Deep Learning

데이터 수집과 전처리 (영상처리)

강사 양석환

영상처리를 위한 데이터셋 만들기

이미지 수집

- 영상처리를 위해서는 일단 이미지 데이터가 필요함
  - 지도학습 모델의 경우
    - 이미지 데이터와 함께 라벨 데이터도 필요함
  - 비지도학습 모델의 경우
    - 비지도 학습 모델, 자기 지도(Self-Supervised) 학습 모델, 생성 모델 등에서는 라벨 데이터를 생략할 수 있음

자기 지도(Self-Supervised) 학습 모델
라벨이 지정되지 않은 데이터에서 학습을 진행하는 모델로 지도학습과 비지도학습의
중간형태를 가진 모델이며 반지도학습과 비슷한 위치에 있음

GAN, AutoEncoder 등의 모델을 들 수 있는데, 이 모델들은 생성 모델로 분류하기도 하고
자기 지도 모델로 분류하기도 함(뭔가 분류하기가 애매함)

이미지 수집

- 머신러닝/딥러닝 프로젝트의 첫 단계는 데이터 수집 → 영상처리에서는 이미지 데이터 수집
- 이미지 데이터의 수집 방법
  - 장비를 활용하는 방법
    - 예시: 교차로에 카메라를 설치하여 데이터를 얻는 방법 등
  - 기술을 활용하는 방법
    - 자동차 부품 사진을 얻기 위하여 디지털 카탈로그에 접속하여 데이터를 얻는 방법 등
  - 비용을 투자하는 방법
    - 위성 이미지 아카이브를 구매하는 방법 등

참고
머신러닝/딥러닝 프로젝트의 성공을 위해서는 이미지 데이터만이 아니라
이미지를 획득할 때의 맥락(Context)에 대한 메타데이터가 요구될 때도 있음
(예) 교차로의 사진을 찍을 당시의 날씨, 해당 교차로의 모든 신호등 상태 등

이미지 수집

- 사진 데이터
  - 가장 흔하게 얻을 수 있는 이미지 데이터
  - 이미지를 수집할 때는 카메라의 배치, 이미지의 크기, 해상도 등 데이터 수집의 기준을 먼저 결정하여야 함
  - 고해상도의 이미지가 늘 좋은 것이 아님. 다양한 상황에 맞는 이미지가 요구됨
  - 고해상도 이미지 사용 시의 단점
    - 큰 이미지로 모델을 학습시키려면 모델 자체의 크기도 커야 함
    - 머신러닝/딥러닝 장비는 메모리의 제한이 있음
    - 고해상도 이미지는 노이즈가 두드러짐
    - 이미지의 해상도가 높을수록 수집과 저장에 오랜 시간이 소요됨
    - 고해상도의 이미지는 전송 시에도 많은 시간이 소요됨

예: 256x256 이미지 학습→ 128x128 이미지 사용 시의 4배의 파라미터 필요

이미지가 클 수록 배치로 묶을 수 있는 이미지 수 감소 → 정확도 감소 가능성이 높아짐

해상도가 낮아질수록 훈련속도와 정확도 상승

특히 Edge Computing의 경우 치명적

이미지 수집

- 이미징 데이터
  - X선, MRI, 분광기, 레이더(Radar), 라이더(Lidar) 등의 장비에서 생성하는 2, 3차원의 영상
  - 계측 장치가 여러가지 수치를 측정한다면 파장별 반사, 도플러 속도, 기타 측정값을 이미지의 개별 채널로 처리 가능
  - X 선의 경우
    - 3차원 객체의 투영을 표시하며 일반적으로 회색조(Grayscale) 이미지인 경우가 많음
    - 일반적인 사진의 경우 3개 채널(RGB) 포함하지만 회색조 이미지의 경우 1개 채널 보유
  - 단층 촬영의 경우
    - 단층 촬영에서의 투영은 얇은 3차원 슬라이스 영상이며 여러 장의 단면 이미지를 생성
    - 각 단면을 단일 이미지의 채널로 취급할 수 있음

이미지 수집

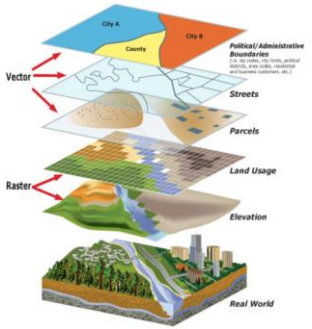
- **센서의 특성에 따른 이미징 데이터의 경우**
  - 극격자
    - 레이더와 초음파 데이터는 극좌표계(Polar Coordinate System)으로 되어 있음
    - 극좌표계의 2차원 데이터를 입력으로 사용할 경우, 극좌표계를 데카르트 좌표계로 변환할 수 있음

