RGB Image Character Classification

Gihwan "Kris" Kwon

Dec-07-2018

Abstract

This is Abstract.

1 Introduction

Meme is a humorous image, video, or piece of text that is copied and spread rapidly by Internet users. Sometimes, people want to search meme online and show it to their friends that they thought it was funny. However, often, it is hard to find the meme that a person is looking for. Currently, online meme search engines are relying on the hash tags to match user inputs with the results. However, normally, users do not catch the phrase from the hash tags, but the actual contents in the image. Therefore, it would be useful to have meme search engines to have an algorithm such that can find the memes based on the word contents inside the images.

Convolutional Neural Networks is a popular method to recognize images and classify its label. Currently, there are many research focusing on utilizing this method for recognizing text images in real life. However, classifying memes is different in that words in the memes are graphically added as contents. Therefore, making a model for classifying graphically oriented characters would be useful for developing a search engines for memes.

The main goal of this project is to contribute to the developers who would build a meme search engine for the users. To begin with, this project will focus on classifying graphical characters. This paper will explain how the model was built with the convolutional neural networking methods.

2 Method

The method used for this project is the Convolutional Neural Network. This section explains methods for this project: Data preprocessing, Training, Prediction.

2.1 Data Preprocessing

The original data sets are organized in the .csv file. The objects are consist of attributes such as *font* (string), *fontVariant* (string), *label*(int), *strength* (int), *italic* (int), *orientation* (int), *m top* (int), *m left* (int), *originalH* (int), *originalW* (int), *h* (int), *w* (int), and 20 by 20 size grey scale image matrix. Most of the attribute names are self-explanatory. However, to be more specific, *m top* and *m left* are coordiate where image of the character starts in the real image in terms of row and column. Moreover, *originalH* and *originalW* are original image size. Finally, *h* and texitw are the height and weidth of the image.

To begin with, some of the attributes are omitted in this process because recognizing font and whether the character is italic are not the part of this project. Moreover, all the character images are original image. In other words, image in the datasets only contains one character. Therefore, *label* of the character and the 20 by 20 size grey scale image is going to be used in this project.

Next step is to convert the grey scale matrix into the RGB matrix which is three dimensional. The result matrix after preprocessing would produce 20 by 20 by 3 image matrix. To be specific, conversion took place by appending the original grey scale matrix into the result matrix three times. The result is the N number of RGB image matrices.

Furthermore, data preprocessing step produces a vector that values in each index contains character label and this corresponds the image matrices stored in the image matrices. Finally, this step produces a matrix size of N by the number of unique characters. The value in the indices are initially zeros in floating point. After image conversions, based on the image label, vector for each image will contain 1.0 where index corresponds the image label. For instance, if there are five different unique image labels in 10 different objects, the result will be ten by five matrix containing 1.0 in the vector where index matches the character label, otherwise values will be 0.0s. In this case, the floating point one value specifies that the index, character label, is true for corresponding image.

2.2 Training

The method of Convolutional Nerual Networks requires multiple layers in the process of trianing. In a big picture, the layers could be divided into three layers: input, hidden, and output layers. The input layer receives input, data such as matrices, and output layer produces the result values produced by the model.

Result values are N by the number of unique characters matrix that each vector contains the probability of input to be the character of the corresponding index numbers.

The hidden are further divided into three layers: Convolution, Flattening, and Fully Coonnected layers.

Convolution Layer

Flattening Layer

Fully Connected Layer

2.3 Prediction

3 Experiment and Result

3.1

$$f(x) = x^2 \tag{1}$$

4 Conclusion

5 Reference

Hello World!