Image/Video BM 이해
-이미지 응용기술-

김 승 환

swkim4610@inha.ac.kr

감사합니다

4. 이미지 응용기술

4.1 Transfer Learning

4.2 Style Transfer

4.3 AutoEncoder

4.4 Image Colorization

4.5 Image Search

4.6 GAN

4.7 Image Segmentation

4.8 Object Detection

4.9 Object Tracking

4.1 Transfer Learning

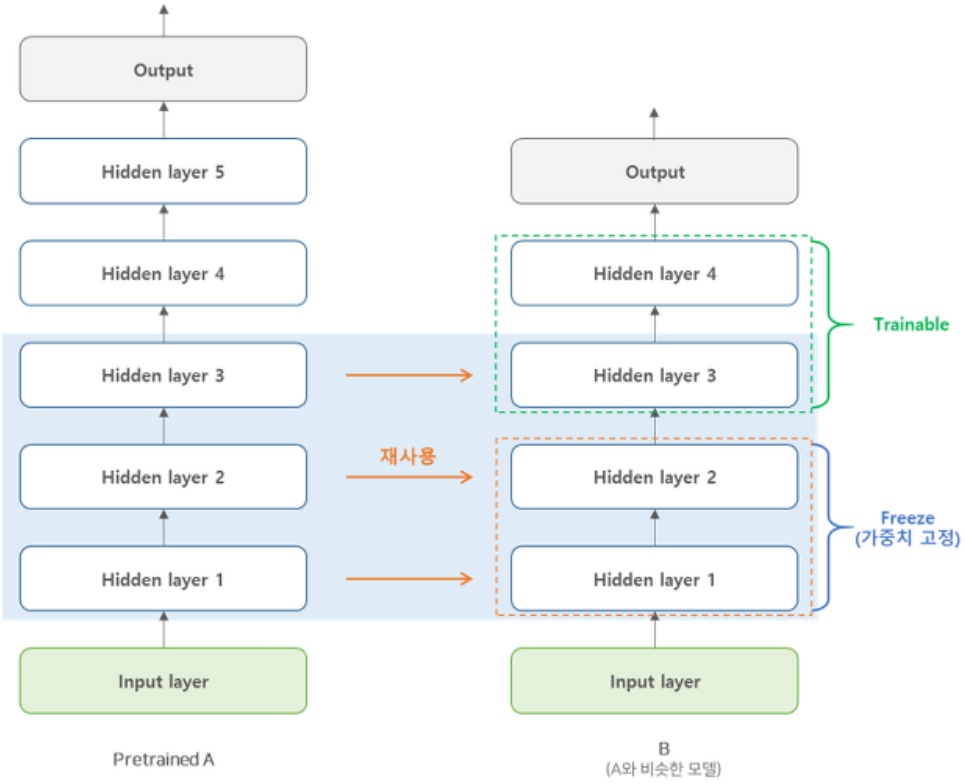
"피아노를 배운 사람은 다른 악기도 쉽게 배운다"

- 이미지 학습은 매우 힘든 일임
  복잡한 모델일수록 시간 혹은 컴퓨터 비용문제로 학습시키기 어렵다.
  어떤 모델은 2주정도 걸릴 수 있으며, 비싼 GPU 여러 대를 사용한다.
- 전이학습은 이미 잘 만들어진 학습모형(pre-trained Network)을 계승해서 해당 모델과 유사한 문제를 해결하고자 하는 방법임
- pre-trained Network로는 Xception,구글 Inception V3,ResNet50,VGG16, VGG19 등이 있음
- 아래는 VGG16 모형을 Keras에서 사용하는 방법을 보여주는 예임
  include_top은 FC Layer를 사용 안한다는 의미이고, conv_base.trainable은 학습과정에서 VGG16 모형의 파리미터를 갱신할 지 여부를 지정함. 전이학습에서는 반드시 False임

```
conv_base = VGG16(weights='imagenet', include_top=False, input_shape=(150, 150, 3))

model = models.Sequential()
model.add(conv base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
conv_base.trainable = False
```

4.2 Transfer Learning

- CNN 망은 Input, Convolution, FC(Fully Connected) Layer로 구분됨
- 전이학습은 아래 망에서 입력과 Convolution Layer를 재사용하고 FC Layer는 새로 구성하는 아이디어다. 이렇게 하면 Convolution Layer 학습에 필요한 리소스를 절약할 수 있음
- 아래 그림처럼 가중치의 일부만 고정하고 일부는 미세조정(fine-tuning)할 수 있음



4.2 Transfer Learning

- VGG16을 이용한 전이학습(개, 고양이 분류): 91% 분류함
  Cat_Dog_VGG16_2_Train.py, Cat_Dog_VGG16_2_Test.py

- RESNET을 이용한 전이학습(CIFAR10)
  인식률 92.16% 다. 좀 더 복잡한 모형은 95% 정도라고 알려져 있음
  Cifar10_RESNET_Train.py

4.2 Style Transfer

- CNN의 Conv. Layer Feature의 구조를 이용해 아래와 같이 일반 사진을 합쳐 새로운 스타일을 만들 수 있음
- Conv. Layer의 상위층은 개괄적인 윤곽을 나타내고 하위층은 국소적인 특징을 나타냄
- 아래 그림에서 튀빙겐에서 찍은 사진이 Content이고 고흐의 『별이 빛나는 밤에』가 Style임
- 우리의 목표는 아래의 Loss를 최소화하는 이미지를 만드는 것임

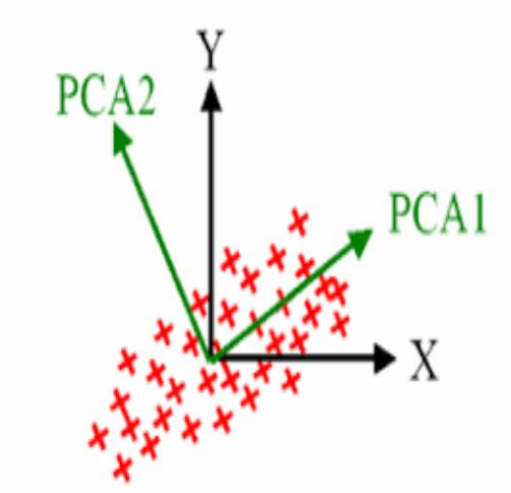
$\alpha \cdot L(\text{content}(\text{생성한 그림}) - \text{content}(\text{튀빙겐 사진})) + \beta \cdot L(\text{style}(\text{생성한 그림}) - \text{style}(\text{고흐 그림}))$

StyleTransfer.py

4.3 AutoEncoder

- 오토 엔코더는 비지도학습 방법이다. 즉, 레이블이 없음
- 오토 엔코더는 p개의 입력차원을 p 보다 작은 k개의 차원으로 압축(Encode)하고 이를 다시 p 차원으로 복원(decode) 한다. 이는 통계학의 PCA(Principle Component Analysis)와 유사함
- PCA는 선형 AE라고 볼 수 있음
- 통계학에서는 PCA를 다차원 변수의 차원축소 방법으로 사용함

PCA(Principal Component Analysis)



PCA is a method of reducing dimensions by using the correlation of multidimensional variables.
The goal of the PCA is to summarize the p-dimensional variables into k-dimensional variables (k < p) with minimal loss.

$$X \sim (\mu, \Sigma)$$

$$\Sigma = \Gamma \cdot D_\lambda \cdot \Gamma'$$

Γ is an p by p orthononal matrix composed of p eigenvectors
D_λ is an p by p diagonal matrix composed of p eigenvalues