각 좌표계 별 장단점
- 극좌표계: 이미지의 픽셀 크기가 균일하지 않음. 보간 또는 중복되는 픽셀이 없음
- 데카르트 좌표계: 픽셀 크기가 일정함. 재 매핑 시 많은 데이터가 누락, 보간, 집계됨 → 학습하기가 매우 까다로움

  - 위성 채널
    - 위성 이미지 작업 시, 이미지를 지구 좌표에 재 매핑하는 것보다 원래의 위성 뷰 또는 시차 보정 격자에서 작업하는 것이 좋음
    - 투영된 지도 데이터를 사용하는 경우 데이터 원본 투영에 대해 학습을 수행하는 것이 좋음
    - 같은 장소에 대하여 거의 동시에 수집된 파장별 이미지들은 각각 별도의 채널로 취급함

이미지 수집

- **지리공간 레이어**
  - **여러 개의 지도 레이어가 서로 다른 투영에서 수집된 경우**
    - 동일한 투영으로 재매핑하고 픽셀을 정렬
    - 각 레이어를 이미지의 채널로 취급해야 함
    - 픽셀의 크기 변화를 설명할 수 있도록 픽셀의 위도를 모델에 대한 추가 입력 채널로 포함하는 것이 유용함



이미지 수집

- 개념 증명(Proof of Concept, PoC)
  - 본격적으로 프로젝트를 시작하기 전에 프로젝트의 가능성, 타당성을 증명하기 위한 PoC의 경우
    - 당장 쓸 수 있는 데이터가 없거나 데이터의 수집에 오랜 시간이 소요되는 경우가 많음
    - 데이터 수집 루틴에 투자하기 전에 프로젝트의 타당성을 검증하기 위한 유사한 데이터를 구매하는 것이 효율적
    - 이미지 데이터 구매 시에는 실제 프로젝트에서 사용할 수 있는 이미지와 품질, 해상도 등과 유사한 이미지의 조건을 염두에 두고 선택해야 함
  - 이미지를 시뮬레이션하여 PoC를 진행하는 경우
    - 구하기 어려운 데이터는 새롭게 이미지를 만드는 것보다 기존 이미지를 수정, 활용하여 시뮬레이션을 수행하는 것이 좋음

데이터 타입

- 영상처리에서 사용되는 데이터 타입
  - 사진 데이터 외에도 다양한 유형의 이미지를 적용할 수 있음
  - 수학적으로 4차원 텐서(배치 x 높이 x 너비 x 채널) 입력만 있으면 머신러닝/딥러닝을 적용할 수 있음

    → 데이터를 이런 형태로 표현하기만 하면 영상처리(컴퓨터 비전) 기법을 사용할 수 있음
  - 특정 기술이 유효한지 판단하기 위해서는 기본 원리를 이해하여야 함
    - 예: 컨볼루션 필터는 인접한 픽셀 사이에 공간적 상관관계가 있을 때만 잘 작동함

데이터 타입

- 채널
  - 일반적으로 3개의 채널(RGB)이 있는 24비트 RGB 이미지로 저장됨
  - 각 채널은 0~255 범위의 8비트 숫자로 표시됨
  - 컴퓨터로 생성한 이미지 중에는 픽셀의 투명도를 표시하는 알파 채널을 포함하는 경우도 있음
  - 스케일링
    - 머신러닝/딥러닝 프레임워크 및 사전 학습된 모델은 픽셀 값을 [0, 255]에서 [0, 1]로 조정하는 경우가 많음
    - 머신러닝/딥러닝 모델은 일반적으로 알파 채널을 무시함
  - 채널 순서
    - 일반적인 이미지 입력의 모양은 [높이, 너비, 채널 수]로 구성됨(RGB 이미지: 3채널, 회색조 이미지: 1채널)
    - Tensorflow를 포함한 대부분의 경우, 채널의 위치는 가장 마지막에 표시 함

