基於OpenCV DNN模塊給黑白老照片上色(附Python/C++源碼)

原創 Color Space OpenCV與AI深度學習 2022-02-16 08:05

收錄於話題

#深度學習 18 #OpenCV 63 #計算機視覺 15 #圖像處理 29 #DNN 2

導讀

本文給大家分享一個用OpenCV DNN模塊給黑白老照片上色的實例，並給出Python和C++版本源碼。

背景介紹

這個項目是基於在加利福尼亞大學，伯克利，Richard Zhang，Phillip Isola和Alexei A. Efros開發的研究工作--Colorful Image Colorization，對應論文地址：https://arxiv.org/pdf/1603.08511.pdf，作者項目github地址：https://github.com/richzhang/colorization/tree/caffe

正如在最初的論文中所解釋的，作者們接受了問題的潛在不確定性，將其作為一項分類任務，在訓練時使用類別再平衡來增加結果中的顏色多樣性。人工智能（AI）方法在測試時在CNN（“卷積神經網絡”）中作為前饋傳遞實現，並在100多萬張彩色圖像上進行訓練。

這個項目將使用的顏色空間模型是“Lab”。CIELAB顏色空間（也稱為CIE L*a*b*或有時簡稱為“Lab”顏色空間）是國際照明委員會（CIE）在1976年定義的顏色空間。它將顏色表示為三個數值，L*表示亮度，a*和b*表示綠色、紅色和藍黃色。

深度學習的過程：正如引言中所述，人工智能（AI）方法在測試時作為CNN（“卷積神經網絡”）中的前饋傳遞實現，並在100多萬張彩色圖像上進行訓練。換句話說，數百萬張彩色照片使用Lab模型進行分解，並用作輸入特徵（“L”）和分類標籤（“a”和“b”）。為了簡單起見，我們分成兩部分：“L”和“a+b”，如方框圖所示：

有了經過訓練的模型（可以公開獲得），我們可以用它給一個新的黑白照片上色，這張照片將作為模型或組件“L”的輸入。模型的輸出將是其他組件“a”和“b”，它們一旦添加到原始“L”中，將返回一張完整的彩色照片，如下所示：

简言之，使用ImageNet的130万张照片的广泛多样的对象和场景数据集，并应用深度学习算法（前馈CNN），生成了最终模型，可在以下网址获得：

https://github.com/richzhang/colorization/tree/caffe/models

效果展示

01:09

详细步骤与演示

OpenCV DNN模块可以直接使用Caffe训练好的模型，下面是加载模型测试步骤：

【1】下载模型和配置文件：

```
wget http://eecs.berkeley.edu/~rich.zhang/projects/2016_colorization/files/demo
```

【2】准备测试图片--黑白老照片：

【3】代码测试：

Python OpenCV实现代码：

```
# 公众号：OpenCV与AI深度学习
# Importing libraries
import numpy as np
import matplotlib.pyplot as plt
import cv2

print(cv2.__version__)

# Path of our caffemodel, prototxt, and numpy files
prototxt = "./model/colorization_deploy_v2.prototxt"
caffe_model = "./model/colorization_release_v2.caffemodel"
pts_npy = "./model/pts_in_hull.npy"

img_path = './imgs/11.jpg'
# Loading our model
net = cv2.dnn.readNetFromCaffe(prototxt, caffe_model)
```

```
pts = np.load(pts_npy)

layer1 = net.getLayerId("class8_ab")
print(layer1)
layer2 = net.getLayerId("conv8_313_rh")
print(layer2)
pts = pts.transpose().reshape(2, 313, 1, 1)
net.getLayer(layer1).blobs = [pts.astype("float32")]
net.getLayer(layer2).blobs = [np.full([1, 313], 2.606, dtype="float32")]

# Converting the image into RGB and plotting it
# Read image from the path
test_image = cv2.imread(img_path)
# Convert image into gray scale
test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2GRAY)
# Convert image from gray scale to RGB format
test_image = cv2.cvtColor(test_image, cv2.COLOR_GRAY2RGB)
# Check image using matplotlib
plt.imshow(test_image)
plt.show()

# Converting the RGB image into LAB format
# Normalizing the image
normalized = test_image.astype("float32") / 255.0
# Converting the image into LAB
lab_image = cv2.cvtColor(normalized, cv2.COLOR_RGB2LAB)
# Resizing the image
resized = cv2.resize(lab_image, (224, 224))
# Extracting the value of L for LAB image
L = cv2.split(resized)[0]
L -= 50   # OR we can write L = L - 50

# Predicting a and b values
# Setting input
net.setInput(cv2.dnn.blobFromImage(L))
# Finding the values of 'a' and 'b'
ab = net.forward()[0, :, :, :].transpose((1, 2, 0))
# Resizing
ab = cv2.resize(ab, (test_image.shape[1], test_image.shape[0]))

