Auto Logo Detection

CUAI 4기 스마트팩토리 A팀
김민구(기계공학), 서혜련(기계공학), 신선우(컴퓨터예술), 이예진(산업보안)

[요약]
기존 방법에서는 영상에서 상표를 가리려면 하나씩 모자이크 처리를 하는 반면 이를 효율적으로 처리하기 위해, 본 연구에서는 딥러닝을 활용한 객체 검출 방식을 도입하여 자동화된 방법을 통해 모자이크 처리를 용이하게 했다.
YOLOv3모델을 이용해 modeling을 진행했고 bounding box 좌표를 통해 최종적으로 영상의 상표를 모자이크 하는 것에 성공했다.

1. 서 론

방송에서는 특정 담배, 흉기 등의 유해물질과 각종 브랜드, 그리고 일반인의 얼굴 노출을 금한다. 그리고 이를 처리하기 위해 현재 사용되는 방식은 영상편집 프로그램에서 일일이 객체에 모자이크를 씌우는 것이다. 우리 조는 이를 보다 효율적으로 해결하기 위해 딥러닝을 활용한 객체 검출 방식을 도입하여 자동으로 영상의 상표를 모자이크 처리할 수 있도록 하고자 하였다.

2. 본 론

1) 이론 및 선행연구

기존의 영상편집 프로그램에서의 모자이크 처리방식은 크게 두가지로 행해졌다. 원하는 영역을 지정하여 프레임별로 모자이크 처리하는 방식과 영상의 특정 객체를 지정하여 추적하는 방식으로 이는 각각 해당 영역을 지속적으로 갱신해야 하는 것, 그리고 객체운동성이 높을수록 정확도가 현저히 떨어진다는 단점이 드러난다.[1]

이러한 한계를 극복하기 위해 딥러닝 기반 객체 검출을 사용한다. 딥러닝 기반의 객체 검출은 1-stage detector와 2-stage detector로 나뉜다.[2] 2-stage detector은 region proposal과 classification의 단계가 명확하게 구별된 구조로 R-CNN 계열의 알고리즘이 이에 해당한다.

이때 2-stage detector은 1-stage detector에 비해 속도가 느리다는 단점이 있어, 이를 보완하기 위해 region proposal과 classification이 동시에 이루어지는 구조를 가진 1-stage detector을 사용하면 객체 검출의 속도와 정확도 모두를 얻을 수 있다.

YOLO[3] 계열의 알고리즘이 여기에 포함되며 쉽게 실시간 객체 검출이 가능한 YOLOv3모델을 활용한다.

2) Dataset – collection

본 연구에서는 상표를 모자이크 하기 위해 미국의 'nike' 브랜드 로고를 통해 진행하였다. 'Web Crawling' 기술을 이용해 브랜드 로고가 있는 이미지를 약 5000개 가량 수집했고, 'labelimg'를 통해 로고를 2개의 class로 분류하여 annotation을 진행하였다. Web Crawling 특성상 해상도가 떨어지며 정확한 로고를 나타내지 못하는 이미지를 포함했기 때문에 이들을 제외하고 labeling을 진행하였다. 따라서 label이 존재하지 않는 이미지 삭제 코드를 구성해 정리하는 작업을 거쳤다.
'labelimg'을 통해서는 'coco format'[4]을 만들 수 없기 때문에 확장자가 .xml인 'Pascal VOC'[5]으로 만든 후 'voc2coco.py'를 이용해 'coco format'으로 변환을 거쳤다.

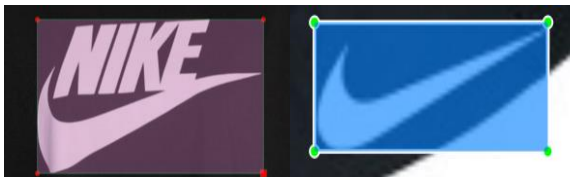


그림 1. Labeling 예시

3) Dataset – preprocessing

1차적으로 수집 및 분류된 데이터는 총 3280개이며 로고만 있을 경우 'nike', 로고 및 영문을 포함할 경우 'nike_txt' 총 2개의 class로 진행했다. Training 및 Validation, Test data의 비율을 80/15/5 으로 설정하여 진행했으며 각각 2624개, 492개, 164개로 구성되도록 하였다.

모델의 성능을 향상시키기 위해 'Augmentation'기법[6,7]을 사용했다. 본 기법을 이용해 데이터의 수를 직접 늘리는 것은 비용과 시간이 많이 필요하기 때문에 인위적인 데이터를 만드는 방식으로 데이터의 over-fitting을 방지할 수 있다. 'Augmentation'의 방식으로는 90° rotation, random crop, −15°~15° random rotation을 사용했다.

