Photo Search by Sketch

요 약

다수의 사진 속에서 한 장의 사진을 찾는 프로그램을 개발한다. 최근 늘어나는 저장 공간 장치의 용량으로 더 많은 사진을 찍고, 저장할 수 있다. 그중 한 사진을 빠른 시간 안에 꺼내 쓰거나 찾고 싶을 때, 유저의 스케치를 이용하여 유사도 분석을 통해 유사도가 가장 높은 사진을 출력하도록 Detectron2 의 Instance segmentation 과 custom trained model 을 이용하여 프로그램을 구현한다.

1. 서론

1.1. 연구배경

핸드폰, 외장하드, 노트북 SD 카드 등 저장매체의 발달이 빠르게 진행 중이다. 아이폰은 현재 512 기가바이트까지 출시되었고, 삼성의 외장하드는 4 테라바이트까지 출시되었다. 유저들이 접할 수 있는 저장공간이 커짐에 따라 유저가 저장할 수 있는 사진의 개수도 늘어나게 되었다.

현재 고화질 사진 한 장에 12 메가바이트 정도이고, 보통 4~10 메가 정도이다. 가장 화소가 큰 사진을 기준으로 계산해도 유저는 이제 약 4 만장을 저장하고 가지고 다닐 수 있게 되었다.

그중 유저가 중요 문서를 찍은 사진이나 의미 있는 사진과 같이 한 장의 사진을 찾고 싶어할 때, 기존에는 4 만장의 사진을 일일이 확인하며, 사진첩을 뒤져야 했다. 이는 매번 상당히 많은 시간이 소요되곤 한다.

메모리의 개발로 점점 더 저장공간의 크기는 커질 것이고, 더 많은 사진을 저장할 것이고, 더 많은 사진 중에 한 장의 사진을 찾기 위해 수만가지 사진을 일일이 비교하여 찾는 것은 비효율적이다.

기존 안드로이드의 사진 검색 기능은 "사람, 문서, 노트, 셀피, 풍경" 정도의 texture 를 입력하여 사진을 검색한다. 사용해본 결과 한번의 검색으로 많은 수의 사진이 출력되고, 그 카테고리도 너무 광범위하다고 느껴졌다.

구글의 이미지 검색도 또한, 검색창에 입력한 texture 의 정보를 and 와 or 조건으로 분석하여 이미지를 검색하는 정도이고, 이 역시 너무 많은 사진을 출력하고, 원하는 사진도 잘 나오지 않았다.

따라서, 유저가 생각하는 사진을 더 정확하게 찾기 위해 찾고자 하는 사진의 특징을 스케치를 통해 그림을 그리면, 스케치와 사진의 특징 분석을 통해 유저에게 유저가 원하는 사진과 가장 비슷한 사진을 출력해주는 프로그램을 개발한다.

1.2. 연구목표

구글과 안드로이드에서도 사용할 수 있는 프로그램을 만드는 것이 목표이며, 유저가 찾고자 한 사진이 프로그램이 제시한 사진 5 개 중 포함되게끔 정확도를 보유한다.

유저가 원하는 사진과 사진첩의 사진 간의 유사도 분석을 위해 새로운 사진이 저장될 때마다 Detectron2 의 instance segmentation 을 이용하여 feature 간 거리 정보, feature 정보 등을 계산하여 저장한다

스케치를 입력하였을 때, 스케치의 정보를 위에서 정한 특징으로 분석하고, 추출한 label 을 이용하여 사진첩 내의 사진이 갖고 있는 특징과 가장 비슷한 사진 5 장을 사용자에게 제시한다.

2. 관련연구

2.1. OPEN CV

실시간 컴퓨터 비전을 목적으로 한 프로그래밍 라이브러리이다. 컴퓨터 비전 응용의 예는 제품 결함 검사(industrial Inspection), 문자인식(Character Recognition), 얼굴 인식(Face Recognition), 지문인식(Fingering recognition), 움직이는 물체 검출(Motion Detection) 및 물체추적(Object Tracking) 등이 있다. 이 중, 물체 분할화(Object segmentation)를 위해 이를 사용한다.

2.2. Yolo v1~v6

V1: 2016 년 5 월에 게시된 논문으로 Anchor Box 를 사용하지 않고 Cell 단위로 Bounding Box Regressor 과정을 통해 Box 를 찾는다. Localization Error 가 발생할 확률이 높고 성능이 낮은 편에 속한다. 특히 작은 물체에 관해 찾을 때, 성능이 나쁘다는 단점이 있지만, Detection 속도는 아주 빠르다는 장점이 있다.