```

```
# Combining L, a, and b channels
L = cv2.split(lab_image)[0]
# Combining L,a,b
LAB_colored = np.concatenate((L[:, :, np.newaxis], ab), axis=2)
# Checking the LAB image
plt.imshow(LAB_colored)
plt.title('LAB image')
plt.show()

## Converting LAB image to RGB
RGB_colored = cv2.cvtColor(LAB_colored,cv2.COLOR_LAB2RGB)
# Limits the values in array
RGB_colored = np.clip(RGB_colored, 0, 1)
# Changing the pixel intensity back to [0,255],as we did scaling during pre-pr
RGB_colored = (255 * RGB_colored).astype("uint8")
# Checking the image
plt.imshow(RGB_colored)
plt.title('Colored Image')
plt.show()

# Saving the colored image
# Converting RGB to BGR
RGB_BGR = cv2.cvtColor(RGB_colored, cv2.COLOR_RGB2BGR)
# Saving the image in desired path
cv2.imwrite('result.jpg', RGB_BGR)

```

C++ OpenCV实现代码:

```
// ImgColorization_OpenCV_DNN.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并
// 公众号：OpenCV与AI深度学习
#include "pch.h"
#include <iostream>
#include <opencv2/opencv.hpp>
#include <opencv2/dnn.hpp>

using namespace cv;
using namespace cv::dnn;
using namespace std;
```

```
11
12  // the 313 ab cluster centers from pts_in_hull.npy (already transposed)
13  static float hull_pts[] = {
14    -90., -90., -90., -90., -90., -80., -80., -80., -80., -80., -80., -80., -80.
15    -70., -70., -60., -60., -60., -60., -60., -60., -60., -60., -60., -60., -60.
16    -50., -50., -50., -50., -50., -50., -40., -40., -40., -40., -40., -40., -40.
17    -30., -30., -30., -30., -30., -30., -30., -30., -30., -30., -30., -30., -30.
18    -20., -20., -20., -20., -20., -20., -20., -20., -20., -10., -10., -10., -10.
19    -10., -10., -10., -10., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
20    10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 20., 20., 20., 20., 2
21    20., 20., 20., 30., 30., 30., 30., 30., 30., 30., 30., 30., 30., 30., 30., 3
22    40., 40., 40., 40., 40., 40., 40., 40., 40., 40., 40., 40., 40., 40., 40., 4
23    50., 50., 50., 50., 50., 50., 50., 50., 50., 60., 60., 60., 60., 60., 60., 6
24    60., 60., 60., 70., 70., 70., 70., 70., 70., 70., 70., 70., 70., 70., 70., 7
25    80., 80., 80., 80., 80., 80., 80., 80., 80., 80., 80., 80., 80., 80., 80., 8
26    90., 90., 90., 90., 90., 90., 90., 90., 90., 100., 100., 100., 100., 100., 1
27    20., 30., 40., 50., 60., 70., 80., 90., 0., 10., 20., 30., 40., 50., 60., 70
28    60., 70., 80., 90., -30., -20., -10., 0., 10., 20., 30., 40., 50., 60., 70.,
29    30., 40., 50., 60., 70., 80., 90., 100., -50., -40., -30., -20., -10., 0., 1
30    -40., -30., -20., -10., 0., 10., 20., 30., 40., 50., 60., 70., 80., 90., 100
31    30., 40., 50., 60., 70., 80., 90., 100., -70., -60., -50., -40., -30., -20.,