$$PCA' = \Gamma \cdot X'$$

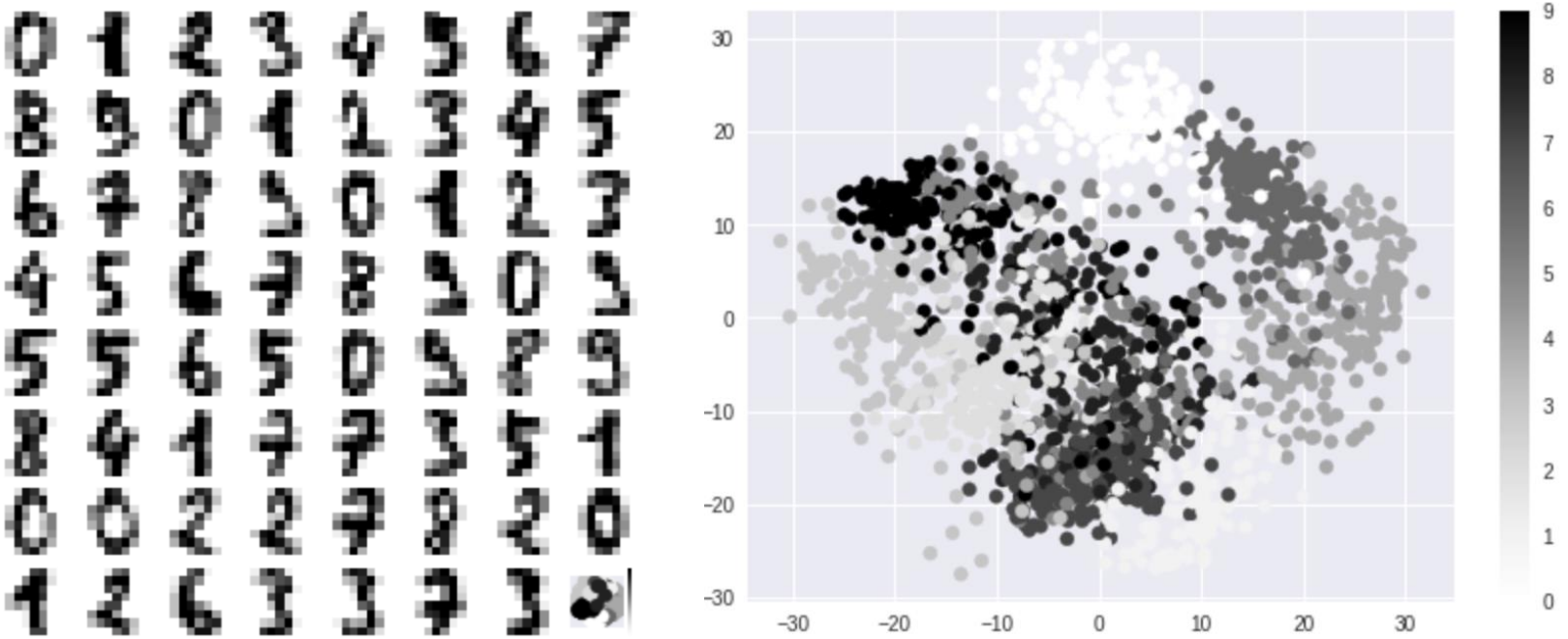
4.3 AutoEncoder

- 아래 코드는 8*8 숫자 이미지를 PCA를 통해 3개 차원으로 축소하는 코드임

```
# PCA.ipynb
from sklearn.datasets import load_digits
from matplotlib import pyplot as plt
digits = load_digits()
print(digits.data.shape)     # digits는 (1797, 64) 데이터로 8*8 숫자 이미지다.
fig = plt.figure(figsize=(6, 6)) # figure size in inches
for i in range(64):
    ax = fig.add_subplot(8, 8, i + 1, xticks=[], yticks=[])
    ax.imshow(digits.images[i], cmap=plt.cm.binary, interpolation='nearest')

from sklearn.decomposition import PCA
pca = PCA(n_components=3)
proj = pca.fit_transform(digits.data) # PCA Score
proj.shape                  # proj는 PCA를 통해 (1797,3) 데이터로 64개의 차원을 3차원으로 줄인 데이터다.

plt.scatter(proj[:, 0], proj[:, 1], c=digits.target)
plt.colorbar()
```

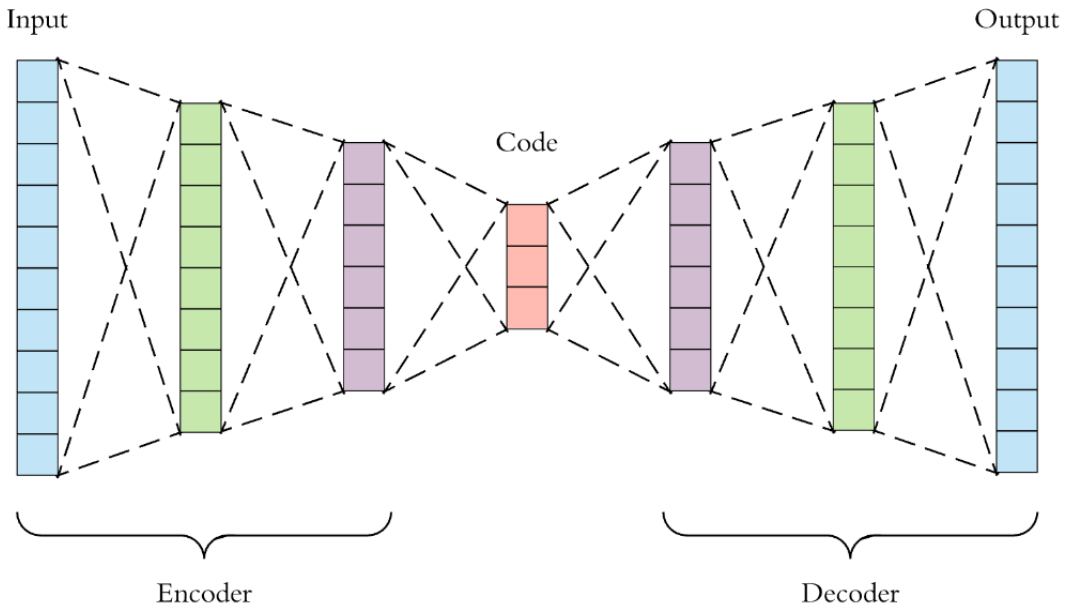
4.3 AutoEncoder

- 좌측 그림은 8*8 숫자 이미지 모양이고 우측은 이를 PCA로 3차원 압축했을 때, 3차원 공간에서 숫자들(점들)이 군집화 된 모양을 보여주는 결과임
- 군집화 되어 있다는 것은 같은 숫자들은 3차원 공간에서 모여 있음을 의미하는 것으로 64차원을 3차원으로 줄여도 어느 정도 숫자 분류가 가능함을 시사하는 결과임



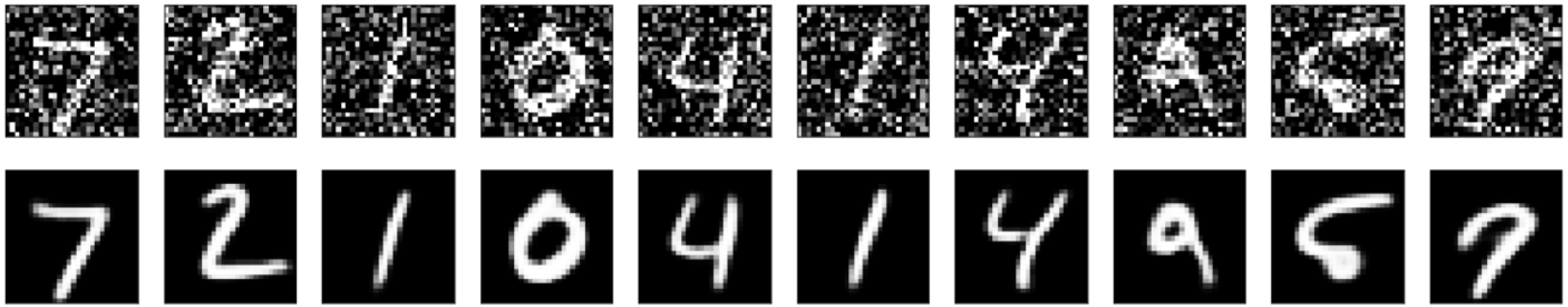
4.3 AutoEncoder

- AutoEncoder는 Input과 Output이 같은 딥러닝 모형임
- Target이 없으므로 Unsupervised Learning 형태임
- Input이 히든 레이어를 통과했다가 다시 재현되므로 암호화, 복호화가 일어나는 것임
- 일종의 통계학의 PCA(Principal Component Analysis)에 대한 비선형 버전으로 볼 수 있음
- 아래 그림에서 Code는 무엇일까? Code는 Input의 압축정보로 볼 수 있음
- AutoEncoder는 Noise Reduction, Anomaly Detection, ImageSearch 등의 용도로 사용됨



