Face Detection And Recognition

The Software module is designed and developed to be applied for Real time mode of detection and recognition of face for example in door bell/door lock access using deep learning and OpenCV together (with scikit-learn libraries)

Getting Started

- Build the image dataset itself.
- Detect faces
- Implement a CNN capable of performing liveness detector (we'll call this network "LivenessNet").
- Compute 128-d face embeddings to quantify a face
- Train the liveness detector network.
- Train a Support Vector Machine (SVM) on top of the embeddings
- Recognize faces using our trained liveness detector model, Face Recognizer model and apply it to real-time video.

Prerequisites

- Python 3.6.X
- OpenCV 3.4.2 or higher
- Numpy
- Imutils
- scikit-learn
- keras
- TensorFlow

Project Structure And Installing

```
## Project Structure And Installing

├── dataset
│   ├── mahek [images]
│   ├── ashish_sir [images]
│   └── jay [images]
├── dataset_live
│   ├── real[image]
│   └── fake[image]
├── face_detection_model
│   ├── deploy.prototxt
│   ├── res10_300x300_ssd_iter_140000.caffemodel[Deep learning model]
│   └── haarcascade_frontalface_default.xml[Opencv Frontalface detector model]
├── output
│   ├── embeddings.pickle[image matrix]
│   ├── le.pickle[label encoder]
│   ├── label.pickle[label encoder]
│   └── recognizer.pickle
├── live_imagesearch
│   ├── __init__.py
│   └── livenessnet.py[Liveness Library]
├── videos
│   ├── real.mp4[Recorded a ~25-second video]
│   └── fake.mp4[Replayed the same 25-second video, this time facing Phone towards desktop]
├── build_face_dataset.py
├── extract_embeddings.py
├── Face_Recognition_liveness.py
├── openface_nn4.small1.v1.t7[Torch Model][Face_Recognition][Deep learning model]
├── openface_nn4.small2.v1.t7[Torch Model][Face_Recognition][Deep learning model]
├── train_model.py
├── gather_examples.py
├── train_liveness.py
├── liveness.model[Liveness Model]
└── plot.png[Histogram]
```

Project Structure

There are quite a few moving parts for this project — take the time now to carefully read this section so you become familiar with all the files in this project.

Our project has six directories in the root folder:

dataset/ : Contains our face images organized into subfolders by name.

dataset_live/ : Our dataset directory consists of two classes of images:

* Fake images from a camera aimed at screen while playing a video of face.

* Real images from a selfie video with phone.

face_detection_model/ : Contains a pre-trained Caffe deep learning model provided by OpenCV to detect faces. This model detects and localizes faces in an image.

output/ : Contains output pickle files. If you're working with your own dataset, you can store your output files here as well. The output files include:

- embeddings.pickle : A serialized facial embeddings file. Embeddings have been computed for every face in the dataset and are stored in this file.
- le.pickle : Our label encoder. Contains the name labels for the people that our model can recognize.
- label.pickle : Our liveness class label encoder.
- recognizer.pickle : Our Linear Support Vector Machine (SVM) model. This is a machine learning model rather than a deep learning model and it is responsible for actually recognizing faces.

live_imagesearch/ : This module contains our LivenessNet class.

videos/ : Provide two input videos for training our LivenessNet classifier.

Running the tests

Step #1

build_face_dataset.py : Dataset is created by capturing the image containing face with different positions with different poses and scales to provide reasonable accuracy. We may even perform this process over multiple days or weeks to gather examples of their face in:

- Different lighting conditions
- Times of day
- Moods and emotional states

Step #2

gather_examples.py : This script grabs face ROIs from input video files and helps us to create a deep learning face liveness dataset.

Step #3

extract_embeddings.py : Step #3 which is responsible for using a deep learning feature extractor to generate a 128-D vector describing a face. All faces in our dataset will be passed through the neural network to generate embeddings.

Step #4

train_liveness.py : As the filename indicates, this script will train our LivenessNet classifier. We'll use Keras and TensorFlow to train the model. The training process results in a few files:

- label.pickle : Our class label encoder.
- liveness.model : Our serialized Keras model which detects face liveness.
- plot.png : The training history plot shows accuracy and loss curves so we can assess our model (i.e. over/underfitting).

Step #5

train_model.py : Our Linear SVM model will be trained by this script in Step #2. We'll detect faces, extract embeddings, and fit our SVM model to the embeddings data.

openface_nn4.small1.v1.t7 & openface_nn4.small2.v1.t7 : A Torch deep learning model which produces the 128-D facial embeddings. We'll be using this deep learning model in Steps #3, #5, and #6 as well as the Bonus section.

Step #6

Face_Recognition_liveness.py: Our demonstration script will fire up your webcam to grab frames to conduct face liveness detection and recognize who is in frames in real-time.

Authors

Mahek Vasoya - Face Detection And Recognition

Jay Thakkar

Chandresh Maniya

Poojan Sutariya

Built With

Anaconda Navigator- Anaconda Navigator is a desktop graphical user interface (GUI) (https:/docs.anaconda.com/anaconda/navigator/)

Spyder- Spyder is a powerful scientific environment(IDE)(https://www.spyder-ide.org/)