V2: 2017 년 12 월에 게시된 논문으로 V1 과 달리, Darknet19 모델을 사용하여 속도를 빠르게 하였다. 또한, Batch Normalization, High Resolution Classifier, Anchor Boxes, Dimension Cluster, Direct Location Prediction, Fine-Grained Features, Multi-Scale Training 을 이용하여 성능이 좋아지고, 다양한 size 의 input 을 넣을 수 있었지만, 여전히 작은 물체에 대해서는 성능이 낮았다.

V3: 2018 년 4 월에 게시되었으며, sigmoid 를 택했고, Darknet-53, Predictions Across Scales 을 이용해 속도는 다소 줄었지만, 작은 물체에 대해서도 detection 할 수 있게 되었고, 성능이 향상되었다.

V4: CSPDarknet53, SPP, PAN, BoF, BoS 를 사용하여 모델을 더 Robust 하게 만들었고, 다양한 Augmentation 을 적용하여 성능이 향상되었고, 속도는 빨라졌다.

V5: Pytorch 로 구현된 BackBone 을 사용하였다.

V6: 3 개의 Scale 로 detection 하던 것을 4 개로 늘려 더 다양한 size 의 object detection 이 가능해졌다.

2.3. Detectron 2

FAIR (Facebook Artificial Intelligence Research)에서 만든 pytorch 기반 object detection 과 semantic segmentation 을 위한 training/inference 플랫폼이다. Mask R-CNN 모델을 사용하여 2019 년 10 월 발표하였고, Box, Mask, Key-point, Dense-pose, Semantic segmentation 을 제공한다.

Detectron2 은 instance segmentation 으로 사람과 여러 물체를 yolo 보다 정확하게 검측할 수 있고 Key-point-RCNN 으로 사람의 자세 검측이 가능하다.

Detectron2 은 "Model Zoo and Baselines"에서 다양한 훈련된 모델을 사용할 수 있고 검측 속도가 늦은 반면 검측 정확율을 높일 수 있다.

facebookresearch/detectron2: Detectron2 is a platform for object detection, segmentation and other visual recognition tasks. (github.com)

