

Image Processing with Python

Introduction to Python

What is Python

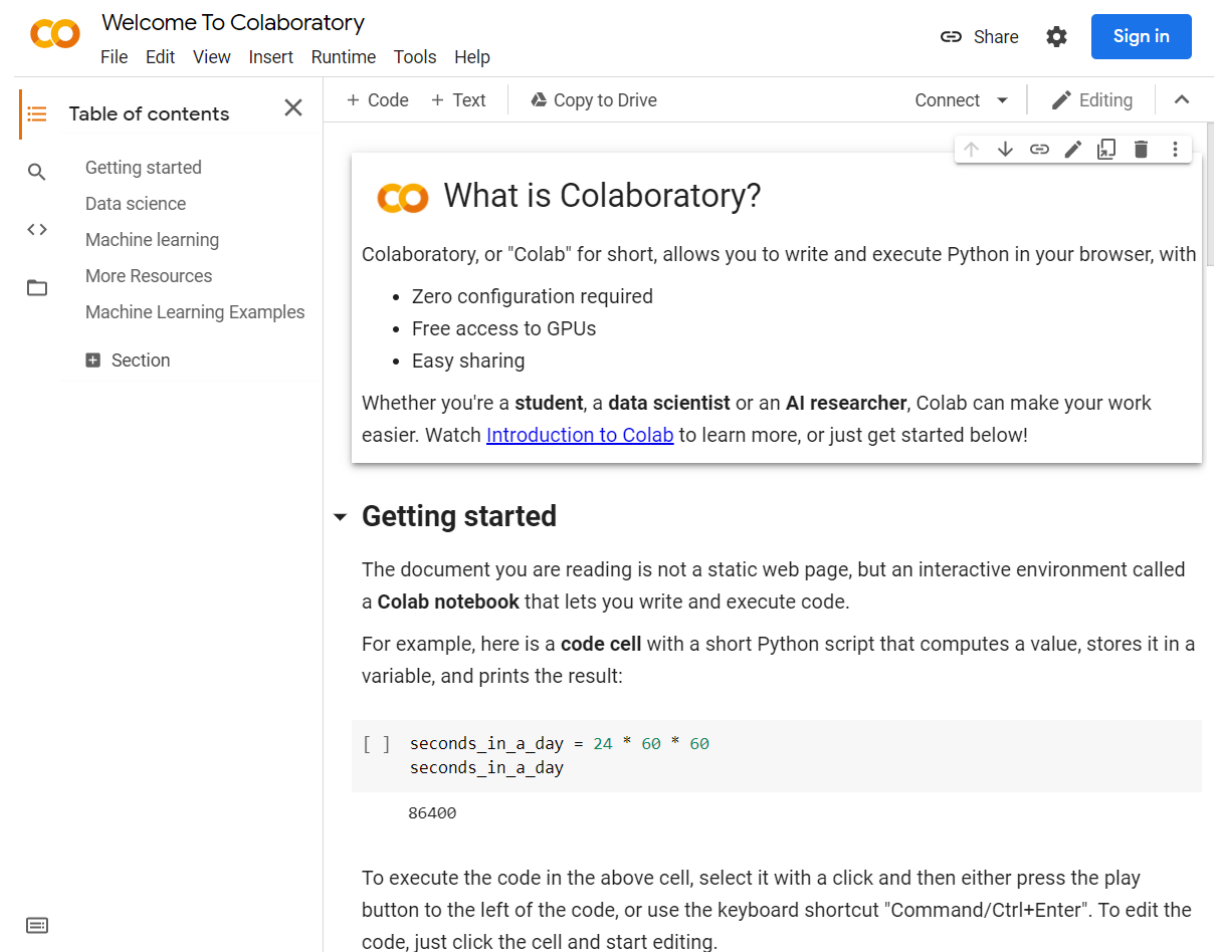
Python is a programming language that lets you work quickly and integrate systems more effectively. It's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Why should you know Python?

In academic and research we can use Python and its lots of libraries to do a wide range of things from basic calculation to complex statistical methods including machine learning. Since it is open-source, it is free to use.

What is Google Colab?

Colaboratory is an online workspace for Python programming, which is developed by the Google Research team. It has several features that improve learning experience, such as Pre-Installed libraries, Stored on the cloud, Linked with Google drive, Collaborative coding, Integrated GPU and TPU, and It is free to use!