4.3 AutoEncoder

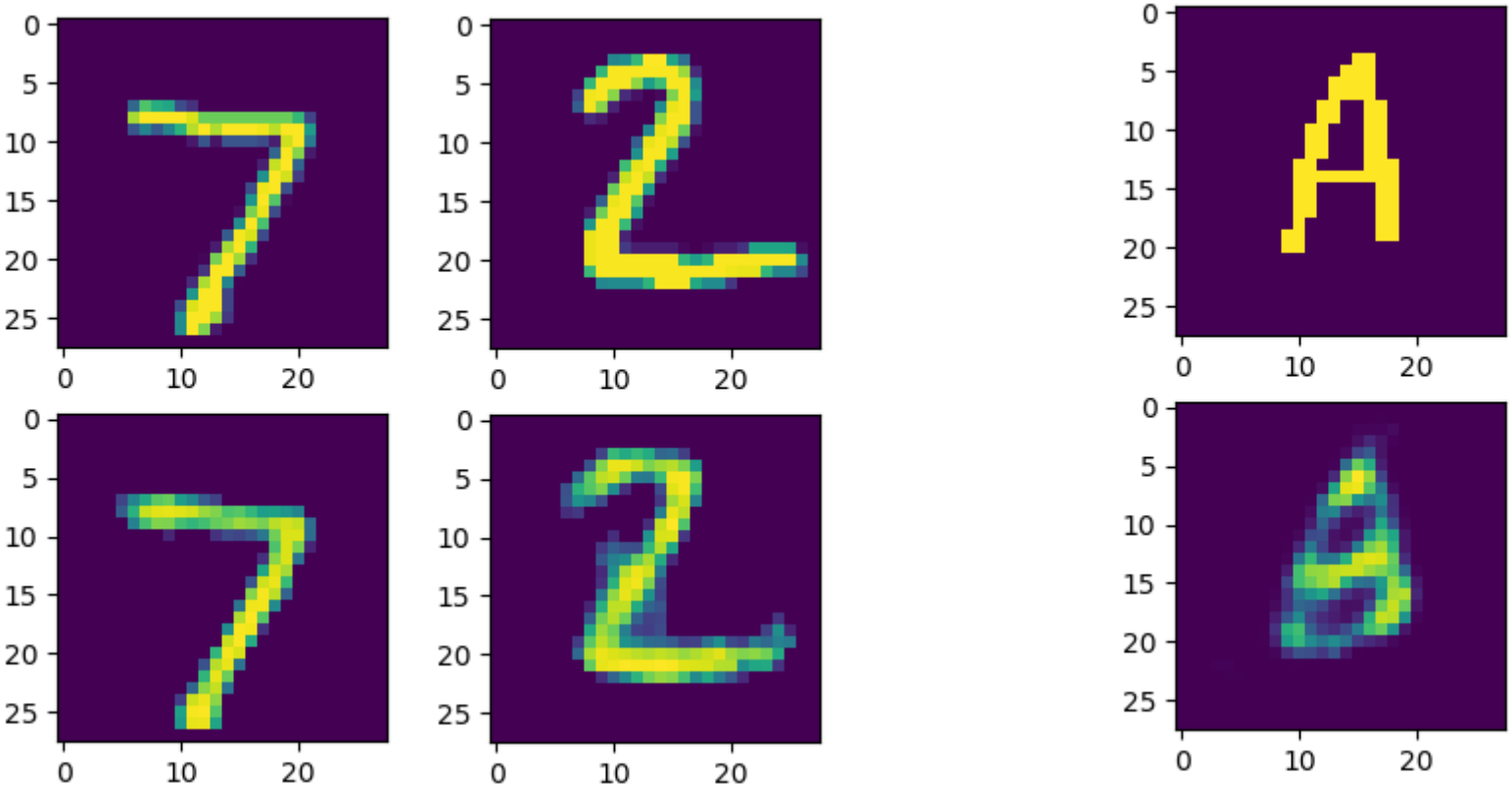
- Noise Reduction, Anomaly Detection 의 원리는 차원축소 과정에서 뚜렷하지 않은 특징은 사라지는 원리를 이용함
- 아래 그림은 상단이 노이즈가 추가된 원본 이미지이고, 하단이 노이즈가 제거된 이미지임
- 이 원리를 이용하면 노이즈가 섞인 이미지에서 노이즈를 제거할 수 있음
- 같은 원리로 비정상적인 신호가 포함된 신호를 오토엔코더에 넣어 학습시킨 후, 새로운 신호를 넣었을 때, 다른 신호에 비해 입력과 출력이 많이 다르면 비정상 신호로 판정하는 원리로 Anomaly Detection을 할 수 있음



AE_NoiseReduction.py

4.3 AutoEncoder

- 좌측 2개는 학습한 숫자를 AE로 Encode, Decode한 결과이고, 우측 A는 학습하지 않은 알파벳 A를 AE로 복원한 결과다. 학습한 숫자는 잘 복원하는데 학습하지 않은 알파벳은 “A”를 “8” 처럼 복원함을 알 수 있다. AE_Mnist_anomaly.py
- 이처럼 anomaly pattern은 복원이 안되는 특징을 이용해 anomaly detection에 활용한다.



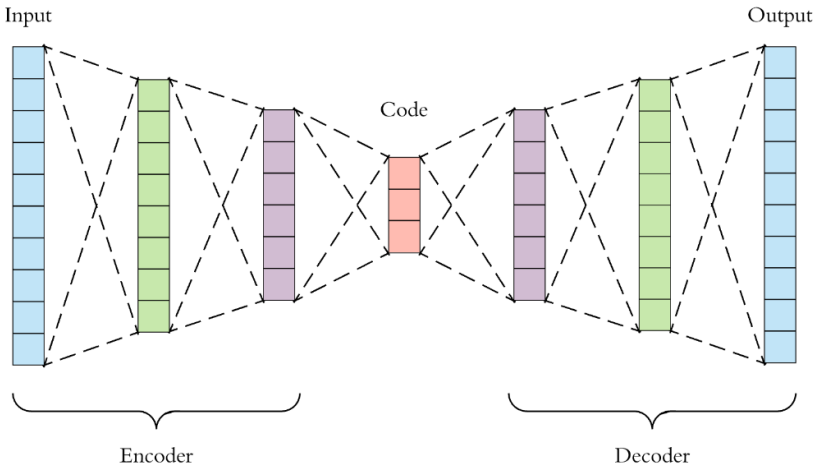
4.4 Image Colorization

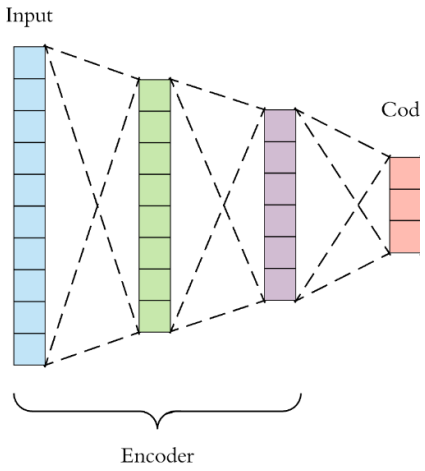
- 아래는 오토엔코더를 이용해 그레이 이미지를 입력해서 Color 이미지를 출력한 예임
- 이 모형은 다량의 컬러 이미지를 그레이화 하여 그레이 이미지는 입력으로, 컬러 이미지는 출력으로 학습시킨 후, 새로운 그레이 이미지를 주면 컬러로 만들어 주는 기술임
- 컬러 복원이 아니고 색칠하기임
- 이를 응용하면 흑백TV 시절의 이미지나 동영상을 컬러로 만들 수 있음

Colorization.py

4.5 Image Search

- AutoEncoder로 유사 이미지를 찾을 수 있음
- Encoded는 이미지에 대한 특징벡터 임
- 먼저, 좌측의 AutoEncoder 모형을 학습하여 Code까지의 가중값 확보
- 우측과 같이 모형의 Decode를 제거한 모형을 이용하여 아래의 방법을 수행함
- 이미지 DB의 이미지를 Encoded 벡터로 만들고 찾고자 하는 이미지의 Encoded 벡터를 만든 다음 kNN 기법으로 유사도를 측정하여 가장 가까운 이미지를 찾아주는 방법임

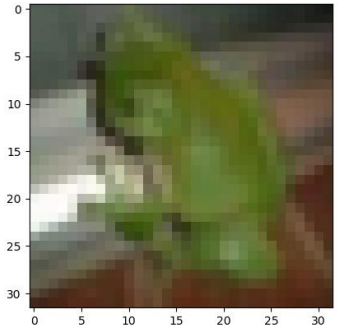




4.5 Image Search

- 아래는 CIFAR10 자료로 개구리 이미지와 유사 이미지를 찾은 결과임
- 개구리에 녹색이 많아 녹색 배경을 가진 사슴 사진 2장, 개구리 사진 3장이 검색됨

imagesearch_cifar10_ae.py

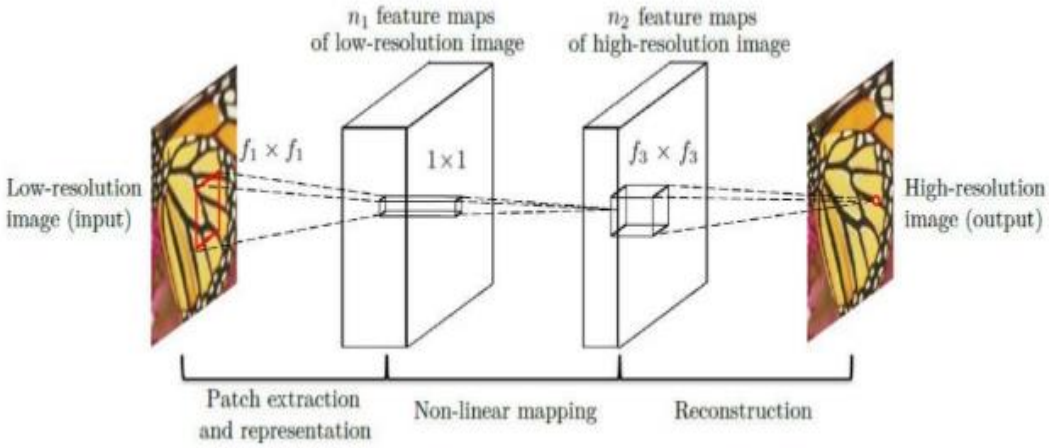


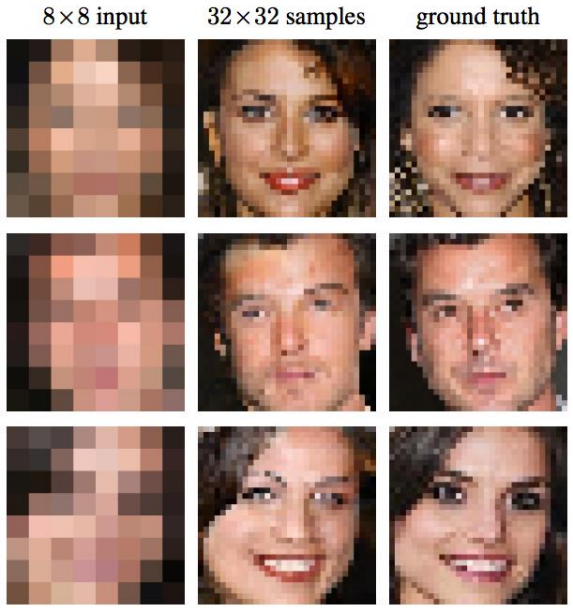




4.6 Super Resolution

- SRCNN은 저화질의 이미지를 고화질 이미지로 추정 복원하는 것임
- 학습기는 고화질 이미지를 저화질로 낮춘 이미지를 X, 고화질 이미지를 Y로 지정하고 Cost 함수는 MSE로 지정하여 학습함
- 학습기에 실제 저화질 이미지를 주면 고화질 이미지로 추정하는 원리임
- 위성 사진 확대, 저화질 CCTV 복원 등 다양한 활용분야가 존재함
- 최근에는 SRGAN 등의 알고리즘으로 발전하고 있음