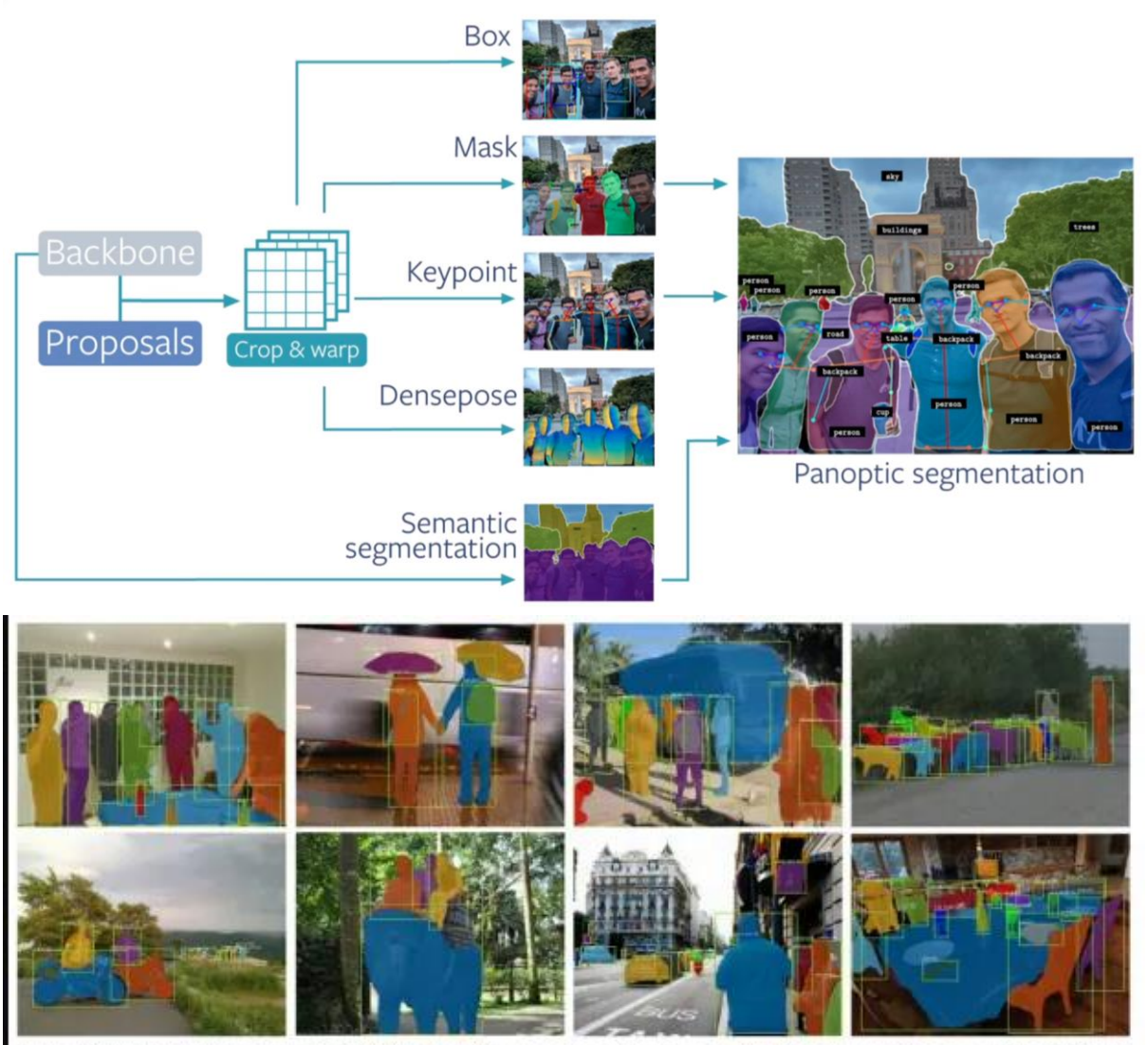


Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

2.4. Tensorflow

구글(Google)에서 만든 딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공해주는 라이브러리이다.

Tensorflow 연산은 상태를 가지는 데이터 흐름(stateful dataflow) 유향 그래프로 표현된다. Tensorflow는 파이썬 API를 제공하며 문서화가 다소 부족하지만 C/C++ API 도 제공한다.

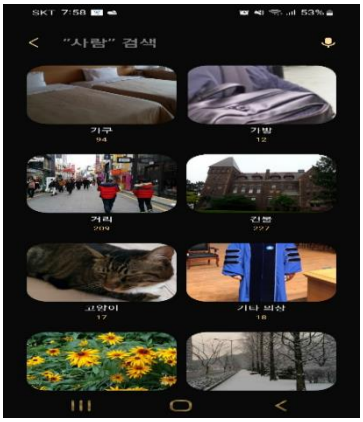
2.5. Pytorch

Facebook 의 인공지능 연구팀이 개발했으며 Python 을 위한 오픈소스 머신 러닝 라이브러리로 Torch 를 기반으로 하며, 자연어 처리와 같은 애플리케이션을 위해 사용된다. GPU 사용이 가능하기

때문에 속도가 빠르다. 아직까지는 Tensorflow 의 사용자가 많지만, 비직관적인 구조와 난이도 때문에, Pytorch 의 사용자가 늘어나고 있는 추세이다. Pytorch 는 강력한 GPU 가속화를 통한 Tensor 계산 (ex, NumPy)과 테이프 기반 자동 삭제 시스템을 기반으로 구축된 심층 신경망을 Python 패키지 형태로 제공한다.

2.6. 기존 사진 검색 관련 프로그램 조사

2.6.1. 안드로이드 사진 검색 프로그램

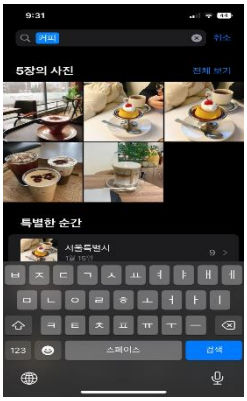


현재는 오른쪽 그림과 같이 검색창에 원하는 단어를 입력하여 찾거나, 카테고리로 분류되어 있는 사진 중에 원하는 사진을 일일이 찾아야하는 구조이다.

이미지 내의 "가구, 가방 등" 물체에 대한 인식은 거의 정확하지만, 사진을 검색할 때의 결과가 너무 많이 나온다.

또한, 카테고리가 한정적이고, "사람 여러 명", "서있는 사람" 과 같은 검색어는 검색이 되지 않는다.

2.6.2. ios 사진 검색 프로그램



검색어에 사용자 이름, 날짜, 위치 등을 입력하여 사진을 찾을 수 있다. 안드로이드와 마찬가지로 주소 정보에 대한 사용자의 동의가 필요하고, 누군가에게 전송한 사진을 저장하거나 컴퓨터로 사진을 옮겼을 때에는 위치 정보가 사라지기 때문에 검색이 불가능하다.

안드로이드와 달리, 사람을 식별하여 유저가 원하는 사진을 제공한다. 예를 들어, 사람 A 를 검색하고 싶으면, 분류된 사진 중에 항목 이름을 A 라고 등록하면, 같은 사람을 찾아준다.

iPhone 에서 사진 검색하기 - Apple 지원 (KR)

2.7. 기존 기술의 문제점

2.7.1. 연구의 문제점

두 프로그램 모두 검색어가 포함된 여러 사진을 찾아서 유저에게 제공한다. 또한, 사람의 자세와 사진 속에 몇 명의 인원이 있는지 등 세부 사항에 대해서는 사진을 찾기 힘들다. 두 프로그램 모두 위치 정보에 대해 같이 기록하는 것을 알 수 있는데, 이는 다른 사람이 전송한 사진을 받거나 사진을

컴퓨터로 옮겼을 때에는 사라지는 정보들이다. 따라서, **위치 기반 정보(GPS)에 의존한 이미지 저장이 아닌**, 사진의 특징을 이용하여 유저가 찾고자 하는 사진의 장소를 맞출 수 있는 프로그래밍이 필요하다.