그림 2. Augmentation 예시(왼쪽부터 원본, random crop, 15° random rotation)

4) Model

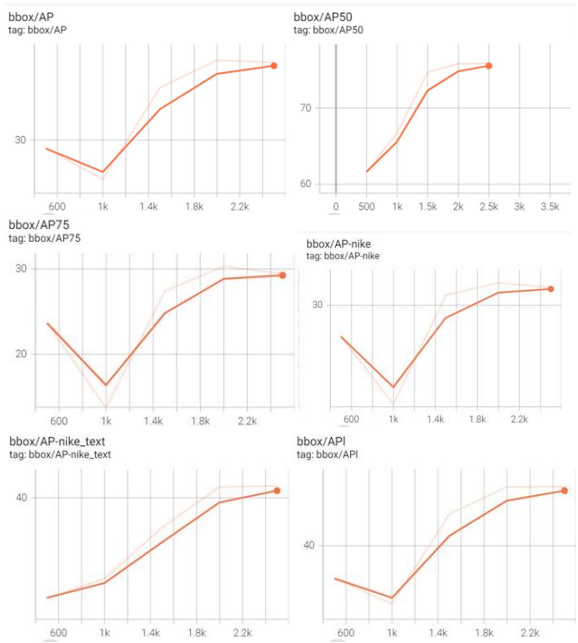
우리는 구현의 용이와 model 개발에 대한 집중을 위해 'Detectron2'[8]를 이용해 학습시켰다. Faster R-CNN[9] 구조를 사용해 backbone의 경우 ResNeXt101과 pretrained model의 weight를 가져온 뒤 전이 학습(transfer learning)시켜 본 연구의 data set에 적합하도록 weight를 바꾸었다.
다음 단계인 모자이크 처리 과정에서는 bounding box의 좌표를 가져온 뒤 interpolation을 수행하고 Gaussian Blur 처리를 하는 방식을 진행했다.

5) 실험 및 결과

Training은 2500 iteration 동안 수행하였고, AP를 통한 성능 검증은 아래의 [그림 3]과 같다. AP는 전반적인 average percent를 말해주는 것이며 AP50, AP75의 경우 IOU threshold가 각각 50, 75일 때이다. Aps, APm, APl의 경우 small, medium, large object에 대한 결과치이며 표의 2번째 줄의 경우 각각 category에 대한 AP를 뜻한다. Bounding box의 AP50에서의 점수가 76.632로 가장 높게 나왔으며 'nike'와 'nike_text'의 경우 AP 각각31.977 그리고 41.061의 점수를 얻을 수 있었다.

```
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 36.529 | 76.632 | 29.306 | 18.357 | 43.663 | 45.202 |
[08/31 11:50:49 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP   | category   | AP     | category   | AP     |
|:-----------|:-----|:-----------|:-------|:-----------|:-------|
| logo       | nan  | nike       | 31.977 | nike_text  | 41.081 |
OrderedDict([('bbox',
              {'AP': 36.52905411842812,
               'AP-logo': nan,
               'AP-nike': 31.977035582383586,
               'AP-nike_text': 41.08107265447266,
               'AP50': 76.63247980822896,
               'AP75': 29.306017123752447,
               'APl': 45.20201999620054,
               'APm': 43.66321129929595,
               'APs': 18.35748048224302})])
```

그림 3. 성능 검증 결과값



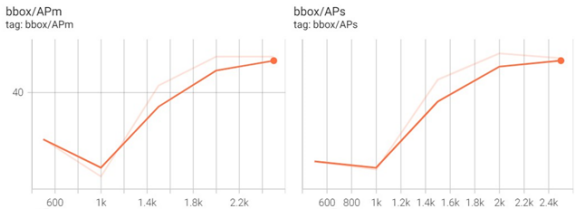


그림 4. 성능 검증 결과 그래프

그림 5. 이미지에 대한 bounding box와 blur처리 결과물

그림 6. 영상에 대한 bounding box와 blur처리 결과물

3. 결 론

적은 학습시간과 데이터에도 불구하고 이미지와 영상에 모자이크 처리가 잘 되는 것을 확인할 수 있었다. 온라인 객체 추적 알고리즘인 SORT를 사용하여 이전 프레임에서 검출된 객체와 현재 프레임에서 검출된 객체를 매칭시켜 객체추적을 한다면 더욱 좋은 결과를 얻을 수 있을 것이다.

참고 문헌

[1] 박성현, "실시간 특정 객체 모자이크 처리 시스템", 추계학술발표대회 논문집, 2019.11
[2] Woo-Seok Sim, "Portrait Right Protection System Using Real-Time Face Mosaic Technique Based on Deep Learning", JKITS, 2021.06
[3] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi "You Only Look Once: Unified, Real," Computer Vision and Pattern Recognition 2016, pp. 779-788, June 2016, arXiv:1506.02640.
[4] Tsung-Yi Lin, Microsoft COCO: Common Objects in Context, Computer Vision and Pattern Recognition (cs.CV), Submitted on 1 May 2014 (v1) last revised 21 Feb 2015 (this version, v3), arXiv:1405.0312 [cs.CV]
[5] Mark Everingham, Luc Van Gool, The PASCAL Visual Object Classes (VOC) Challenge, Int J Comput Vis (2010), Received: 30 July 2008 / Accepted: 16 July 2009 / Published online: 9 September 2009.
[6] Perez, Luis, and Jason Wang. "The effectiveness of data augmentation in image classification using deep learning." arXiv preprint arXiv:1712.04621 (2017).
[7] Shorten, Connor, and Taghi M. Khoshgoftaar. "A survey on image data augmentation for deep
[8] Wu, Y., Kirillov, A., Massa, F., Lo, W. Y. and Girshick, R. (2019), Detectron2, Github, https://github.com/facebookresearch/detectron2
[9] S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 39, No. 6, pp. 1137-1149, Jun, 2017.