Case Study Using OpenCV for Image Recognition

Table of Contents

I - Introduction

The following case study was built to demonstrate the potential of utilizing OpenCV in addition to Python along with multiple other libraries, packages and modules used in the process. The purpose of our case study is to intend to build off of the Boss Sensor repository to detect images and to experiment, attempt to improve accuracy and learn useful information along the way.

OpenCV is a core piece of technology that is being used in a range of industries including autonomous vehicles, security operations and much more. Facial recognition and image detection will play an integral part of our study and by completing it we hope to identify the main parameters to be able to recognize an incoming person via the computers webcam that will then allow us to take further decisions such as loading an image or shutting down processes.

Our overall goal is to detect the designated “boss” image quickly.

Licensing

The following case study was built using the Boss Sensor repository created by the author Hironsan that can be found here https://github.com/Hironsan/BossSensor, fully MIT licensed https://github.com/Hironsan/BossSensor/blob/master/LICENSE

II - Setup

Setup may vary slightly depending on the operating environment. Please see specific sections IE Ubuntu Setup for more specific instructions. Requirements include the following for setup. Please add them in a contained or virtual environment that you will create with the instructions within anaconda.

Requirements include the following

- Python 3.5
- OSX (currently in testing for Ubuntu)
- Anaconda
- OpenCV
- PyQt4
- H5py
- Scipy
- Sklearn
- Keras
- Tensorflow

III - Installation Instructions General

Please add a folder and call it Store. Leave it in the main directory.

Moreover, please use the following commands when the file structure is in order.

1. Conda create –n (name of virtual environment you want to use) python=3.5
2. Source activate (name of virtual environment you set up)
3. Conda install –c https://conda.anaconda.org/menpo opencv3
4. Conda install –c conda-forge Tensorflow
5. Pip install –r requirements.txt

Additional packages may be needed as mentioned above and you can install them via anaconda or pip (for example pip install Tensorflow) when you are working in your environment.

IV - Installation Instructions for Ubuntu

Please add the following piece of code to the camera_reader.py file:

If you run it on Ubuntu and it doesn't open the camera please see the following:

1. Change VideoCapture(1) to VideoCapture(0) – test camera_reader.py
2. If no success, please add a print statement to the following after ret, frame= cap.read() add print(ret). This will return a false statement in an infinite loop if currently encountering problems. Test – Does it return false? If so, proceed to step 3.
3. Try in the terminal v4l2-ctl –list-devices to see a list of all available devices. Do you see an output with available devices? If yes, continue to step 4.
4. Check if camera is on place with ffmpeg –f v4l2 –list_formats all –I (path from step 3)
5. Try the following: ffmpeg –I (path again, for example /dev/video0) output.mkv and wait for a short period of time (10-20 seconds) and then press Ctrl-C. Look for the video at f.e mplayer output.mkv or go to your video folder on your system and check. Is it there and does it play? If yes, continue to step 6.
6. In terminal please run the command groups. Do you see the group video? If no, please run under the root account: adduser (yourLogin) video. After adding group video please logout and log back in. Type groups again and see if video is added. If video is added try to run the camera_reader.py script again and it should now start trying to detect the "boss".

*Please make sure that your environment has no conflicts and openCV has been installed without a problem (only have one version of openCV installed).

VI - Debug Solutions

Debug solutions if error occurs:

- Be careful in the order of operations that are run
- Boss train should be executed first followed by camera_reader.py
- If error occurs relating to tensorflow (when set as the backend) add the following to the file of boss_train.py: import tensorflow as tf, tf.python.control_flow_ops = tf.

For Ubuntu troubleshooting use for pyqt
Sudo apt-get install python-qt4

```
Conda install pyqt=4
```

If receiving an error such as the following:

```
cv2.error: /Users/jenkins/miniconda/1/x64/conda-
bld/conda_1486587097465/work/opencv-
3.1.0/modules/objdetect/src/cascadedetect.cpp:1639: error: (-215) !empty() in
function detectMultiScale
```

This error is most likely due to the xml file path either due to installation or setup. You can set in a hard or direct path by downloading the files from opencv directly at https://github.com/opencv/opencv/tree/master/data/haarcascades and find the frontalface that you can download and import into your working directory.

VII - Running the program

- * Take a picture of the background that you would like. For example, take a picture or a screen shot on what your normal work environment or your background on the computer. Save it and add it to the folder and title it my_work_background.jpg. This is the file that will load when the "boss" is detected.
- Add your files into the two folders in the folder data. Add your "boss" subject to the folder boss and add the other images, normal image background or a different person to other.
- Once images are in the folders you can now run the boss_train.py file. This will train on the images, run through 10 epochs and save the model with an accuracy score. Once this is complete move on to the following step.
- Run the camera_reader.py file. This is the detector file that will open your camera and use the camera to scan for the designated "boss". For the purpose of the case study I designated myself as the boss and used other random images for my other category to train on.
- I started with about 10 images with no results.
- Increased to 30 images – still no results
- Increased image set to around 100 and was successfully detected (and fast). Can the accuracy and rate be improved?
- Depending on the training and your environment around 1000-1200 images are recommended for both the designated "boss" and for the other category. Again using my work environment and image backgrounds I was able to detect quickly with around 100 images.
- * If you add new photos or make changes remember to re-train your model with the boss_train.py file.

VII – File Changes

- For Ubuntu users change VideoCapture(1) to VideoCapture(0)
- Add the following print statement if running into errors to check the program – in camera_reader.py after ret, frame= cap.read() add print(ret).
- If adding the xml file directly (haarscascade_frontalface_default.xml) add the file path of cascade_path as cascade_path = "haarcascade_frontalface_default.xml"

IX – Overview

- The program does a great job detecting the designated "boss" by using OpenCV for image and facial recognition. It's a great approach that allows the user to change information in a simple format.
- Boss successfully detected with higher accuracy results the closer to the camera that the boss was located.
- Upon experimentation, using more photos (as mentioned earlier in the document between 1000-1200 is optimal) but I have had successful results with around 100 images.
- Some ideas for future improvements would be to work on detecting quicker (from a close range, for example a seated position detects fast but optimally it would need to spot from say a hallway or path across the room).
- Build with more epochs to analyze results and compare
- Test with a more dynamic environment such as a busy office with multiple faces being analyzed at once