2.8.2. 해결 방안

2.8.2.1. sketch

유저가 찾고자 하는 사진을 스케치로 그리면, 스케치의 각 요소들을 파악하여 특징을 태그로 구성하여 준다. 각 특징을 저장하여 사진첩의 사진들과 가장 유사한 사진을 제공한다. 스케치 안에 있는 그림들은 사용자가 찍었을 당시 뇌리에 박혔던 내용이기 때문에 대부분 어떤 사진의 인상 깊은 특징일 경우가 많다. 따라서, 사용자가 그린 스케치를 가중치를 높게 두어 분석한다.

예를 들어 사용자가 동상과 사람 1 명을 그렸다고 하면, 동상은 사람 1 명 보다 중요한 단서가 될 것이다. 한 가지 예를 더 들자면, 커다란 문과 사람 1 명을 그렸을 때도 역시 커다란 문이 핵심 키워드가 될 것임을 알 수 있다. 따라서, 문과 동상과 같이 특별한 것에 가중치를 더 주어 정확도를 높인다.

2.8.2.2. Yolo v6

스케치와 사진첩의 사진을 Yolo v6 를 이용하여 segmentation 하여 각각의 태그를 pytorch 로 저장한다. 각각의 태그의 비교를 통하여 사진과 스케치의 유사도를 나타내고, 가장 비슷한 것을 출력한다.

2.8.2.3. Detectron2

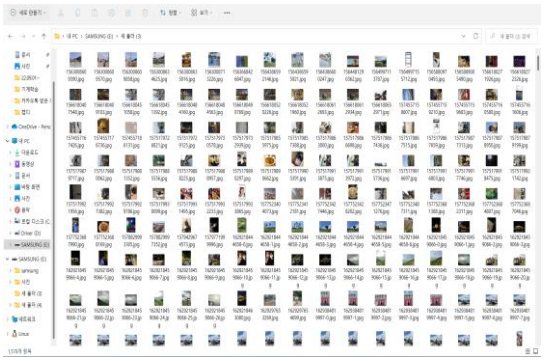
Detectron2 는 faster-RCNN, mask-RCNN, key-point-RCNN, panoptic-FPN 등 검측 방법으로 이미지의 물체를 정확하게 구분되어 검측할 수 있다. 또한 Detectron2 Model Zoo 에서 많은 pre-train 모델을 제공하여 즉시 물체인식이 가능하다. (랜드마크는 직접 데이터셋 훈련해야 된다.)

반면, 물체인식은 잘 되지만 배경인식은 하늘, 나무, 가로등 같은 간단한 배경은 되지만 다른 복잡한 배경은 아직 안되어 훈련할 필요 있다.

3. 프로젝트 내용

3.1. 시나리오

3.1.1. 이미지 feature 저장

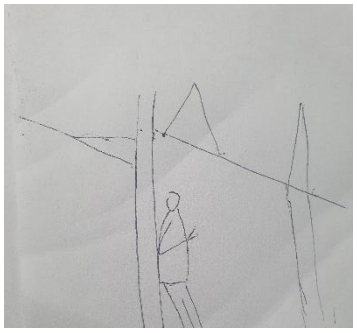


	특징1	특징2	특징3	특징4	특징5
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0
10	5.4	3.7	1.5	0.2	0
11	4.8	3.4	1.6	0.2	0
12	4.8	3.0	1.4	0.1	0
13	4.3	3.0	1.1	0.1	0
14	5.8	4.0	1.2	0.2	0
15	5.7	4.4	1.5	0.4	0
16	5.4	3.9	1.3	0.4	0
17	5.1	3.5	1.4	0.3	0
18	5.7	3.8	1.7	0.3	0
19	5.1	3.8	1.5	0.3	0
20	5.4	3.4	1.7	0.2	0
21	5.1	3.7	1.5	0.4	0

왼쪽과 같이 수많은 사진의 feature 를 사진이 디바이스에 처음 저장되었을 때, 사진 속의 인원, 대략적인 상대적 위치, 사람의 자세, 계절감, 낮과 밤 등 다양한 정보를 Detectron2 를 이용하여 분석하고, pytorch 를 이용하여 오른쪽 같이 저장한다.

