ICV Assignment #4

전기정보공학부

2016-13343 유상윤

1. Histograms of Oriented Gradients for Human Detection

설계 시 유의점:

svm train:

neg sample을 무작정 많이 뽑는 것보다 오히려 false positive box들을 neg sample에 포함시키는 것이 성능 향상에 큰 도움이 되었다. 하지만 반복할수록 precision은 높아지지만 false negative가 높아지면서 첫번째 그래프에서 꼬리가 없어지는 구조가 나오게 되었다. 이를 약 5회 정도 반복해 아래와 같은 결과를 얻어 내었다.

(boost_svm.py)

hog 벡터 정규화 (min,max)를 (0,1) scale로 정규화해 성능을 향상시켰다.

detector:

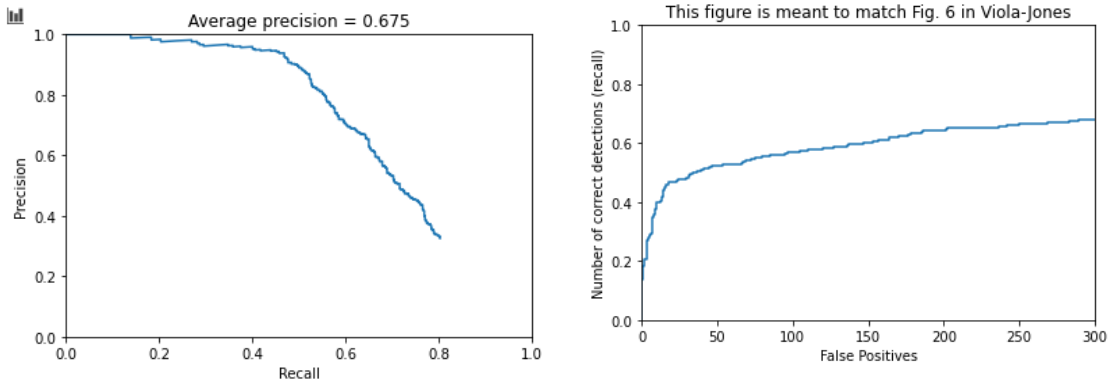
sliding window의 window 별 간격이 어느 정도 촘촘하지 않으면 ground truth를 놓치는 경향이 있었다. 따라서 간격을 가로세로 모두 4에 맞추었다.

scale의 정도 또한 촘촘하게 설계해야 놓치지 않아, 기준인 (64,128)에서 20% 씩 가로세로를 키워 image를 넘어갈 때까지 detect했다.

nms:

non-maximum-supression을 통해 중복을 제거했다. 단, 촘촘하게 detect하는 만큼 겹치는 정도 (intersection/union)을 0.4 이상일 때 동일한 box로 판단해 제거했다.

결과:



<evaluate_detections 결과 그래프>

< detection을 이미지 위에 표시한 것 >

2. Fast Human Detection Using a Cascade of Histograms of Oriented Gradients

설계 시 유의점:

각 block 구조:

```
block_sizes = [] # (w,h)
for i in range(12, 65, 4):
    for (w,h) in [(i,i), (i,2*i), (2*i,i)]:
        if w <=32 and h <= 64:
            block_sizes.append((w,h))

block_strides = []
blocks = []
for w,h in block_sizes:
    for x in range(0,65-w,4):
        for y in range(0,129-h,4):
            blocks.append(((x,y,x+w,y+h)))
```

논문과 비슷한 로직으로 4059개의 block을 사용해서 설계했다.

adaboost 알고리즘 구현:

adaboost 적 관점에서 best clf를 고르기 위해서 남은 sample들의 정답과의 오차의 정도를 weight로 주었다.

```
sample_weight = np.square(sample_confidences - sample_labels)+0.1
```

이 sample weight를 기반으로 svm을 학습한 후,

```
score = 1 - np.dot((sample_labels != sample_pred), sample_weight)/np.sum(sample_weight)
```

sample weight만큼 score에 영향을 주는 scoring function을 통해 score가 가장 높은 svm을 뽑아 layer 에 추가했다.

위 2가지를 하지 않았을 때는 layer당 svm이 많게는 40개 이상까지 쌓인 반면 위 2가지를 통해 적절한 svm을 찾아 각 layer를 가볍게 유지할 수 있었다.

성능 향상:

이외에 성능을 향상시키기 위해 1번과 동일한 과정을 진행했다. 단 false positive에서 바로 hog vector를 뽑아내 dataset에 추가하는 것이 block 별로 달라 까다로워서 false positive를 저장 후 data_preprocess 파일들의 path를 맞추어 추가로 생성 후 별도로 concat하는 과정을 거쳤다. 이 때문에 1번 만큼 실시 할 수 없었다. 따라서 성능 차이가 나는 것으로 보인다.

time 비교, 모델 분석:

```
import time
start_time = time.time()
path='../../data/INRIAPerson/Test/pos/person_085.png'
img = cv2.imread(path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

cur_bboxes, cur_confidences = multi_scale_detector(img, clf, 0.4)
print(time.time() - start_time)
```

single svm(1번) : 약 16초,

cascade(2번): 약 75초

대략 5배 정도 오히려 느려졌다.