데이터 타입

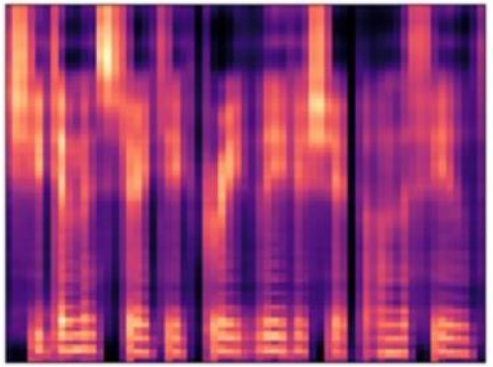
- 지리 공간 데이터(Geospatial Data)
  - 지도 레이어에서 생성되거나 드론, 위성, 레이더 증의 원격 감지 결과(Remote Sensing)로 생성됨
  - 래스터 데이터(Raster Data)
    - 채널로 처리할 수 있는 래스터 밴드(픽셀 값의 2차원 배열)가 포함된 지리 공간 데이터
    - 래스터 밴드
      - 특정한 2차원적 구조에서 분할된 직사각형으로 한정된 2차원적 기하원시요소
      - 인구 밀도, 토지피복 유형, 범람 성향 등 다양한 토지 영역을 나타내는 밴드가 있음
    - 래스터 데이터에 컴퓨터 비전 기술을 적용하려면 개별 밴드를 읽고 쌓아 올려 이미지를 형성하면 됨
  - 래스터 데이터 외에도 도로, 강, 주 또는 도시의 위치와 같은 벡터 데이터도 있을 수 있음
    - 모델에서 데이터를 사용하기 전에 데이터를 래스터화 해야 함

데이터 타입

- 원격 감지
  - 주로 이미징 기기에 의해 수집됨
    - 장비가 카메라인 경우: 3개의 채널이 있는 이미지가 됨
    - 여러 기기가 있는 위성 또는 기기가 여러 주파수에서 작동할 수 있는 경우: 채널 수가 많은 이미지가 됨
  - 원격 감지 이미지 중에는 시각화를 위해 색이 입혀지는 것도 있음
    - 이러한 가공된 컬러 이미지보다는 기기에서 감지한 원시(Raw) 데이터 값을 그대로 사용하는 것이 좋음
  - 원격 감지 이미지는 데이터 누락이 자주 발생함 → 가능하다면 누락된 부분은 잘라서 버리는 것이 좋음
  - 원격 감지 이미지는 가능한 한 (또는 반드시) 정규화를 수행할 것
  - 원격 감지 이미지에는 이상치가 포함될 가능성이 높음 → 이미지의 크기 조정 이전에 적절한 범위로 자를 것
- 지리 공간 데이터, 원격 감지 이미지는 모델에 입력되기 전에 상당량의 처리가 요구됨 → 자동화된 단계/파이프라인을 갖추는 것이 좋음

데이터 타입

- 오디오와 비디오
  - 오디오 데이터는 1차원 신호, 비디오 데이터는 3차원 신호
  - 시제품을 간단히 만들 경우, 오디오 및 비디오 데이터에 이미지 머신러닝/딥러닝 기술을 응용할 수 있음
  - 스펙트로그램
    - 오디오 데이터
      - 오디오를 덩어리로 나눈 뒤 시간 창(time window)에 머신러닝/딥러닝 적용 (시간 창의 크기는 검출 대상에 따라 달라짐)
      - 처리 결과는 1차원 신호가 되므로 Conv2D 대신 Conv1D를 사용하여 오디오 신호를 처리할 수 있음



데이터 타입

- **프레임별로 처리하는 접근 방식**
  - 동영상(비디오)은 여러 장의 이미지, 즉 프레임(Frame)으로 구성됨
  - 동영상을 처리하려면 개별 프레임에 대해 이미지 처리를 수행한 결과를 종합하여 전체 동영상을 분석하면 됨
  - OpenCV 패키지를 주로 활용함
- **Conv3D 모델을 사용한 접근 방식**
  - 동영상을 처리할 때 한 번에 한 프레임 씩 처리하는 대신, 프레임의 이동 평균(Rolling Average)을 계산하여 컴퓨터 비전 알고리즘을 적용할 수 있음 → 화면에 노이즈가 많을 때 특히 유용함
  - 이동 평균 방법은 프레임 간의 상관관계를 이용해서 이미지의 노이즈를 제거함
- **순환신경망(RNN)의 시퀀스 방법을 사용한 접근 방식**
  - 시계열 데이터에 적합한 RNN 모델을 활용하는 방법
  - 동영상 시퀀스의 RNN 적용은 훈련이 어려우므로 Conv3D 방식이 더 실용적임