The screenshot displays the Google Colaboratory web interface. At the top, there's a header with the Colab logo, 'Welcome To Colaboratory', and navigation links like 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right, there are 'Share', 'Settings', and 'Sign in' buttons. A left sidebar contains a 'Table of contents' with links to 'Getting started', 'Data science', 'Machine learning', 'More Resources', 'Machine Learning Examples', and 'Section'. The main area shows a document titled 'What is Colab?' with the following content:

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

86400

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

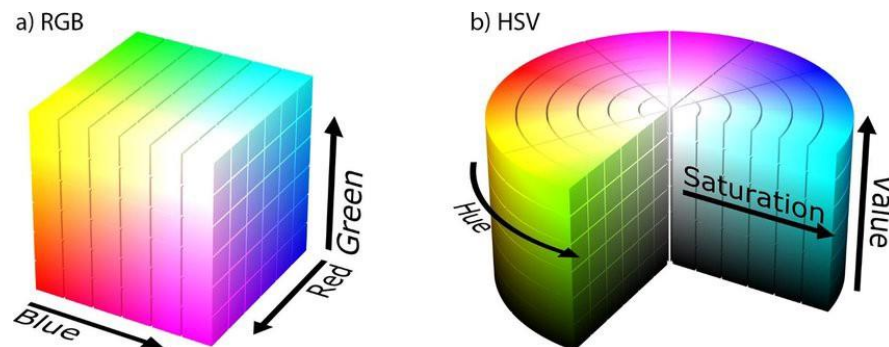
Basic Image Processing

RGB Colorspace

RGB is the most common way to describe the chromaticity of the color. It consists of three values that represent the light intensity of each frequency. For example, R is referred to red color which has a wavelength peaking around 570 nm, G is green having peaking around 540 nm, and B is blue having peaking around 430 nm. By mixing these 3 values, we can generate any colors within the human perception. This color space is also used as the base unit of display monitors.

HSV Colorspace

HSV is an alternative way to explain colors. This color space can be modeled into the cylinder shape, which has the rainbow radial slice as the upper face, pure black as the lower face, and other colors are in between. This model was designed to reflect the way humans describe each color. It is separated into 3 components. H or hue represents color as angle. S or saturation represents expressiveness of color, it is covering from neutral to colorful. V or value represents lightness, it is covering from dark to bright.

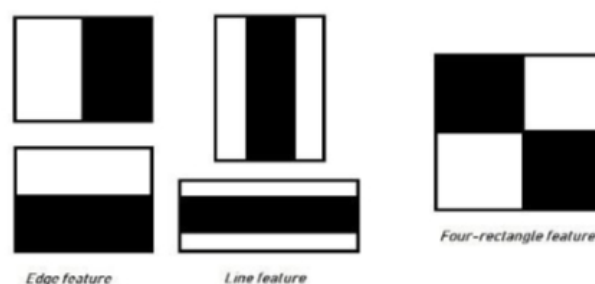


Face Detection

Face detection is one case of object detection. Object detection is one of the computer technologies, which finds the locations and sizes of detecting instances of an object such as human faces, buildings, trees, cars, etc. The main task of Face-detection algorithms is to determine whether an image is a human face or not. There have been a lot of studies in the field of face detection research for a long time. However, the most effective algorithm to be used in general is the method which has been proposed for the first time by Paul viola and Michael J. Jones in 2001. The technique has been accepted and recognized in many face detection research. Since it has the ability to detect faces quickly, effectively and can be used in practical use such as a digital camera or a photo management software.

Viola-Jones face detection algorithm

1) Haar-like features: relevant features



Haar features is a sequence of rescaled squared-shape functions.