32    100., -80., -70., -60., -50., -40., -30., -20., -10., 0., 10., 20., 30., 40.
33    -40., -30., -20., -10., 0., 10., 20., 30., 40., 50., 60., 70., 80., 90., -90
34    0., 10., 20., 30., 40., 50., 60., 70., 80., 90., -100., -90., -80., -70., -6
35    40., 50., 60., 70., 80., 90., -100., -90., -80., -70., -60., -50., -40., -30
36    80., -110., -100., -90., -80., -70., -60., -50., -40., -30., -20., -10., 0.,
37    -90., -80., -70., -60., -50., -40., -30., -20., -10., 0., 10., 20., 30., 40.
38    -60., -50., -40., -30., -20., -10., 0., 10., 20., 30., 40., 50., 60., 70., -
39    -20., -10., 0., 10., 20., 30., 40., 50., 60., 70., -90., -80., -70., -60., -
40  };
41
42  int main()
43  {
44    string modelTxt = "./model/colorization_deploy_v2.prototxt";
45    string modelBin = "./model/colorization_release_v2.caffemodel";
46    string img_path = "./imgs/10.jpg";
47
48    Mat img = imread(img_path);
49    if (img.empty())
50    {
```

```
    cout << "Can't read image from file: " << img_path << endl;
    return 2;
}
// fixed input size for the pretrained network
const int W_in = 224;
const int H_in = 224;
Net net = dnn::readNetFromCaffe(modelTxt, modelBin);

// setup additional layers:
int sz[] = { 2, 313, 1, 1 };
const Mat pts_in_hull(4, sz, CV_32F, hull_pts);
Ptr<dnn::Layer> class8_ab = net.getLayer("class8_ab");
class8_ab->blobs.push_back(pts_in_hull);
Ptr<dnn::Layer> conv8_313_rh = net.getLayer("conv8_313_rh");
conv8_313_rh->blobs.push_back(Mat(1, 313, CV_32F, Scalar(2.606)));
// extract L channel and subtract mean
Mat lab, L, input;
img.convertTo(img, CV_32F, 1.0 / 255);
cvtColor(img, lab, COLOR_BGR2Lab);
extractChannel(lab, L, 0);
resize(L, input, Size(W_in, H_in));
input -= 50;
// run the L channel through the network
Mat inputBlob = blobFromImage(input);
net.setInput(inputBlob);
Mat result = net.forward();
// retrieve the calculated a,b channels from the network output
Size siz(result.size[2], result.size[3]);
Mat a = Mat(siz, CV_32F, result.ptr(0, 0));
Mat b = Mat(siz, CV_32F, result.ptr(0, 1));
resize(a, a, img.size());
resize(b, b, img.size());
// merge, and convert back to BGR
Mat color, chn[] = { L, a, b };
merge(chn, 3, lab);
cvtColor(lab, color, COLOR_Lab2BGR);
imshow("color", color);
imshow("original", img);
waitKey();
return 0;
```

```
91  }
```

【4】效果展示：

下载1：Pytoch常用函数手册

在「**OpenCV与AI深度学习**」公众号后台回复：**Pytorch常用函数手册**，即可下载全网第一份Pytorch常用函数手册，涵盖**Tensors介绍、基础函数介绍、数据处理函数、优化函数、CUDA编程、多线程处理**等十四章章内容。

下载2：145个OpenCV实例应用代码

在「**OpenCV与AI深度学习**」公众号后台回复：**OpenCV145**，即可下载145个OpenCV实例应用代码(**Python和C++双语言实现**)。

—THE END—

CV相关内容讨论交流欢迎加入微信交流群！

觉得有用，记得点个赞和在看哟

收录于话题 #深度学习 18

下一篇 · 如何在OpenCV DNN模塊中使用NVIDIA GPU加速--(基於Windows)

喜歡此內容的人還喜歡

Opencv實現透視形變

小白學視覺

你能用OpenCV做什麼

新機器視覺