수동 라벨링

- 라벨링 지정 작업
  - 머신러닝/딥러닝 프로젝트에서 데이터 과학 팀이 참여하는 첫 번째 단계
  - 라벨링을 자동화 하더라도 개념 증명의 처음 몇 개의 이미지는 거의 항상 수동으로 라벨링 됨
  - 라벨링의 형식과 구성
    - 문제 유형(이미지 분류 또는 객체 검출 등)에 따라 결정
    - 이미지에 여러 라벨이 있을 수 있는지, 아니면 하나만 있을 수 있는지에 따라 결정

수동 라벨링

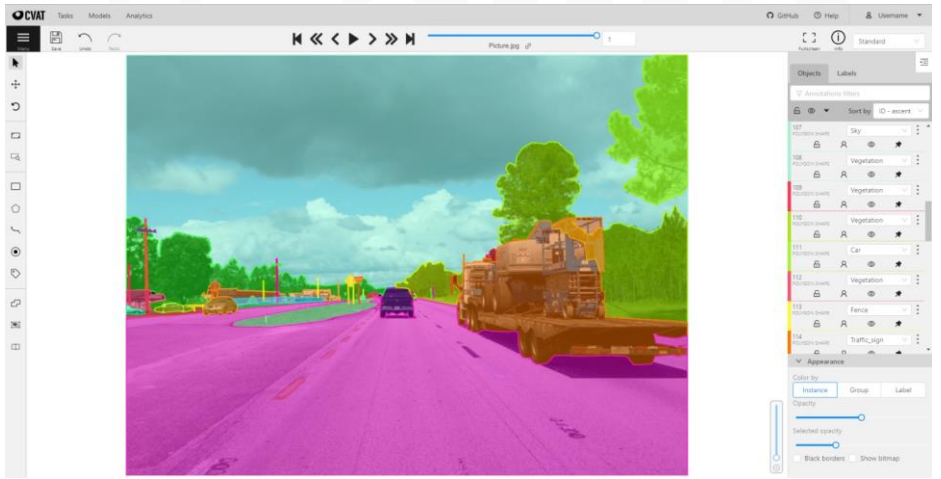
- **수동 라벨링의 기록 방법**
  - 폴더 구조를 이용하는 방식
    - 평가자가 각 이미지를 라벨 별 폴더로 이동시키는 방법
    - 대부분의 운영체제에서 이미지 미리보기 및 이미지 그룹별 선택/이동 기능을 제공하므로 빠르게 작업할 수 있음
    - 문제점: 이미지에 여러 라벨이 있을 수 있는 경우, 데이터의 중복이 발생함
  - 메타데이터 테이블을 이용하는 방식
    - 폴더 방식의 대안으로 제안된 방식
    - 최소한 두 개의 열이 있는 메타데이터 테이블(CSV 파일 등)에 라벨을 기록하는 방법
    - 한 열에는 이미지 파일의 URL을 기록하고, 다른 열에는 해당 이미지의 유효한 라벨 목록을 기록
  - 폴더 방식의 효율성 + 메타데이터 테이블 방식의 일반화 능력을 절충한 접근 방식
    - 이미지를 폴더로 구성한 뒤 스크립트를 써서 메타데이터 테이블을 생성하거나 함

수동 라벨링

- 다중 라벨
  - 이미지가 여러 라벨과 연결될 수 있는 경우
    - 이미지를 두 폴더에 복사해 넣고 메타데이터 테이블의 두 줄을 만드는 식으로 할 수 있음
    - 중복이 있는 경우 다중 라벨 클래스 문제를 훈련하기가 어려움 → 일치하는 모든 범주를 라벨 열에 넣는 것이 좋음
- 객체 검출
  - 이미지에 포함된 객체의 경계 박스 정보를 메타데이터 파일에 포함
    - 경계 박스 꼭지점들의 좌표를 미리 정해 둔 순서에 따라 메타데이터의 세 번째 열에 기록
- 영역 분할(세분화)
  - 객체 검출 시와 비슷하나 경계 박스가 다각형임

