Assignment 1: Huấn luyện bộ phân lớp phân tầng Viola - Jones để phát hiện khuôn mặt

Giới thiệu

Sử dụng cửa sổ trượt kết hợp với một bộ phân lớp là một cách phổ biến để phát hiện vật thể (object) trong ảnh. Nguyên lý cơ bản của thuật toán là phân lớp từng vùng ảnh nhỏ (image patch) có chứa vật thể hay không. Phát hiện khuôn mặt cũng là một bài toán phát hiện vật thể đã đạt được nhiều thành công trong ngành thị giác máy. Ví dụ, các camera hiện đại hay các phần mềm xem ảnh hiện nay đều có khả năng phát hiện khuôn mặt. Sự thành công này phải kể đến bộ phân lớp phân tầng sử dụng đặc trưng Haar của Viola - Jones. Ở bài tập này anh/chị sẽ thực hiện huấn luyện một bộ phân lớp phân tầng dựa trên giải thuật Adaboost của Viola & Jones để phát hiện khuôn mặt trong ảnh. Anh/chị sẽ sử dụng các công cụ có sẵn của thư viện OpenCV để thực hiện việc huấn luyện.

Các file cần thiết cho assignment1 có thể tìm thấy tại đây. Anh/chị cần cài đặt trước python và OpenCV, hướng dẫn xem tại đây

Code

- **`CascadeClassifierTraining.ipynb`**: Jupyter notebook giúp anh/chị thực hiện từng bước của quá trình huấn luyện và thử nghiệm mô hình huấn luyện được trên một ảnh chứa mặt người.
- **`train_cascade/prepare_negative.py`**: chứa đoạn mã python cho phép chuẩn bị các mẫu negative (mẫu không chứa mặt người)
- **`train_cascade/prepare_positive.py`**: chứa đoạn mã python cho phép chuẩn bị các mẫu positive (mẫu chứa mặt người)
- **`train_cascade/train_cascades.py`**: chứa đoạn mã python để sinh ra file vecfile chứa thông tin dạng nhị phân của mẫu positive và đoạn mã huấn luyện bộ phân lớp
- **`train_cascade/config.py`**: chứa các thiết lập để huấn luyện bộ phân lớp. Anh/chị sẽ chỉnh sửa chính trên file này
- **`prepare_samples.py`**: chạy file này bằng lệnh `python prepare_samples.py` sẽ chạy các đoạn mã chuẩn bị mẫu negative và positive
- **`train_cascade.py`**: chạy file này bằng lệnh `python train_cascade.py` sẽ chạy đoạn mã huấn luyện bộ phân lớp.

Dữ liệu

Dữ liệu là một phần cực kỳ quan trọng trong bài tập này cũng như trong bất kỳ bài toán học máy nào khác. Bài tập này về cơ bản là huấn luyện các bộ phân lớp với hai lớp: mặt và không phải mặt. Vì vậy, anh/chị sẽ phải chuẩn bị hai loại dữ liệu: các ảnh chứa mặt (mẫu positive) và các ảnh không chứa mặt (mẫu negative).

- Mẫu positive: Ở bài tập này, anh/chị cần phát hiện các khuôn mặt nhìn từ chính diện (frontal face), vì vậy ảnh huấn luyện cần phải là các ảnh mặt chính diện. Nên sử dụng các ảnh có độ rõ ràng nhất

định và được cắt riêng vùng ảnh chứa mặt. Anh/chị có thể lấy ảnh mặt người từ các cơ sở dữ liệu nguồn mở như LFW, Caltech WebFaces và nhiều nguồn khác
- Mẫu negative: là những bức ảnh bất kỳ mà không chứa khuôn mặt. Anh/chị có thể tìm thấy các ảnh này trong bộ cơ sở dữ liệu ảnh ImageNet hoặc SUN scene database.

Công việc

Mục tiêu của bài tập là huấn luyện được một bộ phân lớp Viola - Jones để phát hiện khuôn mặt với hiệu năng tốt: phát hiện nhầm ít (False Alarm Rate thấp) và ít bỏ sót (Hit Rate cao). Để làm được điều đó, anh/chị phải thu thập thêm dữ liệu huấn luyện và thử với các tham số huấn luyện khác nhau. Đầu tiên, anh/chị chạy Jupyter notebook **`CascadeClassifierTraining.ipynb`** để huấn luyện một bộ phân lớp với các tham số mặc định và một số lượng nhỏ dữ liệu (100 ảnh positive và 100 ảnh negative). Kết quả kiểm thử của bộ phân lớp mới huấn luyện này trên một ảnh mẫu là tương đối kém, phát hiện nhầm rất nhiều. Công việc của anh/chị:

Thêm dữ liệu

Thêm ảnh positive vào thư mục **`data/input_positive`** và thêm ảnh negative vào thư mục **`data/input_negative`**.