4.7 GAN

- GAN(Generative Adversarial Network)은 실제 데이터와 비슷한 확률분포를 갖는 허구 데이터(fake Data)를 생성함
- 예를 들어, 필기체 글자를 학습하면 비슷하게 필기체 글자를 흉내내기도 하고, 사람 얼굴을 학습하면 가짜 사람 얼굴을 만드는 것도 가능함
- GAN에는 생성망과 판별망 두 개의 딥러닝 네트워크가 존재함
- 생성망은 임의의 저차원 난수로 부터 고차원 이미지를 생성하고 판별망은 생성된 이미지를 심사해서 진짜인지, 가짜인지 구별하는 네트워크임
- 생성망은 판별망을 속일 때까지 학습하고, 판별망은 생성망에 속지 않도록 학습함

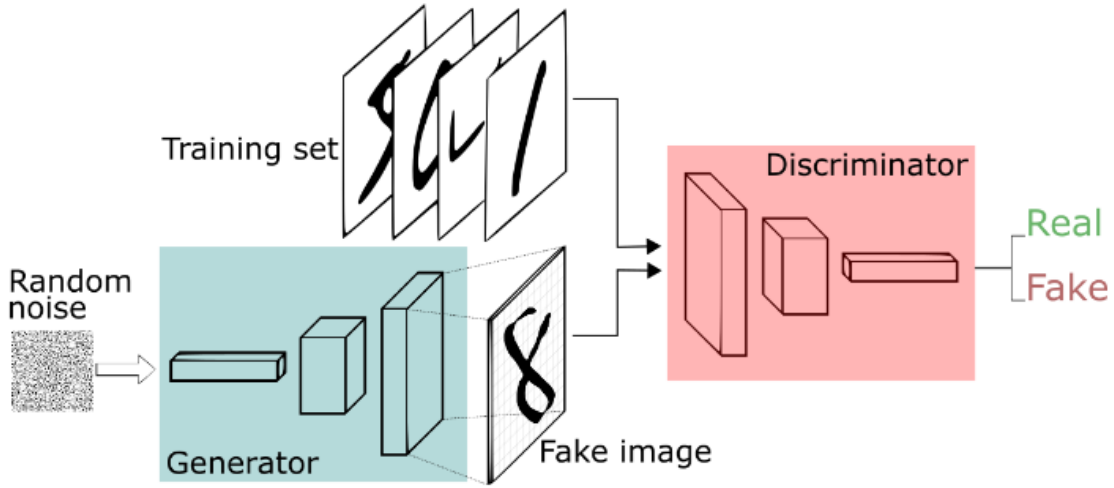
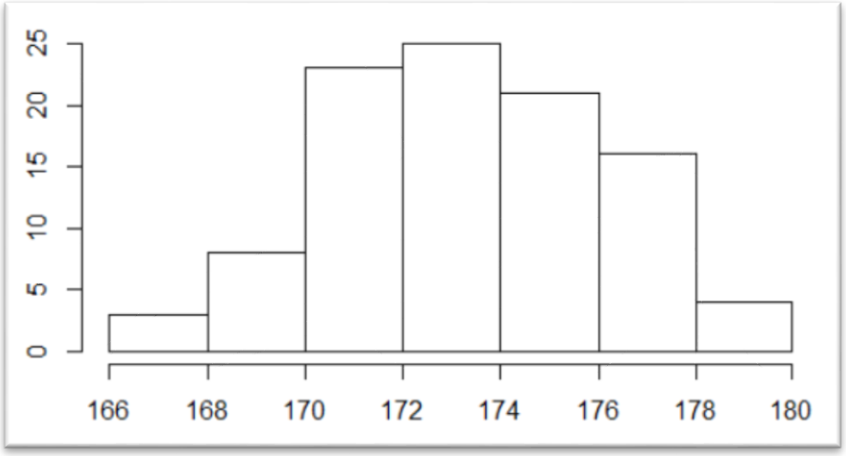


Image credit: Thalles Silva

4.7 GAN

- 좌측 그림은 어느 집단의 키를 히스토그램으로 표현한 것이고 우측은 사람들의 얼굴임
- 사실, 둘 다 가짜임
- 좌측 히스토그램은 평균이 173이고, 표준편차 3인 난수로 만들었고, 우측은 GAN으로 만들었다고 함
- 이론적을 좌측을 복잡하게 확장하면 우측이 될 수 있음(정말?)
- 좌측이 키 자료라고 하면 키와 몸무게를 동시에 만들 수 있을까?
  따로, 만들면 쉽지 않을 것임. 왜냐하면 키와 몸무게는 연관성이 존재하기 때문임
- 이 경우, Gibbs Sampling기법으로 다차원 난수를 만들 수 있음