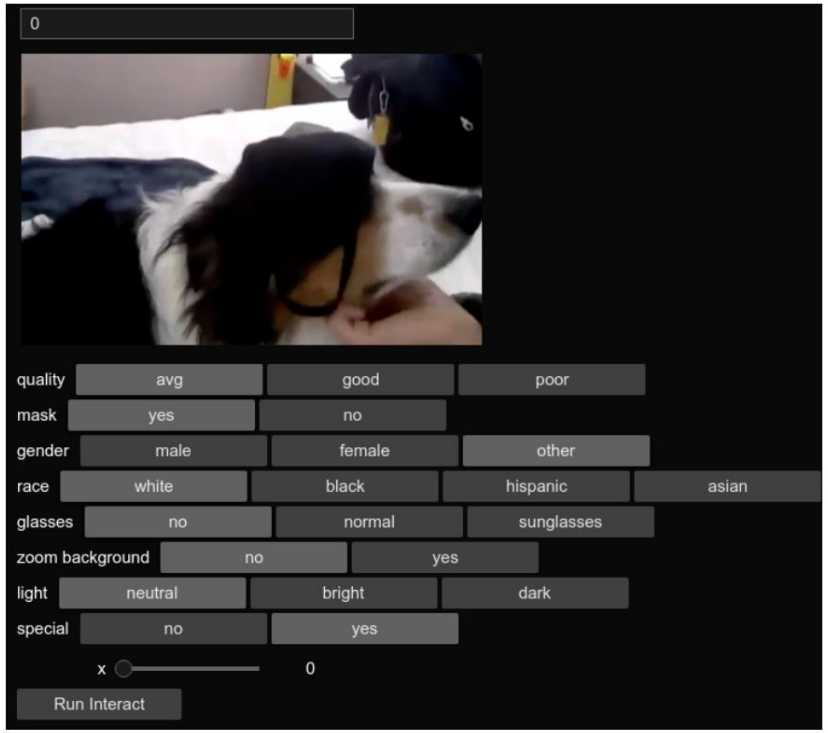
대규모 라벨링

- 수많은 이미지를 수동으로 라벨링하기는 어려움 → 라벨링 도구 활용
- 라벨링 도구
  - 라벨링 도구는 이미지를 화면에 표시할 수 있어야 함
  - 평가자가 유효한 범주를 빠르게 선택하고 평가를 데이터베이스에 저장할 수 있어야 함
  - 객체 식별(검출) 및 이미지 세분화 지원 도구는
    - 주석을 달 수 있어야 함
    - 경계 박스나 다각형을 이미지 픽셀 좌표로 변환할 수 있어야 함



대규모 라벨링

- **다중 과업**
  - **여러 과업을 위해 이미지를 라벨링 해야 하는 경우**
    - **예) 꽃 이미지에 품종, 색상, 장소, 파종법 등을 지정해야 하는 경우**
    - **주피터 노트북의 상호작용 기능을 이용하여 작업능률을 높일 수 있음**



Example of custom Jupyter Notebook labeling process.

대규모 라벨링

- 수동 라벨링의 문제점은 인적 오류 및 불확실성
  - 투표 시스템 구현으로 개선할 수 있음
    - 같은 이미지를 평가자 두 명에게 표시
    - 평가자들의 의견이 일치하면 해당 라벨이 이미지에 할당됨
  - 평가자들이 합의하지 않는 다면 다음 중에서 선택 가능
    - 모호한 데이터를 사용해 훈련하지 않으려면 이미지를 폐기함
    - 이미지를 중립 클래스에 속하는 것으로 간주함
    - 세 번째 평가자가 최종 라벨을 결정하도록 하여, 사실상 3명의 평가자에 의한 과반수 투표 실시
- 라벨링 서비스 활용

자동 라벨링

- 자동 라벨링
  - 100% 정확하지 않더라도 유용한 경우가 많음
  - 평가자가 이미지에 라벨을 하나하나 붙이는 것보다 자동으로 얻은 라벨을 수정하는 것이 훨씬 효율적
  - 관련 데이터의 라벨
    - 이미지가 나타내는 문서 등에서 엔티티 추출을 수행하여 이미지 라벨을 획득 가능
    - 이미지 픽셀의 일부를 보고 실측자료를 확인하여 라벨링 적용
  - “떠드는 학생(Noisy Student)” 모델
    - 일부 이미지에 수동 라벨링 지정 → 수동 라벨링 된 이미지를 사용하여 작은 머신러닝 모델 훈련(지도학습)
      → 훈련된 모델을 사용하여 라벨링되지 않은 이미지의 라벨을 예측
      → 라벨링된 이미지 + 의사라벨 지정 이미지 → “학생모델”이라는 큰 머신러닝 모델 학습 → “학생모델”을 교사로 삼아 반복 수행