Căn chỉnh tham số huấn luyện

Anh/chị cần mở file **`train_cascade/config.py`** để căn chỉnh các tham số huấn luyện. Các tham số có thể căn chỉnh là:

- `WIDTH`: chiều rộng của cửa sổ dịch
- `HEIGHT`: chiều cao của cửa sổ dịch
- `MAX_NUM_POS`: số lượng ảnh positive tối đa để tạo vecfile (số lượng ảnh trong thư mục input_positive)
- `NUM_POS`: số lượng ảnh positive dùng để huấn luyện tại mỗi tầng (cần phải nhỏ hơn MAX_NUM_POS, nếu khi huấn luyện gặp lỗi, thử giảm NUM_POS)
- `NUM_NEG`: số lượng ảnh negative dùng để huấn luyện tại mỗi tầng
- `NUM_STAGES`: số lượng tầng của bộ phân lớp sẽ huấn luyện
- `MAX_FALSE_ALARM_RATE`: Maximal desired false alarm rate for each stage of the classifier. Overall false alarm rate may be estimated as (max_false_alarm_rate ^ number_of_stages)
- `MIN_HIT_RATE`: Minimal desired hit rate for each stage of the classifier. Overall hit rate may be estimated as (min_hit_rate ^ number_of_stages)
- `FEATURE_TYPE`: Type of features: HAAR - Haar-like features, LBP - local binary patterns.
- `BOOST_TYPE`: Type of boosted classifiers: DAB - Discrete AdaBoost, RAB - Real AdaBoost, LB - LogitBoost, GAB - Gentle AdaBoost
- `EXTRA_PARAMS`: Các tham số khác, xem chi tiết tại đây.

Sau khi sửa các tham số huấn luyện, anh/chị có thể dùng Jupyter Notebook **`CascadeClassifierTraning.ipynb`** thực hiện huấn luyện. Kết quả của quá trình huấn luyện sẽ là file đuôi `.xml` chứa các thông tin về bộ phân lớp.

Lưu ý

Khi dùng file Jupyter Notebook **`CascadeClassifierTraning.ipynb`** để huấn luyện, sau khi sửa xong file **`config.py`**, code có thể chưa được update dẫn đến việc anh/chị huấn luyện lại model với tham số cũ. Để tránh việc đó, anh/chị có thể không dùng Jupyter Notebook để huấn luyện, thay vào đó chạy trực tiếp các file `.py` tại thư mục gốc.

Chuẩn bị dữ liệu

Chạy lệnh sau để chuẩn bị dữ liệu. Nếu không có thay đổi dữ liệu và HEIGHT, WIDTH của cửa sổ dịch thì chỉ cần chạy lệnh này lần đầu.

```
python prepare_samples.py
```

Huấn luyện cascade classifier

Dùng lệnh sau để huấn luyện.

```
python train_cascade.py
```

Đánh giá

1. (+20 pts) Thêm training data, ít nhất 100 ảnh mỗi lớp. Mô tả cách lấy dữ liệu, xử lý dữ liệu (lấy từ nguồn nào, bao nhiêu ảnh, có crop, cân bằng sáng, chuyển sang ảnh xám không?)
2. (+20 pts) Thay đổi tham số WIDTH, HEIGHT (kích thước của ảnh khuôn mặt). So sánh thay đổi kết quả của hệ thống phát hiện khuôn mặt (phải chỉ được thời gian huấn luyện lâu hơn bao nhiêu, độ chính xác thay đổi thế nào,...).
3. (+20pts) Thay đổi tham số MAX_FALSE_ALARM_RATE. So sánh kết quả (phải chỉ ra được khi giảm False Alarm Rate thì Hit Rate thay đổi thế nào?).
4. (+10pts) Thay đổi loại đặc trưng sử dụng FEATURE_TYPE (HAAR, LBP). So sánh các kết quả thực hiện (thời gian huấn luyện, độ chính xác) và nhận xét về hiệu quả phân lớp của từng loại đặc trưng. Có thể kết hợp các đặc trưng này không?
5. (+10pts) Thay đổi phiên bản khác nhau của giải thuật BOOST_TYPE. So sánh các kết quả thực hiện.
6. (+20pts) Vận dụng các kiến thức đã học để huấn luyện được file .xml có Hit Rate cao và False Alarm Rate thấp.
7. Extra Credit: thay đổi các tham số khác và so sánh. (+10pts)

Nộp kết quả

Kết quả bao gồm:

- File văn bản chứa các báo cáo của từng mục
- File XML của bộ phân lớp tốt nhất huấn luyện được.