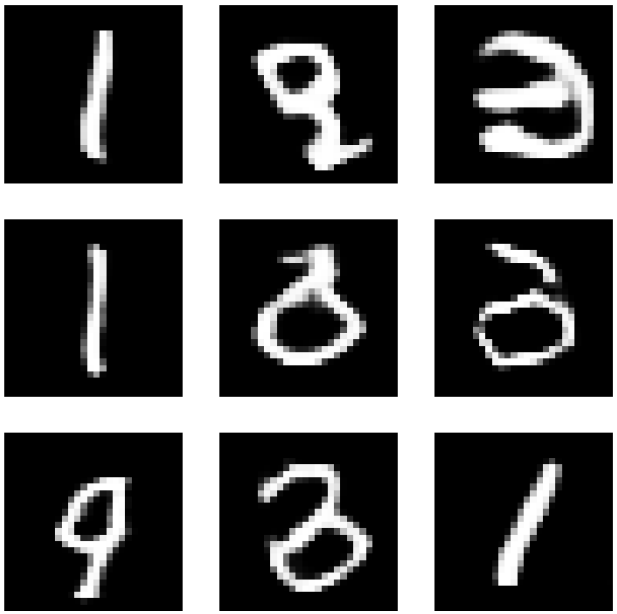


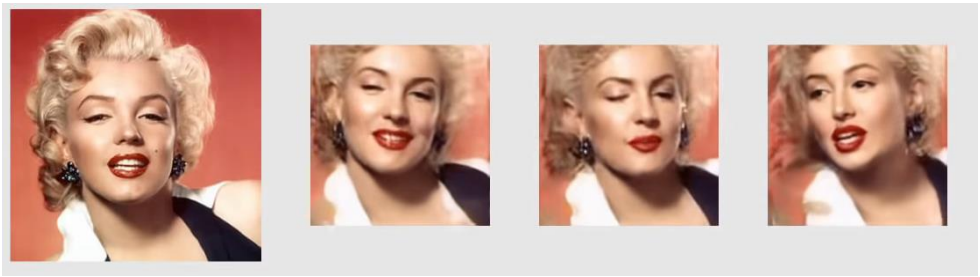
4.7 GAN

- 아래는 GAN 으로 만든 가짜 숫자임

dcgan_mnist.py

https://youtu.be/p1b5aiTrGzY

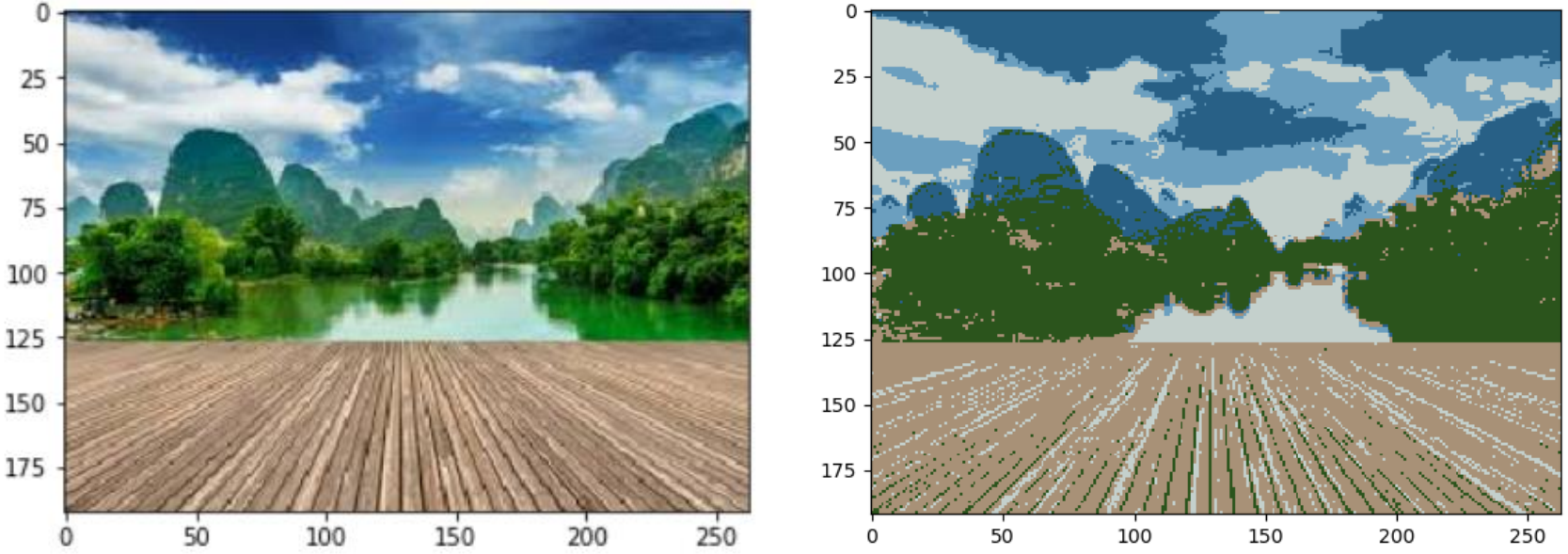




4.8 Image Segmentation

- 아래의 그림은 (192, 263, 3) 차원 이미지로 하늘, 구름, 산, 호수, 녹지, 마루로 이루어져 있음
- 픽셀 192*263 = 50,496개를 3차원에서 k means clustering 방법으로 세그멘테이션 함
- 아래는 5개의 세그먼트로 나누고 동일 세그먼트에 해당하는 픽셀은 세그먼트 중앙값으로 픽셀을 바꿔 표현한 그림임
- 같은 색깔이 같은 세그먼트 임

ImageSegmentation.py



4.9 Object Detection

- 하나의 이미지에 여러 개의 객체를 Detect 하는 문제임
- 한 장의 이미지에 한개의 객체만 있다고 하면 CNN 분류모형으로 해결할 수 있음
- 이미지 안에 여러 개의 이미지가 있다면 CNN 분류모형으로 해결되지 않을 것임

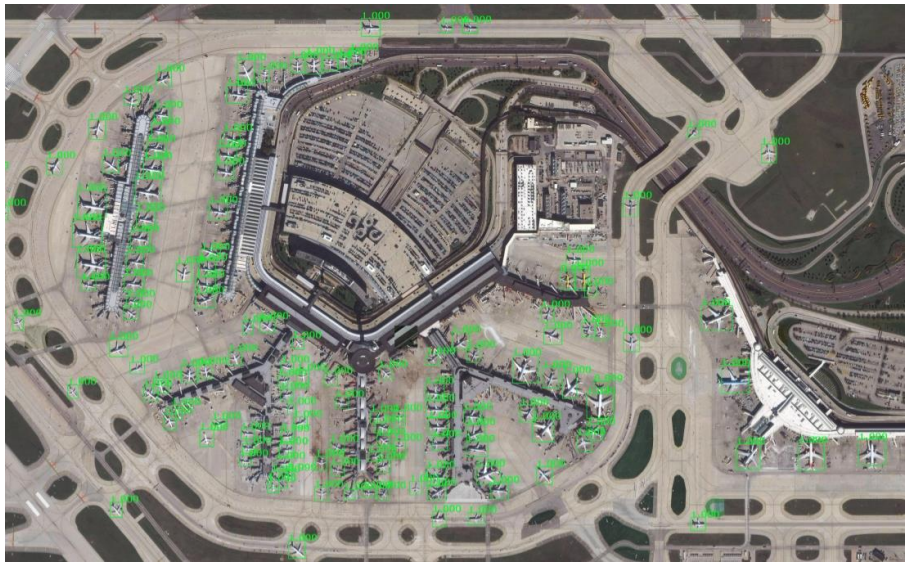
어떤 방법이 있을까?

- 이미지 상에서 객체들을 Crop(자르기)한 다음 CNN 분류모형을 작동하는 방법
  문제는 어디를 Crop할지 모르는 것임
- 방법 중 하나는 Sliding Window를 이용하는 것임
- 이미지를 몇 개의 조각을 만들고 분류기에 넣는다.
- 분류기에서 높은 확률로 Detect한 조각의 이미지 영역과 분류결과를 리턴함
- 이 경우, 조각의 크기를 결정하는 문제와 엄청난 계산량이 발생함
- 이 문제를 해결하기 위해 R-CNN 알고리즘이 개발됨
  R은 Region의 약자로 분류기에 넣기 전에 이미지의 일부만 선택하는 레이어를 사용함
- 이후, 속도와 성능을 개선한 Fast R-CNN, YOLO, SSD 등의 알고리즘이 제안되었음

4.9 Object Detection

록히드 마틴(Lockheed Martin)은 오픈 소스 딥러닝 라이브러리를 사용하는 위성 이미지 인식 시스템 'GATR(Global Automated Target Recognition)'을 개발했다고 지난 4일(현지시각) 밝혔다. 이 시스템은 전 세계의 넓은 지역에서 특정 대상을 신속하게 식별하고 분류함으로써 이미지 내에서 수작업으로 항목을 분류하고 라벨을 지정하는 시간을 절약할 수 있다.

보통 위성 이미지 내에서 항목을 수동으로 분류하고 레이블을 지정하는 작업은 시간이 많이 소요되는 과정이다. 그러나 이번 개발된 딥러닝 모델은 위성 이미지 분석을 가속화하고 자동화하여 상당한 시간을 절약하는 것이다.



4.9 Object Detection(R-CNN)

- 이미지의 색을 여러 개의 작은 군집으로 나누고 이를 합쳐가면서 의미 있는 이미지 군집화 (Selective Search)
- 이미지 군집 (X, Y, W, H) 정보로 Crop한 후, 분류

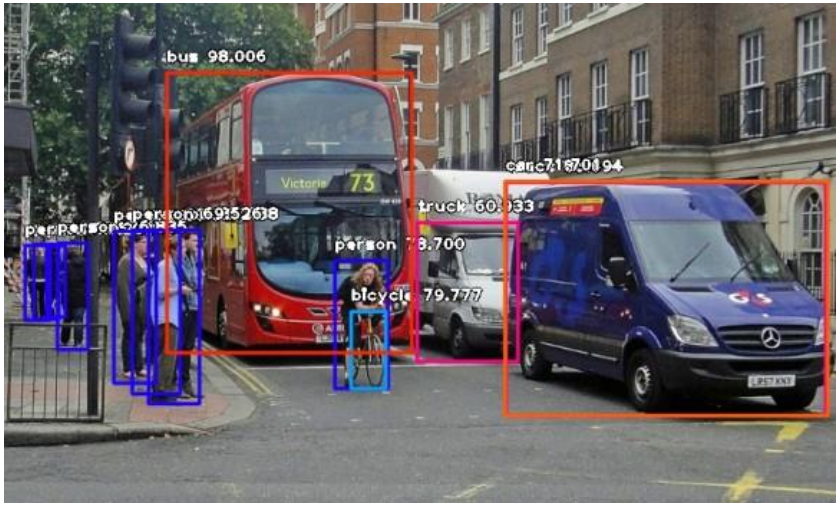
4.9 Object Detection

- 아래는 이미지의 색깔을 몇 개의 군집으로 군집화한 후, 특정 군집만 표시하는 예임
- 이미지 중에 원하는 객체의 위치 정보를 검출할 수 있음

imageClustering.py

4.9 Object Detection(예시)

person : 56.95696473121643
person : 52.80924439430237
person : 70.20382285118103
person : 76.83471441268921
person : 78.70017290115356
bicycle : 79.77737784385681
person : 83.55740308761597
person : 89.43805694580078



truck : 60.93311905860901
person : 69.52624917030334
bus : 98.00647497177124
truck : 83.6944580078125
car : 71.70088291168213

ObjectDetection.py

4.9 Object Detection(구글 API)

1. console.cloud.google.com에 가입

   가입을 위해서는 카드계정을 입력. 하지만, 유료가 되기 까지는 카드에서 청구되지 않으니 걱정 안 해도 됨

2. 구글 api 관리자에 접속(https://console.cloud.google.com/)
3. API 관리자에서 사용자 인증정보 만들기를 통해 API 키와 서비스 계정 키를 생성
4. 사용자 인증정보에 해당하는 json 파일을 받는다.