자동 라벨링

- 자기 지도 학습
  - 오토인코더 등의 모델은 이미지 자체가 라벨의 역할을 수행함
  - 라벨이 일정 시간 후에 알려지게 되는 경우에 자기 지도학습이 가능함
    - 예) 의료 이미지는 환자의 최종 결과에 따라 라벨링 될 수 있음
  - 이러한 라벨링 방법은 미래에 일어날만한 일에 대한 예측에 활용할 수 있음

편향

- 데이터의 편향
  - 모델을 적용했을 때 원하지 않는 동작으로 이어지게 하는 데이터셋의 특성
- 가장 이상적인 데이터셋
  - 시스템에서 가장 좋은 성능을 내도록 모델을 훈련할 수 있는 데이터셋
- 편향된(Biased) 데이터셋
  - 특정 예가 데이터셋에 너무 많이 / 적게 나타나서 해당 시나리오에 대해 정확도를 떨어뜨리는 경우

- 편향의 원인
  - 선택 편향(Selection Bias)
    - 불완전한 데이터 수집, 즉 특정 범주가 제외되거나 제대로 샘플링되지 않도록 데이터 소스를 제한한 결과로 발생
    - 선택 편향이 발생하는 경우
      - 특정 유형의 데이터가 다른 유형보다 수집하기 쉬운 경우
      - 한정된 기간에 수집한 데이터셋으로 훈련한 뒤, 실제 서비스는 다양한 시간에 운영하는 경우 등
      - 이상치 가지치기 및 데이터셋 정리 결과로 발생할 수 있음
    - 선택 편향을 해결하려면 역으로 프로덕션 시스템에서부터 작업해야 함(예시)
      - 어떤 유형의 주택을 식별하고자 하는가?
      - 데이터셋에 해당 유형의 주택의 예가 충분히 있는가?
        - 그렇지 않은 경우의 해결책은 이러한 이미지를 사전에 수집하는 것

- **측정 편향(Measurement Bias)**
  - **측정 편향이 발생하는 경우**
    - **훈련을 위해 이미지를 수집하는 방식과 프로덕션에서 수집하는 방식의 차이로 인해 발생하는 경우**
      - 예: 훈련 이미지 촬영에는 고급 카메라 사용, 프로덕션 시스템에는 성능이 떨어지는 기성품 카메라를 사용하는 경우
    - **훈련과 프로덕션에 데이터를 제공하는 사람이 달라서 발생하는 경우**
      - 예: 전문가가 찍은 사진으로 훈련 데이터 구성, 일반인이 찍은 사진으로 실제 적용
    - **여러 사람이 이미지 라벨링을 수행하는 경우**
      - 각 평가자 별로 다른 기준을 가질 때, 라벨링의 비일관성으로 인해 발생할 수 있음

- **확증 편향(Confirmation Bias)**
  - 데이터를 수집하는 당시에는 존재가 드러나지 않지만, 해당 데이터셋으로 훈련된 모델에서 혼란을 일으킬 수 있음
  - 확증 편향이 발생하는 경우
    - 실세계에서 데이터를 수집한 결과로 발생하는 경우
      - 예: 소방관은 남자가 대부분이므로 소방관 이미지의 무작위 샘플을 수집한다면 데이터의 전원이 남자가 될 수도 있음
  - 확증 편향은 데이터셋이 현실 세계를 반영할 때 사회의 기존 편견이 어떻게 증폭되는지를 보여줌
  - 확증 편향은 라벨 측면에서 기존 편향을 부추길 수도 있음
  - 권장되는 접근 방식
    - 잠재적 편향에 대한 인식
    - 편향을 완화시키기 위한 적극적인 데이터 수집

- 편향 감지
  - 슬라이스 평가
    - 편향을 감지하기 위해 슬라이스 평가를 할 수 있음
    - 모델의 목적 함수를 그룹 구성원에 대해서만 계산해서 그것을 비구성원에 대한 지표와 비교한 다음
    - 슬라이스 지표가 전체 데이터셋의 지표와 매우 다른 그룹을 조사
  - 베이즈 접근 방식 적용

