelitskaya/flower-color-images#0008.png