   이 정보는 자신의 컴퓨터에 저장되고 윈도우 환경변수로 아래와 같이 설정

   <...> 부분이 해당 파일의 path₩filename임

   GOOGLE_APPLICATION_CREDENTIALS = <json file name>
5. 환경변수가 작동하도록 리부팅
6. 실행에 필요한 라이브러리 설치

   pip install --upgrade google-cloud-vision

   pip install --upgrade google-api-python-client

   pip install --upgrade oauth2client

4.9 Object Detection(구글 API)

- 아래는 구글 클라우드와 네이버 클라우드 API를 이용해 Object Detection을 해보는 예제임

GoogleCloudAPI.py

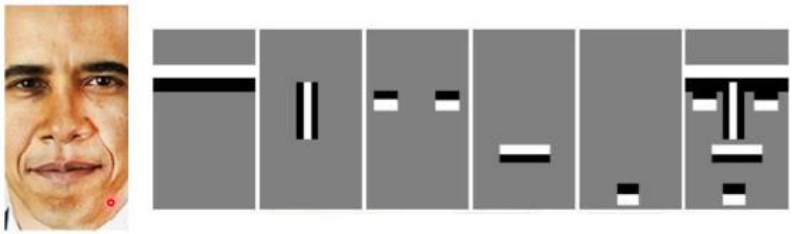
NaverFaceAPI.py

Drum,Musician,Drummer,
Musical instrument,
Tom-tom drum,
Drums, Percussionist,
Drumhead, Percussion,
Bass drum

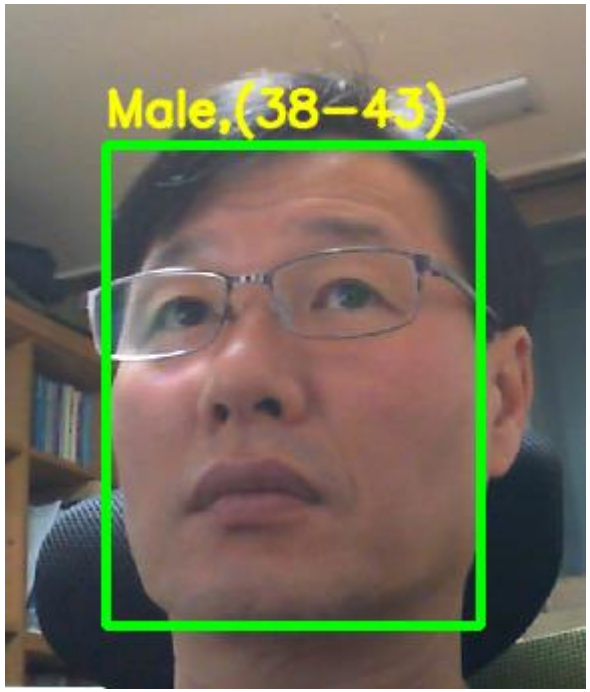
"faceCount":2},landmark":{"leftEye":{"x":350,"y":165},"rightEye":{"x":400,"y":163},"nose":{"x":377,"y":194},"leftMouth":{"x":358,"y":219},"rightMouth":{"x":395,"y":220}},"gender":{"value":"child","confidence":0.830143},"age":{"value":"2~6","confidence":0.984062},"emotion":{"value":"laugh","confidence":0.787536,"age":{"value":"6~10","confidence":1.0},"emotion":{"value":"smile","confidence":0.999406},"pose":{"value":"frontal_face","confidence":0.938255}}]}

4.9 Object Detection(Gender/Age)

- 여러 개 다양한 모양과 값을 가진 필터 중 얼굴 영상을 잘 찾아내는 필터를 AdaBoosting 알고리즘으로 조합한 필터를 개발함
- 얼굴 객체 검출 후, 성별은 남녀 두 개 클래스로 판별망에 넣어 판별하고, 나이는 나이구간 카테고리를 맞추는 판별망에 넣어 결과를 가져 옴(너무 정확함 ♥)



AgeGender.py



4.9 Object Detection(Car License Plate)

licensePlateRecog.py

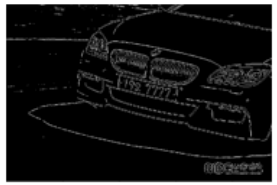
Image

Gray Image

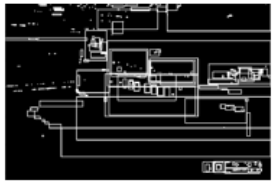
morphology Image

binary Image

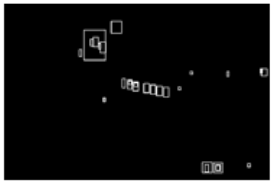
contour Image

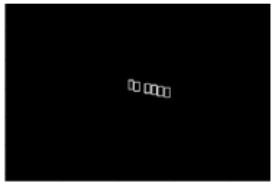


contour to rectangle



Possible Rectangle





Rectangle Cropping

Google Cloud

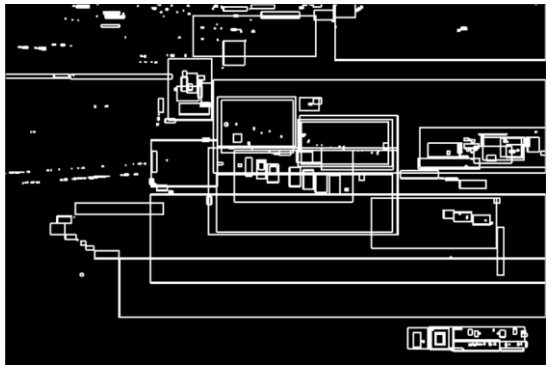
19오 7777

4.9 Object Detection(Car License Plate)

```
gray = cv2.cvtColor(img_ori, cv2.COLOR_BGR2GRAY)

structuringElement = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))

imgTopHat = cv2.morphologyEx(gray, cv2.MORPH_TOPHAT, structuringElement)
imgBlackHat = cv2.morphologyEx(gray, cv2.MORPH_BLACKHAT, structuringElement)

imgGrayscalePlusTopHat = cv2.add(gray, imgTopHat)
gray = cv2.subtract(imgGrayscalePlusTopHat, imgBlackHat)

img_blurred = cv2.GaussianBlur(gray, ksize=(5, 5), sigmaX=0)

img_thresh = cv2.adaptiveThreshold(
    img_blurred,
    maxValue=255.0,
    adaptiveMethod=cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    thresholdType=cv2.THRESH_BINARY_INV,
    blockSize=19,
    C=9
)
```

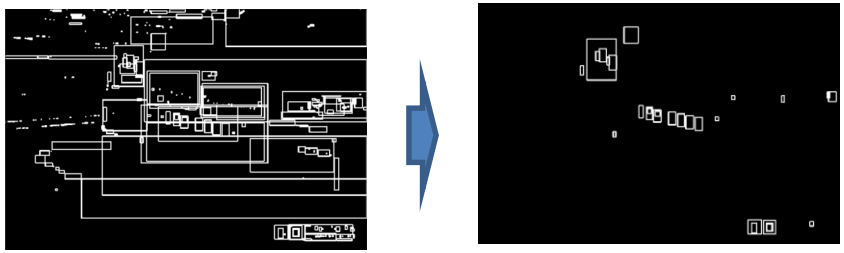
4.9 Object Detection(Car License Plate)

```
# 모든 컨투어 라인을 찾는다.
contours,_= cv2.findContours(img_thresh, mode=cv2.RETR_LIST,
method=cv2.CHAIN_APPROX_SIMPLE)
temp_result = np.zeros((height, width, channel), dtype=np.uint8)
contours_dict = []
for contour in contours:
    # Bounding Rectangle은 컨투어 라인을 둘러싸는 사각형을 그리는 방법이다.
    x, y, w, h = cv2.boundingRect(contour)
    # 사각형 그리기
    cv2.rectangle(temp_result, pt1=(x, y), pt2=(x+w, y+h), color=(255, 255, 255), thickness=2)
    # insert to dict
    contours_dict.append({
        'contour': contour,
        'x': x,
        'y': y,
        'w': w,
        'h': h,
        'cx': x + (w / 2), # cx, xy: 사각형 중심좌표
        'cy': y + (h / 2)
    })
```



4.9 Object Detection(Car License Plate)

```
# 수많은 컨투어 박스 중에 아래의 조건을 만족하지 않는 박스는 버린다.
MIN_AREA = 80                          # 박스 면적
MIN_WIDTH, MIN_HEIGHT = 2, 8           # 박스 최소 폭과 높이
MIN_RATIO, MAX_RATIO = 0.25, 1.0       # 박스 최소, 최대 비율(width/height)
possible_contours = []

 area = d['w'] * d['h']
 ratio = d['w'] / d['h']

if area > MIN_AREA and d['w'] > MIN_WIDTH and d['h'] > MIN_HEIGHT
                        and MIN_RATIO < ratio < MAX_RATIO:
        d['idx'] = cnt
        cnt += 1
        possible_contours.append(d)
```

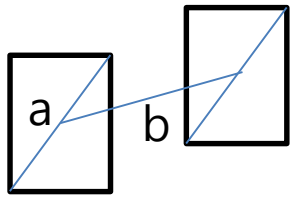


4.9 Object Detection(Car License Plate)

남은 사각형 중에 순차적으로 정렬된 사각형 리스트만 구한다.