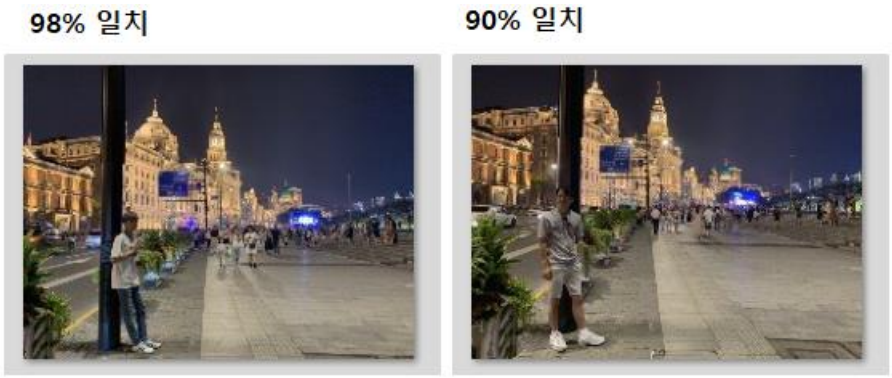
3.1.2. Sketch

사용자가 오른쪽과 같은 그림을 그렸다면, 프로그램은 "사람의 상대적 위치, 사람 1 명, 기둥, 건물" 등의 특징들을 추출하고, 그중 이 사진만의 뚜렷한 특징인 "기둥"을 가중치를 높게 두어 특징을 저장한다.



3.1.3. 유사도 분석 및 사진 출력

Sketch 의 특징과 3.1.1.의 특징을 분석하여 같은 특징을 갖고 있는 사진들을 아래 그림과 같이 유사도를 분석하여 유사도가 높은 상위 10 장의 사진을 출력한다.



3.2. 요구사항

3.2.1. 이미지 feature 에 대한 요구사항

-Detectron2 를 이용하여 추출된 모든 detection 결과를 pytorch 로 저장한다. 사람과 같이 한 사진에 여러 객체가 등장할 수 있는 경우에는 그 수 값을 모두 더해서 전체 인원이나 자동차와 같은 정보를 저장한다.

-사진의 특이점 예를 들어, 사람의 사진상의 위치, 동상, 건물 등 의미 있는 정보에 대해 가중치를 두어 저장한다.

3.2.2. Sketch 에 대한 요구사항

-유저가 그린 그림으로부터, 사람의 자세, 인원 수, 배경, 주변 물체 등의 정보를 추출한다.

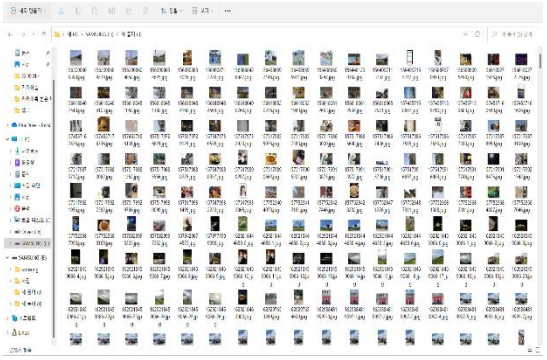
-한 명의 사람이라는 정보보다 근처 조형물이나 여러 명, 사람의 자세 등 그 이미지만의 특징을 가중치를 두어 저장한다.

3.2.3. 유사도 분석 및 사진 출력

-상위 10 개의 사진을 출력하고, 각 항목에 따라 유사도를 분석한다.

3.3. 프로그램 설계

3.3.1. Gallery 부분



왼쪽과 같이 수많은 사진이 디바이스에 처음 저장되었을 때, 갤러리 내의 모든 사진의 사진 크기에 해당하는 image width, image height, 사진 중심부 좌표에 해당하는 image mid, 사진에 포함된 객체들에 해당하는 리스트인 class, 그 클래스의 개수와 이름을 저장한 label_class, label_name, label_num 값을 그림(가)의 label.xlsx 에 저장한다.

또한, 그림(나)의 pic1.xlsx 에서처럼 각 사진들의 instance segmentation 세부 결과인 각 객체의 위치, 정확도, 전체사진에서 차지하는 비율인 size, 각 객체의 위치를 대략적으로 알기 위해 내적 값들을 저장한다.

image name	test1.jpg	test2.jpg	test3.jpg
image width	4032	3456	4032
image height	3024	4608	3024
image mid	[2016,1512]	[1728,2304]	[2016,1512]
class	[0, 0, 9, 0, 2, 0, 0,	[0, 0, 0, 0, 0, 0, 0, 0,	[0, 2, 4, 28, 7, 28,
label_class	[0, 9, 2, 25, 26]	[0, 26, 58, 73]	[0, 2, 4, 28, 7]
label_name	['person', 'bicycle	['person', 'bicycle', '	['person', 'bicycle'
label_num	[18, 1, 2, 1, 1]	[16, 1, 2, 1]	[1, 1, 3, 2, 1]