원인 분석:

Integral Histograms of Orientated Gradients을 적용하지 못해서 그런 것으로 보인다.

매번 새로운 hog descriptor와 hog vector를 계산하다 보니 연산량이 많아졌다

layer 당 svm 수 (총 8 layer)

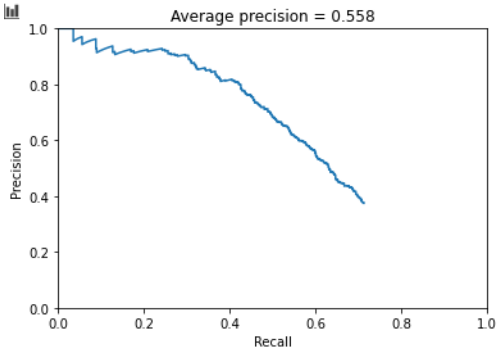
➔ 3 5 7 7 9 6 3 3

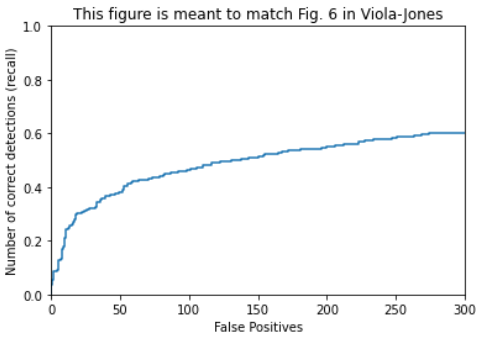
```
{'blocks': [(24, 0, 46, 44), (24, 0, 44, 40), (16, 36, 36, 56)],
 'block_idxs': [1947, 1675, 1388],
 'clfs': [LinearSVC(class_weight='balanced'),
  LinearSVC(class_weight='balanced'),
  LinearSVC(class_weight='balanced')],
 'alphas': [1.10309662365949, 1.4603058215429414, 0.7841813891924564],
 'threshold': 0.23199999999999932}
```

그림 1 layer 0 구조

layer 별 svm 숫자가 적고 모델이 8 layer로 깊어 Integral Histograms of Orientated Gradients를 적용한다면 시간 또한 최적화할 수 있을 것으로 판단된다.

결과:





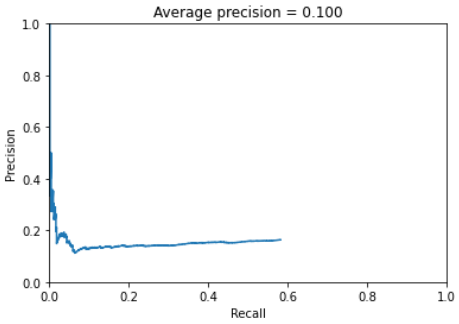
만약 false positive를 더 많이 학습시킨다면 1번과 비슷한 수준의 성능까지 향상될 것으로 기대된다.

3. Discriminative local binary patterns for human detection in personal album

`skimage` 패키지의 `lbp` 를 사용했다.

예상보다 train, test 시간이 오래 걸려 cascade에 대해서는 학습을 하지 못하고 cascade가 아닌 모델만 학습 후 일부 데이터셋에 대해서만 evaluation 해보았다.

1,2 번을 최적화한 것만큼 벡터 정규화, threshold 조절 등의 측면에서 최적화를 완성하지 못해 성능은 낮지만 동작하는 것을 확인할 수 있었다. 최적화를 한다면 성능이 향상될 것으로 보인다.



4. How to Improve

LBP와 hog를 같이 ensemble 해 사용하면 성능이 높아질 것으로 보인다.

정량적인 분석은 lbp가 완전히 완성되지 못해 부족하지만 정성적으로 일부 detection들을 비교해 보았을 때, 각 모델이 어느정도 상호보완적인 것을 확인할 수 있었다.

또한 false positive를 반복학습하는 과정을 수렴할 때까지 진행하지 못했는데 이를 수렴 시까지 높이면 성능이 더욱 향상될 것으로 기대된다. false positive를 학습해 neg sample을 뽑는 것만큼 pos sample을 늘리기 위해 augmentation을 적용해 pos sample의 양을 키우는 것 또한 성능 향상에 도움이 될 것으로 생각한다.

또한 2번 과 3번의 경우 1번에서 진행한 것과 같은 최적화(threshold, 등의 파라미터) 튜닝 과정을 거친다면 성능이 추가로 향상될 것으로 기대된다.