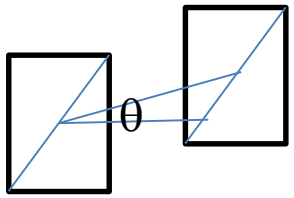
MAX_DIAG_MULTIPLYER = 5 # 5
MAX_ANGLE_DIFF = 12.0 # 12.0
MAX_AREA_DIFF = 0.5 # 0.5
MAX_WIDTH_DIFF = 0.8
MAX_HEIGHT_DIFF = 0.2
MIN_N_MATCHED = 3 # 3

MAX_DIAG_MULTIPLYER



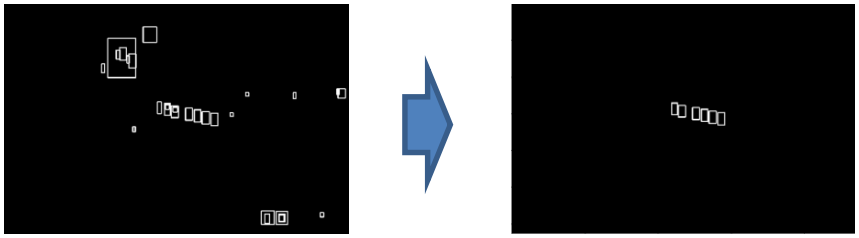
b < 5a

MAX_ANGLE_DIFF



θ < 12도

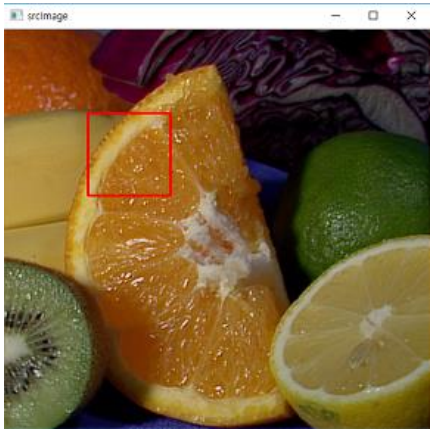
Possible Rectangle



4.10 Object Tracking

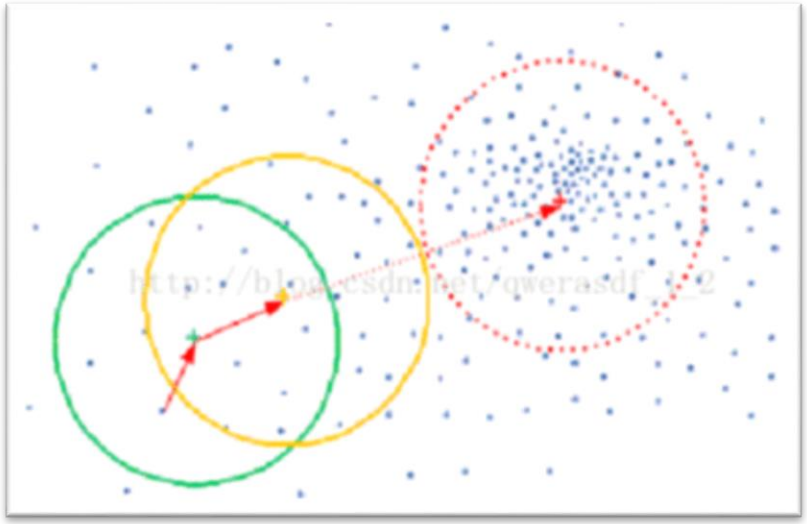
- 객체추적은 움직이는 영상에서 ROI(Region Of Interest)를 추적하는 방법임
- Mean Shift, CAM(Continuously Adaptive Mean) SHIFT 알고리즘 등이 있음

- Mean Shift 알고리즘
  - ▶ ROI의 이미지 히스토그램을 구해 분포함수를 추정한다.(RGB2HSV를 통해 H의 히스토그램 생성)
  - ▶ 추적하고자 하는 이미지의 히스토그램 역투영을 이용해 이미지를 이진화 한다.
  - ▶ Mean-shift 알고리즘을 이용해 1에 가까운 픽셀이 모여 있는 곳으로 이동함
  - ▶ 색상 외에도 다른 특징의 빈도를 이용 가능함



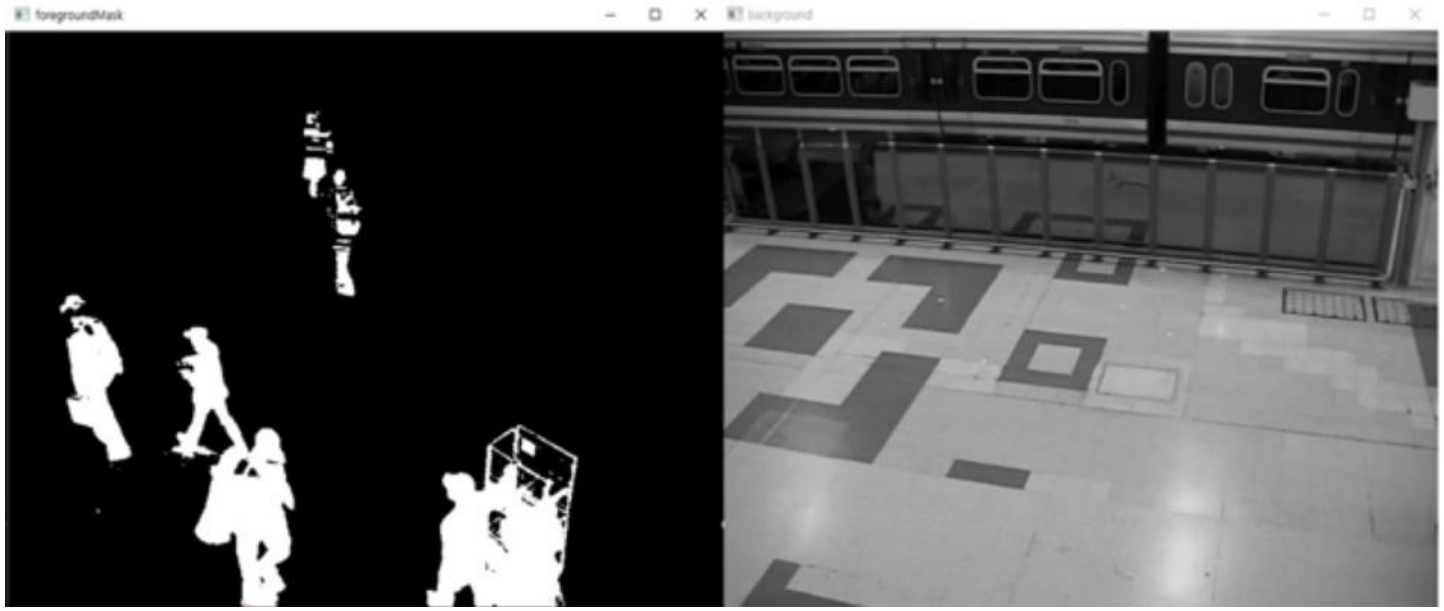
4.10 Object Tracking

- Mean Shift 알고리즘은 아래 그림에서 녹색 원 안에 있는 점들의 무게중심을 구해 노란색 원으로 이동하는 원리로 점이 가장 밀집한 곳으로 이동함
- 이 알고리즘을 이용해 이진화된 이미지에서 하얀색이 밀집된 곳으로 이동하면 객체를 추적할 수 있음
- 단점으로 객체가 멀어지거나 가까워지는 경우, 윈도우의 크기가 고정되어 있어 검출이 정확하지 않음
- 이러한 단점을 보완하기 위해 CAM SHIFT 알고리즘이 개발됨



4.10 Object Tracking

- 비디오에서 움직이는 물체를 찾아내서 구분하고자 한다면 고정된 배경과 움직이는 전경을 구분하는 것임
- 고정된 배경은 과거, n개의 이미지에서 각 픽셀 위치에서 평균, 혹은 중앙값을 구하는 것임
- 배경 이미지가 구해지면 현재 영상과 배경 차이를 구해 특정값 이상이면 움직이는 객체를 찾을 수 있음