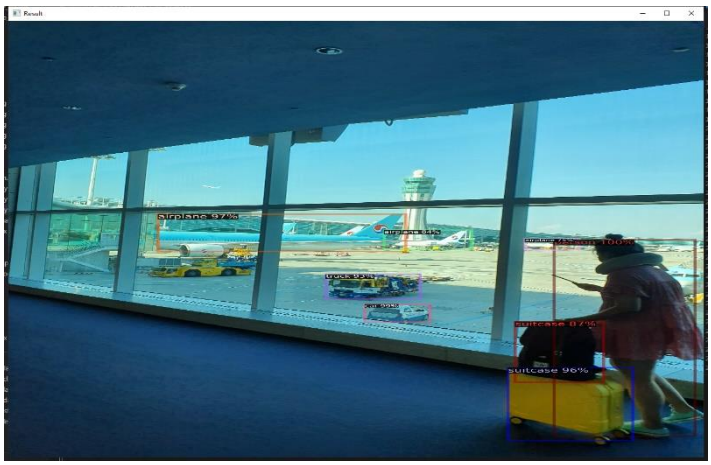
(가)label.xlsx

labelname	position	score	size	pos_mid	pos_sub	pos_dot
person	[444,3176],[908,4027]	0.9994	3.239%	[464,851]	[-1552.0,-661.0]	2222136
person	[1352,3221],[1575,4028]	0.9971	1.476%	[223,807]	[-1793.0,-705.0]	1669752
traffic light	[174,1831],[403,2237]	0.9948	0.763%	[229,406]	[-1787.0,-1106.0]	1075536
person	[342,3215],[427,3450]	0.9749	0.164%	[85,235]	[-1931.0,-1277.0]	526680
car	[2501,3205],[2790,3347]	0.9747	0.337%	[289,142]	[-1727.0,-1370.0]	797328
person	[1629,3247],[1729,3524]	0.9729	0.227%	[100,277]	[-1916.0,-1235.0]	620424
person	[247,3247],[310,3466]	0.9684	0.113%	[63,219]	[-1953.0,-1293.0]	458136
person	[2110,3214],[2253,3561]	0.963	0.407%	[143,347]	[-1873.0,-1165.0]	812952
person	[2254,3186],[2358,3527]	0.9567	0.291%	[104,341]	[-1912.0,-1171.0]	725256
person	[1849,3276],[2182,4029]	0.9533	2.057%	[333,753]	[-1683.0,-759.0]	1809864
person	[882,3215],[950,3429]	0.9309	0.119%	[68,214]	[-1948.0,-1298.0]	460656
person	[1220,3243],[1313,3518]	0.9264	0.21%	[93,275]	[-1923.0,-1237.0]	603288

(나)pic1.xlsx

3.3.1.1. Detectron2 instance segmentation

각 사진의 사이즈를 맞춰주기 위해 opencv2 라이브러리를 사용하였고, 사진의 object detection 을 위해 Detectron2 의 instance segmentation 을 사용하였다. 결과값을 엑셀에 출력하기 위해 xlsxwriter 패키지를 사용했고, 이미지 별로 detection 한 결과를 이미지.xlsx 에 저장하였다. Image segmentation 의 결과값인 labelname, 정확도 등을



포함하였고, 각 객체의 사진 내의 위치 정보를 저장하였다. 또한, 그 위치 정보를 이용해 사진 내의 feature 의 차지하는 비율도 저장하였다.

3.3.2. Sketch

3.3.2.1. sketch 그림판

사용자는 왼쪽과 같이 사용자가 찾고자 하는 사진을 기억에 의존해 화면의 팝업창에 그림을 그린다. 물체 하나를 그릴 때마다 각 그림을 구분해 주기 위해 하단의 "다음 그림" 버튼을 눌러 펜의 색깔을 바꾸어 줘야 한다.

왼쪽의 경우에는 비행기, 캐리어, 사람 이렇게 셋을 그린 경우이고, 각각은 점의 좌표 값(point), 그림의 중앙 좌표 값(mid), 각 그림의 크기 값(max, min, size)을 3 차원 리스트 하나에 저장된다.

유저의 스케치를 좌표로 변경하는 데에는 파이썬의 tkinter 패키지와 canvas 를 사용하였다.

