方法的简述 (50~100)

Open Image -> Preprocess image, reduce complexity -> Open color table & color json file, return matched colors -> Initialize models, fit data -> Iterate through image, stacks result

Ver 1

An image is first preprocessed into python dictionary object to reduce overall complexity, while color table is loaded and filtered. Preprocessed image is then processed by a model for its detailed color composition.

Ver 2

The given input image is first converted into RGB mode without alpha channel, it then goes to preprocess in order to reduce overall complexity. After image processing, the color table containing RGB value and names are opened and filtered, which goes into models as training data. Once complete, the model will iterate through the image, while returning names of each color and it’s reoccurrence. Final result is a recorded set of all color and it’s count, and can be percentage if set.

算法实现细节 (300+)

Ver 1

Opens image and converts it into RGB style while ignoring alpha channel.

If given any method, image will be processed by that method. The returning result of given method must be a 2-dimintional array, with the smallest element being RGB values. The array will then be iterated, each unique pixel will be recorded by its RGB value and reoccurrence. Result will be a Python dictionary object with RGB value as key, and it’s count as value.

The color table [1] is opened by pandas module with default argument, and the color json file is open by python internal function. Opened json file is then loaded by json module. If any given name has matched in json file, a filtered color-set will be returned. Else, the unedited raw table will be returned. If any element has failed matching (no such name), that row of result will instead be an empty row.

In this example, KNeighborsClassifier and LogisticRegressionClassifier are used. Both models are imported from sklearn module. The KNeighborsClassifier is initialized with parameters: result limited to one, and its algorithm set to brute. As for LogisticRegressionClassifier, it has parameters: max iteration set to element count of preprocessed image, and algorithm set to Liblinear. After initialization, RGB values and their Names from color-set are fitted into the module.

Each element in the preprocessed image is then iterated by a model. Each item will have its predicted name in color-set as result. Each result of pixel and its count is then added into the output. Final output has color name and it’s count as its structure.

Ver 2

Opens image and converts it into RGB style while ignoring alpha channel.

If given any method, original image will be processed by that method. Results of the given method must be a two-dimensional array, while the smallest item being RGB values. After image manipulation, the array of image is being iterated. Each unique pixel and their reoccurrence are counted, in format of python dictionary.

The color-table [1] is opened as DataFrame object from pandas module, and color json file loaded as python dictionary by json module. If given any object in color json file, a new DataFrame object with filtered colors is set. Else, unedited table is set. Pandas module then extracts RGB value and its name from table.

Models used are KNeighborsClassifier and LogisticRegressionClassifier, each model has non-default parameters. For KNeighborsClassifier, it has its result limited to one, and algorithm set to brute. For LogisticRegressionClassifier, it has max iteration set to length of preprocessed image, and algorithm set to liblinear. After initialization, RGB values and Names are fit into the model which is returned once completed.

In main processing sequence, each unique color of pixel is thrown into selected model. The closest name and count of color reoccurrence are being recorded and then save as python dictionary. At this point all computation are done.

[1] [Color Names | Kaggle](https://www.kaggle.com/datasets/avi1023/color-names)

字数很难达到要求, 毕竟本来代码就简短. 真的没什么能展开的.

或许应该增加/修改代码内容? 增加功能?

除此之外, 实现细节是否有必要详细到条件语句等内容?