4.10 Object Tracking

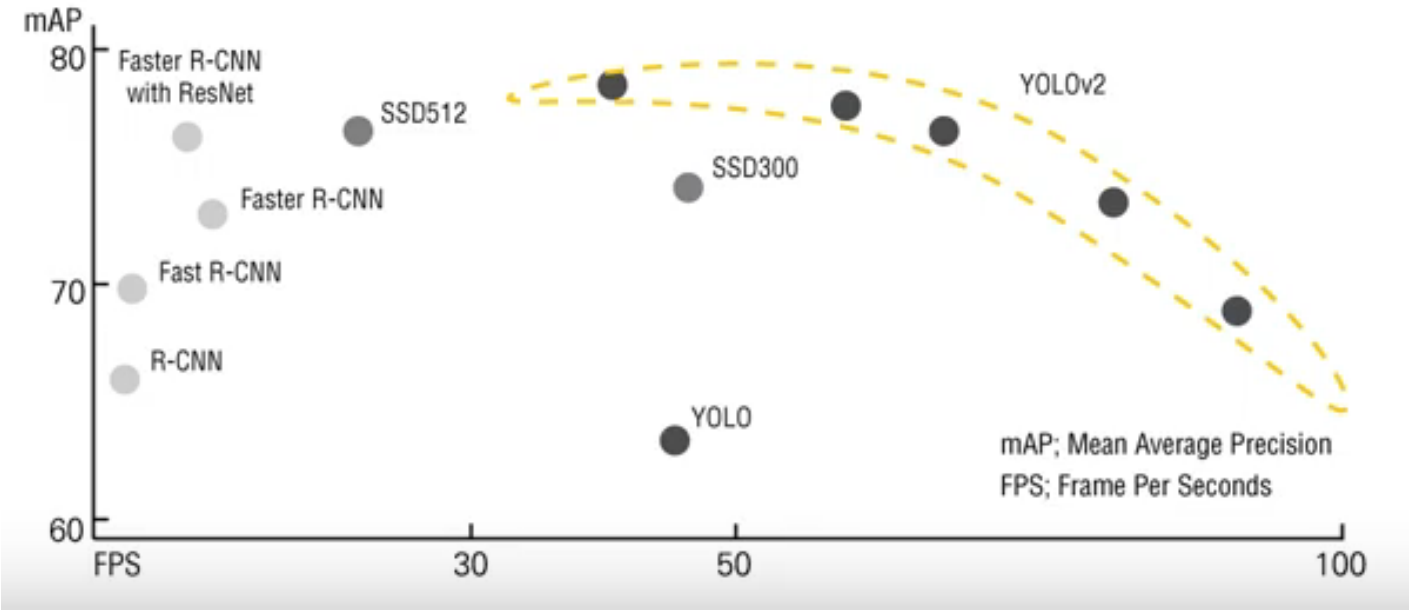
- CV2에서는 성능이 뛰어난 다양한 알고리즘을 API로 제공하고 있음

VideoTrackingAPI.py

https://drive.google.com/open?id=1qEDgC7Dczl_cm3DHNwnMRsvdTjIqv6xo

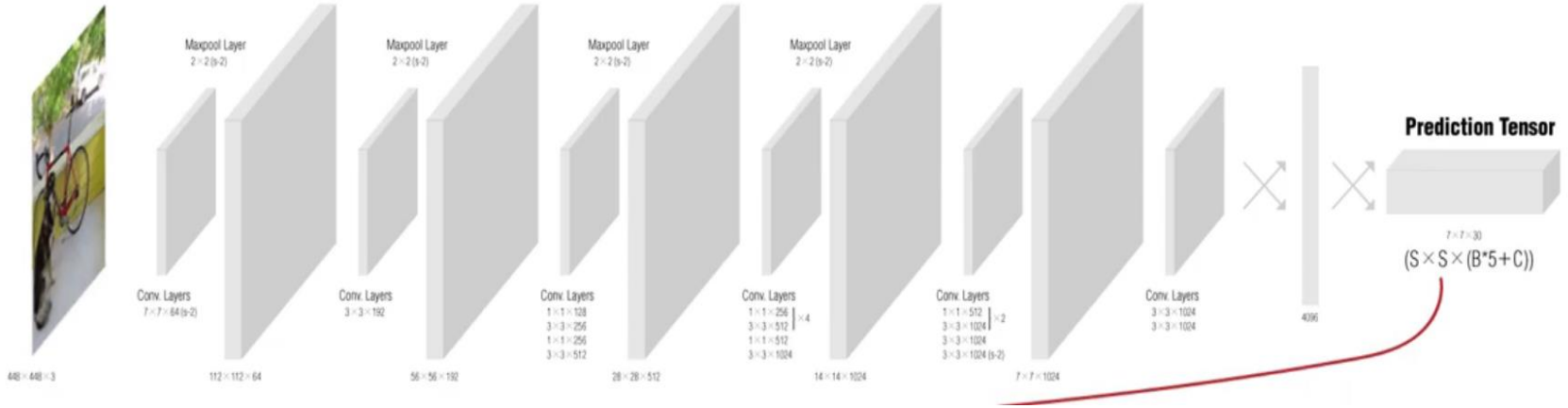
4.10 Object Tracking

- R-CNN과 같은 detection system들은 복잡한 처리과정으로 인해 이러한 Human visual system을 모방하기에는 부족한 부분들을 보임(느린 속도, 최적화의 어려움)
- YOLO(You Only Look Once)는 이미지 내의 bounding box와 class 확률을 single regression으로 간주하여, 이미지를 한 번 보는 것으로 object의 종류와 위치를 추측하는 아이디어임
- mAP: mean Average Precision, FPS: Frame Per Second 상으로 볼 때, YOLOv2가 가장 뛰어난 알고리즘임



4.10 Object Tracking

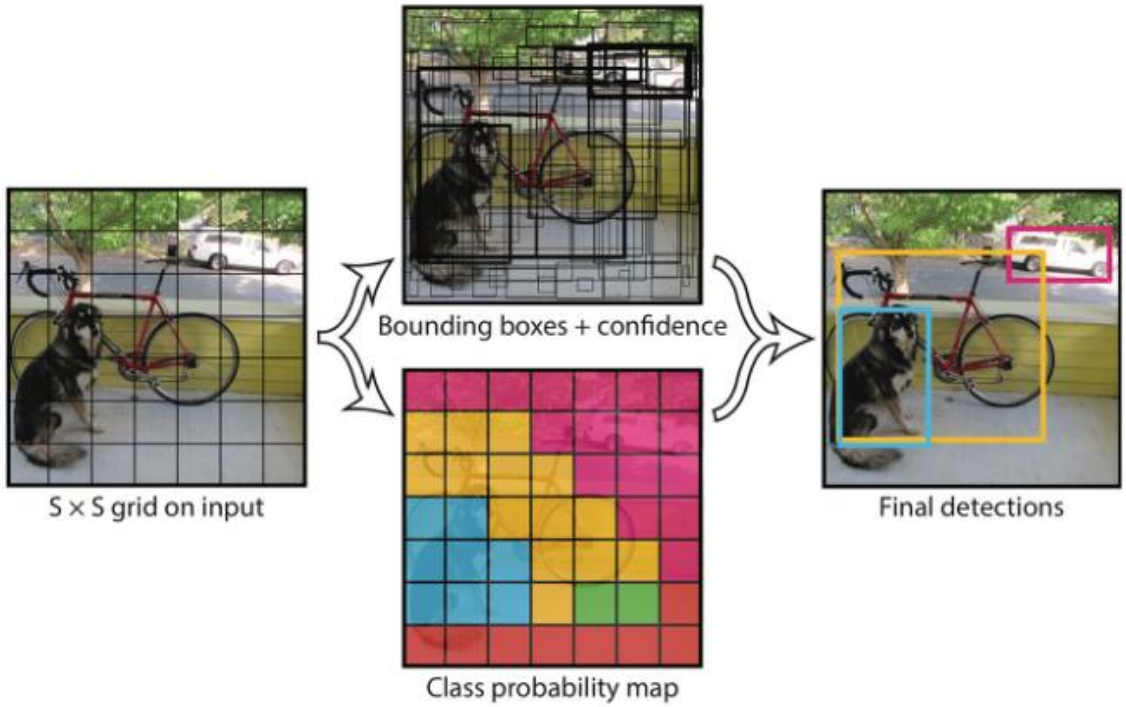
- YOLO는 이미지를 CNN 망을 통해 특징을 추출하고 이를 Prediction Tensor로 변환함
- Prediction Tensor는 7*7 이미지이고 한 장 당 30개의 정보를 가지고 있음
- 30개는 5(x, y, w, h, confidence score) * B + C 로 B = 2, C = 20임
- x, y는 바운딩 박스 중심 위치이고, w, h는 바운딩 박스 width, height, confidence score는 해당 바운딩 박스에 물체가 있을 확률임
- x, y, w, h는 각 셀에서 2개를 랜덤 결정하고 위치를 학습에 의해 조정함



4.10 Object Tracking

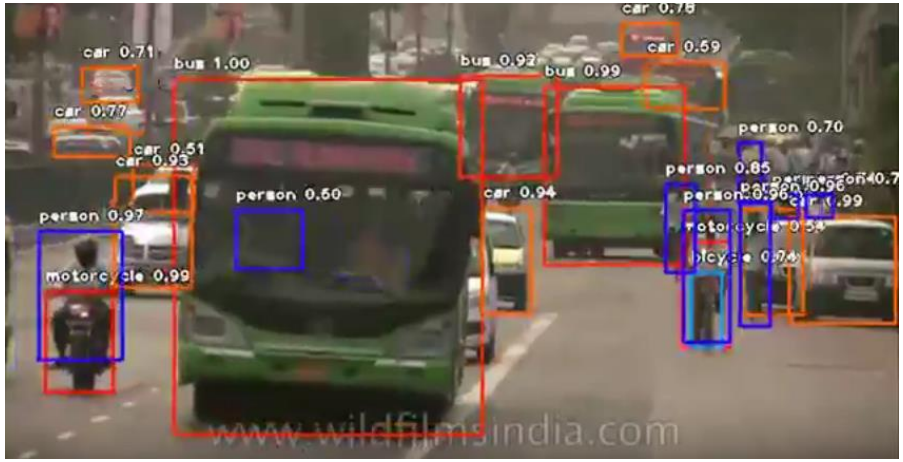
- 이 경우, 7*7*2 = 98개의 (x, y, w, h) 바운딩 박스가 나오고 각 박스에 객체가 존재할 확률과 객체의 Class 확률이 리턴됨. 미세 조정을 통해 (x, y, w, h) 를 결정함

https://docs.google.com/presentation/d/1aeRvtKG21KHdD5lg6Hgyhx5rPq_ZOsGjG5rJ1HP7BbA/pub?start=false&loop=false&delayms=3000&slide=id.g137784ab86_4_1738



4.10 Object Tracking

- YOLO API는 다수의 객체를 Detect할 수 있는 API임
- 필요 라이브러리: scipy, opencv-python, pillow, h5py, keras 설치가 필요하다.
- pip install https://github.com/OlafenwaMoses/ImageAI/releases/download/2.0.2/imageai-2.0.2-py3-none-any.whl
- https://github.com/OlafenwaMoses/ImageAI/releases/download/1.0/resnet50_coco_best_v2.0.1.h5



YOLOVideoObjectDetection.py

https://drive.google.com/open?id=1dKmpqZB-i-S4BK5JKefzWNgILsJZkxy0

감사합니다