PokemonPredicter

Fabien Puissant Mélic Dufrêne

March 31, 2020

1 Introduction

Our project consists in making a pokemon prediction through a favorite pokemon given by the user. This project made us think about how to deal with images features and classification. Besides, we used more than images because we also have a csv file containing other data for classification and prediction

This is the url of the github repository :

https://github.com/fabienpuissant/PokemonPredicter

2 Sources

For this project, we have use a pokemon data set containing images of pokemons and a csv giving the types of each pokemons. This data comes from the website Kaggle : https://www.kaggle.com/vishalsubbiah/pokemon-images-and-typespokemon.csv
To store the data, we made two functions : the first **get_feature_for_csv()** will take all the photos and the csv file of the types and create a list which contains for each pokemon a list of all the data relative to him. The second function **add_data_to_csv(data)** will creater a csv file **data.csv** and store all pokemon information.

3 KMeans algorithm

In order to detect the colors of each pokemons, we used the mini batch KMeans algorithm. Indeed, we want to catch the 3 main colors of the pokemon because they are mostly composed of 3 main colors. This algorithm is implemented in the function **get_colors()**. We chose this algorithm because it is more efficient than KMeans.

4 The features of pokemons

We saw that we have stored the 3 main colors of each pokemon, it means the red, green, blue color of the picture. So we have already 9 features concerning the color of the pokemon. Besides, for each color we stored the percentage of the color considered, which will make our model more accurate. It means that we add 3 features for the pokemon. We also calculated the size of the pokemon by counting all colored pixels.

In our data set, we have a csv containing the types of each pokemon. So we decided to store the types of the pokemon in order to be more precise. We also obviously add the path of the image of the pokemon we deal with.

Eventually, we got 16 features from the initial dataset.

5 The pokemon comparison

The be able to predict pokemons, we had to compare each pokemon using our features. First, we compared the types of the pokemon, if there is not at least one same type, the pokemon does not match with the other. We also compared the size, the color and the percentage of color

5.1 The size

To compare the size of two pokemons, we have a coefficient size and we make the difference between the two sizes. If this difference is below the coefficient, the size matches.

5.2 The color comparison

The color comparison is more complicated than the size. We used a module named colormath which can calculate the difference between two colors. So we used the function **delta_e_cie2000**. This function permit calculate a coefficient **delta_e** which represent how the colors are close. For further information, you can go there :
https://colour.readthedocs.io/en/develop/generated/colour.difference.delta$_{EC}IE2000.html$
We finally compare this coefficient to a threshold that we defined.

5.3 Color distribution comparison

If the color matches to an other, we will compare the color distribution that represent the percentage of the color compare to the two others. If this percentage is below a threshold, the color distribution is valid.

6 Algorithm

We presented all the functionalities we used to make a prediction, now we will see the algorithm which use all of these.

- Get the favorite pokemon of the user through an input
- Then find the pokemon data in **data.csv**
- We will now iterate on the csv file to predict 5 five pokemons. We start with coefficients and threshold very low to get the nearest pokemons. At each iteration, we compare the size, color, type and color distribution of the pokemon and store it in a list if it matches. If after comparing all the pokemons, there is not 5 matches, we increase the thresholds in order to find other pokemons and we iterate all pokemons again. We repeat this operation until we find 5 pokemons and with this method, we get the nearest pokemons listed by likeness. It means that the first pokemon find is the nearest.
- After that, we display the pokemon of the user and the five predicted by the algorithm.

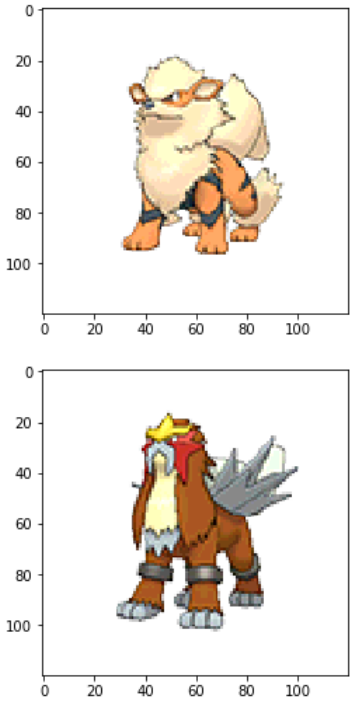

Figure 1: The nearest pokemon of Arcanine

7 Evaluation of the work

We noticed that there are pokemons that are not close to other, it means that with our algorithm, with the threshold increasing, it will predict really different pokemons. So it could be interesting to calculate a maximum threshold. Besides, some of photos are not encoded the traditionnal way so the color calcul could be false in some pokemons, so it would be nice to find these pokemons and make a specific processing.

8 Remarks about practical sessions

We wanted to underline that we did not have as much time as we expect for the project. Indeed, it was hard to understand and master all the concepts we saw in the first exercises and at the same time create a project. The main work of the project was out of practical sessions hours.

9 Conclusion

We manage to create a suitable pokemon predicter but it can still be improved. Indeed, it works well with some groups of pokemon like fire, grass pokemons or those which have color near to other. It is more difficult for isolated pokemons, we think that the threshold increase could be more accurate and maybe work more on matches conditions.