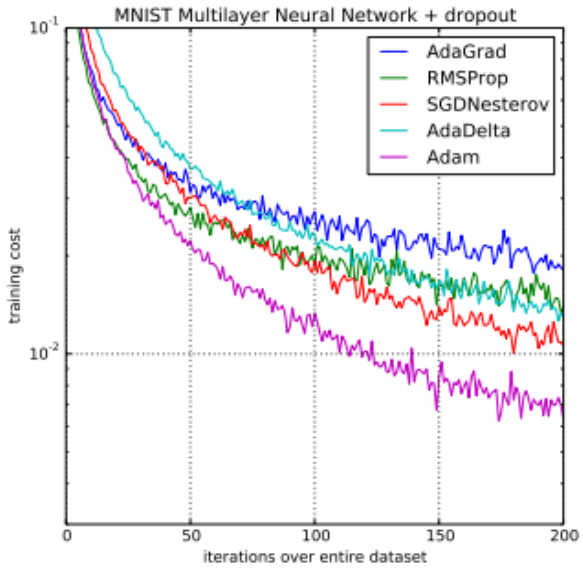
	point	mid	max	min	size	answer	rate
first picture	[0, (48.0, 163.0)	(133, 152)	(192, 193)	(48, 103)	10.8	airplane	89%
second picture	[0, (295.0, 105.(	(292, 151)	(317, 225)	(269, 104)	4.84	person	97%
third picture	[0, (225.0, 169.(	(245, 192)	(274, 231)	(216, 146)	4.11	suitcase	75%

3.3.2.2. sketch training model 부분

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2 [3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNet 구조

입력층에는 Linear 512 차원의 함수를 썼고, CNN 에는 ResNet 18 을 사용하였다. 또한, 옵티마이저로는 Adam 을 썼다. loss 값은 CrossEntropyLoss 를 사용하였고, 배치사이즈 64 로 53,125 번의 트레이닝하였고, 아래 사진 (가)와 같이 이를 20,000 번 반복하여 Train Loss: 1.0826, mAP: 0.294 를 얻었다.

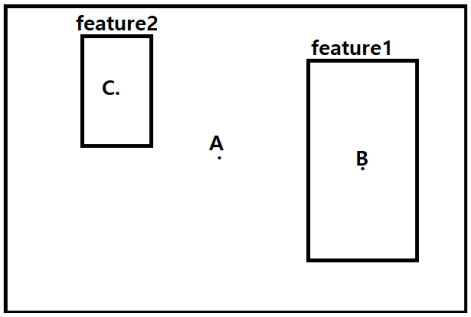


ResNet 18 을 사용한 이유는 GPU 의 용량 소모가 크지 않고, 성능이 좋으며, 파라미터 설정이 간단하다는 점에서 채택하였고, 아담 옵티마이저를 사용한 이유는 비슷한 이유로 구현이 간단하며, 메모리가 많이 요구되지 않기 때문에 아담 옵티마이저를 선택하였습니다.

(가) 트레이닝 결과

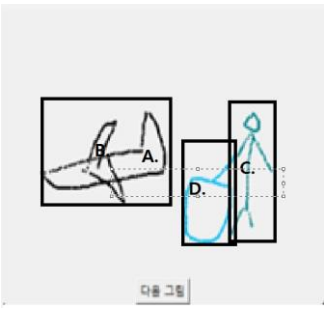
3.3.3. logic

Image1



왼쪽 그림과 같이 사진의 detectron2 결과값인 객체가 feature1, feature2 라고 가정했을 때, 이미지의 중간인 점 A 와 feature1, feature2 의 중점이 B, C 와의 내적을 구한다. 벡터 AB 와 벡터 AC 의 내적 값을 구해 이를 저장하고, 스케치도 같은 방법으로 내적 값을 구해 스케치의 내적 값과 사진의 내적 값을 빼서 값이 작은 순서로 이미지들을 배열한다. 이는 스케치의 물체 위치 벡터들과 이미지의 물체 위치 벡터 간의 배치 유사도를 따진 것을 의미한다. 가장 유사한 사진 상위 5 개를 score.xlsx 에 출력한다.

[스케치 벡터 내적 추출 과정]



벡터 AB, 벡터 AD, 벡터 AC 를 각각을 내적 값을 구해 6 개의 내적 값을 모두 더하여 합산한다.

4. 프로젝트 결과

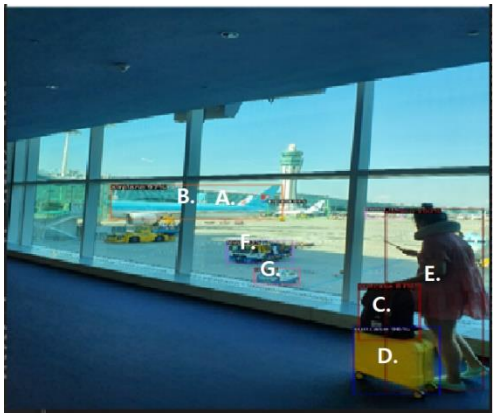
4.1. 연구 결과

유저가 원하는 사진을 갤러리 내의 수만장의 사진 중에 찾고 싶을 때, 스케치를 이용하여 그림의 특징을 기억에 의존하여 그리면, 구현한 프로그램을 통해 스케치와 이미지 간의 유사도를 분석하여 가장 유사한 사진 상위 5 개를 출력하는 것이다.