0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

ideal **Haar-feature**
pixel intensities

0.1	0.2	0.6	0.8
0.2	0.3	0.8	0.6
0.2	0.1	0.6	0.8
0.2	0.1	0.8	0.9

these are real values
detected on an image

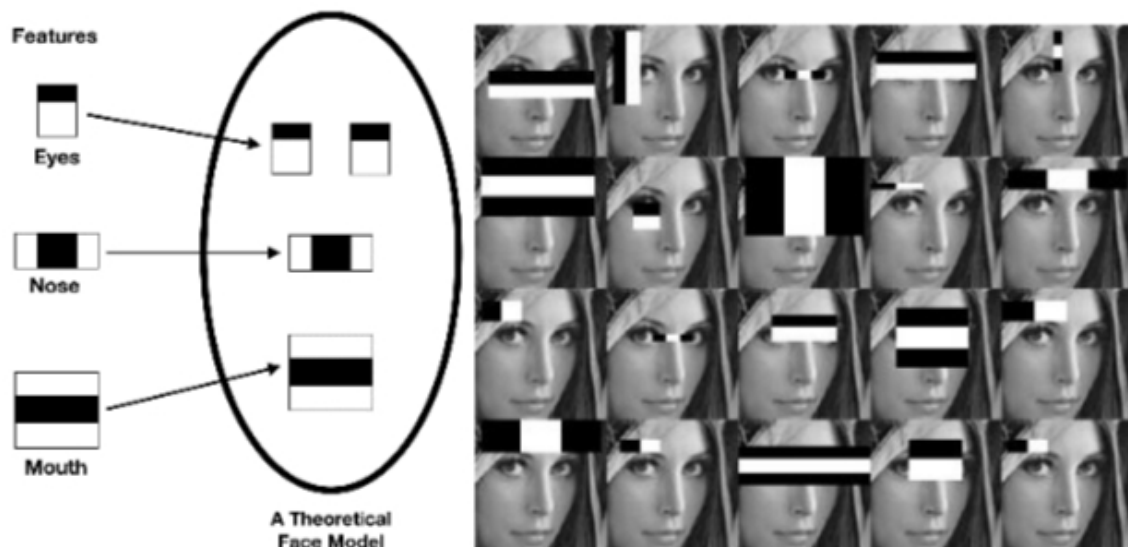
0: white pixel, 1: black pixel

$$\Delta = \text{dark} - \text{white} = \frac{1}{n} \sum_{\text{dark}}^n I(x) - \frac{1}{n} \sum_{\text{white}}^n I(x)$$

$$\Delta_{\text{ideal Haar - feature}} = 1$$

$$\Delta_{\text{real image}} = 0.74 - 0.18 = 0.56$$

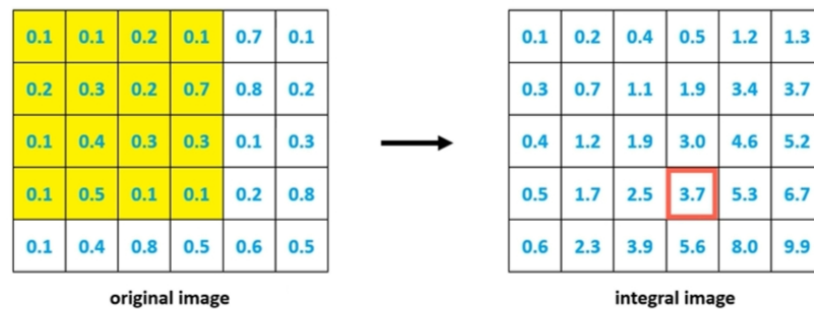
Viola Jones algorithm will compare how close the real scenario is to the ideal case. Each feature results in a single value which is calculated by subtracting the sum of pixels in the light area from the sum of pixels in the dark area. The closer to value 1, the more likely we have found a Haar-feature. We can set the threshold to decide if that area is Haar-feature or not. The values indicate certain characteristics of a particular area of the image.



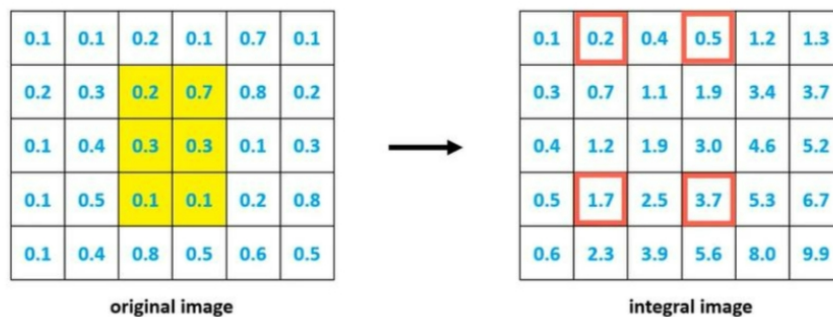
The algorithm is going to use very simple Haar-features to look for the most relevant features as far as human faces are concerned such as the eyes, nose, lips, etc.

2) Integral Image: making the calculation ways faster

In an integral image the value at pixel (x,y) is the sum of all pixels above and to the left.



Sum of the area = 3.7

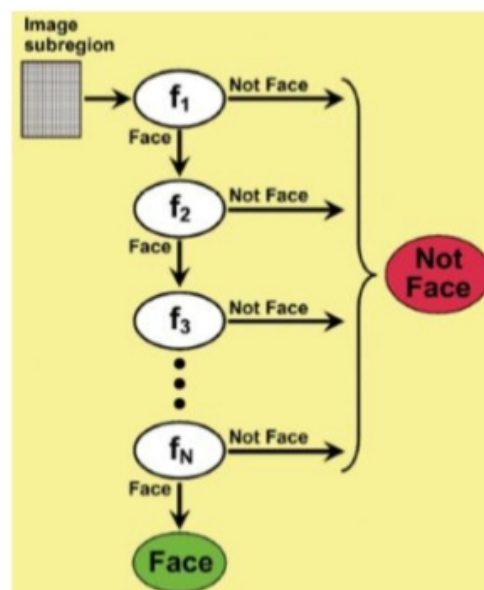


Sum of the area = 3.7 - 0.5 + 0.2 - 1.7

By this method, we can reduce the complexity of calculation.

3) Cascading: are you the face?

Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region



4) Adaboost: finding the best features

Adaboost is a machine learning algorithm which helps in finding only the best features among all 160,000+ features. Initially, the algorithm needs a lot of positive images (face images) and negative images (non-face images) to train the classifier.