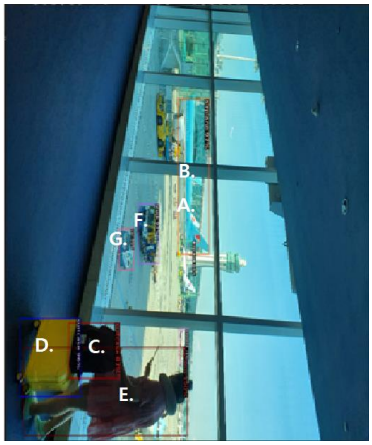
[출력화면]

4.2. 성능 평가

전반적으로 높은 신뢰도를 가졌으나, logic 을 이미지 중심과의 벡터 합으로 구성하여 여러 이미지 중 찾고자 하는 이미지가 90 도 180 도 회전되어 있는 이미지도 똑 같은 정확도를 띄며 아래와 같이 출력한다.



Score: 91%



Score:91%

5. 결론 및 기대효과

수많은 사진 중에서 유저가 원하는 사진을 sketch 를 통해 그리면, 그중 가장 비슷한 사진을 출력할 수 있다. 유저가 sketch 를 잘 못 그리더라도 가중치를 이용하여 특징을 분석하므로 높은 정확도로 이미지를 유추해 낼 수 있다.

나아가 중요 문서와 같이 글씨가 위주인 사진에 대해서도 분석하여 프로그램을 좀 더 광범위하게 보편적으로 사용할 수 있도록 개발할 것이다. 또한, 컴퓨터에 있는 사진에만 국한되지 않고, 네이버나 구글 등 검색 엔진에서도 웹크롤링을 통해 이미지를 분석하여 스케치와의 유사한 사진을 제공할 수 있게 개발할 예정이다.

5.1. 추후 연구 방향

스케치결과 값인 액셀 파일을 A.csv, 그 파일을 이용하여 trained 된 모델을 통과하여 스케치가 무엇인지 나타내는 파일을 B.csv, 전체 갤러리를 image segmentation 하여 모델 결과값을 저장하는 파일을 C.csv, 모델 결과값을 logic 을 통과하여 가장 유사도가 높은 사진 5 장을 저장한 D.csv 라고 하였을 때, A 파일을 만드는 부분, A 파일을 이용하여 B 파일을 만드는 부분, C 파일을 만들어 B 파일과 비교하여 D 파일을 만드는 부분 이렇게 4 가지 부분이 아직 한 프로그램으로 구현이 안되어 있고, 따로 분리되어 개발되어 있다. 유저가 스케치를 하면, 바로 유저가 찾는 사진과 유사도가 높은 5 장의 사진이 나오도록 구현할 예정이다.

현재는 gpu 를 로컬에 있는 gpu 를 사용하였기 때문에 속도가 느려 resNet18 과 adam 을 사용하였고, 배치사이즈도 작고, iteration 도 많이 돌릴 수 없어서 정확도가 낮은 편이다. Gpu 를 성능이 좋은 학교 서버의 공유 GPU 를 사용하여 좀더 트레이닝 정확도를 높이는 방향으로 연구를 진행할 예정이다.

앞서 언급한 바와 마찬가지로 현재는 유저의 스케치 중 물체에 관해서 밖에 인식이 안되지만, 유저가 특정 글이 적힌 문서를 찾고 싶을 때에도 photo_search_by_sketch 를 이용하여 그림판에 글을 적으면, 이또한 분석이 되게 하여 문서 찾기에도 도움이 되는 프로그램을 구현하고 싶다.

사진을 찾을 때에는 보통 노트북 등 로컬에서도 가능하지만, 핸드폰 사진첩에서 찾는 경우가 많을 것 같다고 개발 과정 중 생각했다. 현재는 노트북에서 구현되어 있지만, 모바일 버전도 개발하여 언제 어디서나 사진을 찾고 싶을 때, 어플리케이션을 통해 프로그램을 재구현할 계획이다.

마지막으로 출력한 5 장의 사진 중에 유저가 원하는 사진이 있는지 분석하는 알고리즘의 효과적인 개선을 위해 유저가 원하는 사진이 5 장의 사진 중에 있는 정답의 상황에서 가중치를 높게 두어 정답의 상황이 점점 더 많이 발생하고, 오답의 상황은 그에 반하여 감소하게끔 유저의 선택을 반영하는 가중치 계산을 추가하여 프로그램을 쓸수록 더 정확도가 높아지는 방향으로 연구를 진행할 계획이다.

6. 참고문헌

[1] Detectron2 (facebook.com)

[2] PyTorch

[3] TensorFlow

[4] iPhone에서 사진 검색하기 - Apple 지원 (KR)

[5] Deep Residual Learning for Image Recognition (CVPR 2016), Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (Microsoft Research)

[6] Quick, Draw! Doodle Recognition Challenge | Kaggle

[7] Adam: A Method for Stochastic Optimization, Diederik P. Kingma, Jimmy Ba