영상처리를 위한 데이터 전처리

전처리를 하는 이유

- 이미지 원본을 모델에 집어넣기 전에 전처리를 거쳐야 하는 경우가 많음
- 전처리의 목표
  - 모양 변환
    - 이미지 텐서가 모델의 입력 레이어에서 예상하는 모양을 갖게 함
    - 입력 이미지를 일정한 크기로 재단
  - 데이터 품질 개선(변환)
  - 모델 품질 향상
    - 데이터에 대해 훈련된 모델의 정확도를 개선하는데 도움이 되는 변환 수행
    - 일부 변환은 모델이 학습된 데이터셋의 유효 크기를 늘려 모델 품질을 개선하는데 도움이 될 수 있음

전처리의 종류

- 크기와 해상도 변환
- 훈련-서빙 왜곡
  - 훈련 파이프라인과 추론 파이프라인에 차이가 있을 때, 훈련 중에는 나타나지 않던 예상치 못한 잘못된 작동이 추론 시에 일어나는 것
  - 훈련-서빙 왜곡을 방지하려면 훈련과 추론에서 동일한 코드를 사용하는 것이 이상적임
  - 훈련과 추론 간에 코드를 재사용하기 쉬울수록 미묘한 차이가 나타나지 않고 훈련-서빙 왜곡이 발생하지 않을 가능성이 큼
  - 함수의 재사용, 모델 내에서의 전처리 수행, tf.transform을 사용한 일련의 전처리 및 데이터 정리 루틴의 수행, 데이터 증강, 공간적 변환(좌우/상하 반전, 회전 등) 등의 작업을 사용할 수 있음

전처리의 종류

- 색상 왜곡
  - 쉽게 사용할 수 있는 일련의 증강 레이어에 얽매이지 말 것
  - 모델이 프로덕션에서 접할 수 있는 이미지의 변형 유형
    - 머신러닝/딥러닝 모델에 제공되는 사진은 조명의 측면에서 상당히 다를 수 있음
    - 훈련 이미지의 밝기, 대비, 채도 등을 무작위로 변경해 데이터를 증강
      - 훈련 데이터셋의 유효 크기를 늘리고
      - 모델을 보다 탄력적으로 만들 수 있음
    - Keras에는 여러 빌트인 데이터 증강 레이어가 지원되지만 대비 및 밝기 변경은 현재 지원하지 않음
      - 직접 구현할 필요가 있음

전처리의 종류

- 정보 삭제
  - 데이터 증강과 반대로 이미지에서 정보를 삭제함으로써
  - 훈련 과정을 더욱 탄력적으로 만들어 주고
  - 모델이 이미지의 중요한 특징에 주목하도록 도와줌
  - 정보 삭제 기법
    - 컷아웃(Cutout): 훈련 중에 정사각형 영역을 무작위로 마스킹 → 정보 없는 부분 무시, 구별되는 부분 주목
    - 믹스업(Mixup): 한 쌍의 훈련 이미지를 선형으로 보간 → 해당 보간된 라벨 값을 라벨로 할당
    - 컷믹스(CutMix): 컷아웃+믹스업. 서로 다른 훈련에서 패치를 잘라서 패치들의 영역에 비례해 실측 라벨을 섞음
    - 그리드마스크(GridMask): 삭제된 영역의 밀도와 크기를 제어하면서 균일하게 분포된 정사각형 영역 삭제
      → 이미지가 의도적으로 수집됨. 균일하게 분포된 정사각형 영역은 배경이 되는 경향이 있음

전처리의 종류

- 입력 이미지 형성
  - 지금까지의 전처리 연산
    - 단순히 입력 이미지를 수정하고 입력되는 모든 이미지에 대해 단일 이미지를 모델에 제공 → 일대일 방식
  - 전처리 파이프라인 활용
    - 각 입력을 여러 이미지로 나눈 다음 훈련 및 추론을 위해 모델에 제공
  - 이미지 형성 방법: 타일링
    - 매우 큰 이미지가 있고, 큰 이미지의 일부에 대해 예측을 수행한 다음, 조합하는 방법
      - 지리 공간 이미지, 의료 이미지 및 감시의 경우가 많음
    - 타일과 해당 라벨을 사용하여 이미지 분류 모델 학습
    - 타일을 생성하는 간격을 줄임으로써 훈련 데이터셋 증강 가능

영상처리를 위한 라벨링 실제 사례

학습을 위한 데이터 레이블링 과정

- 딥러닝에서 가장 중요한 것은 데이터(특히 학습을 위한 데이터)

- 같은 계열의 목적을 가진 모델이라면

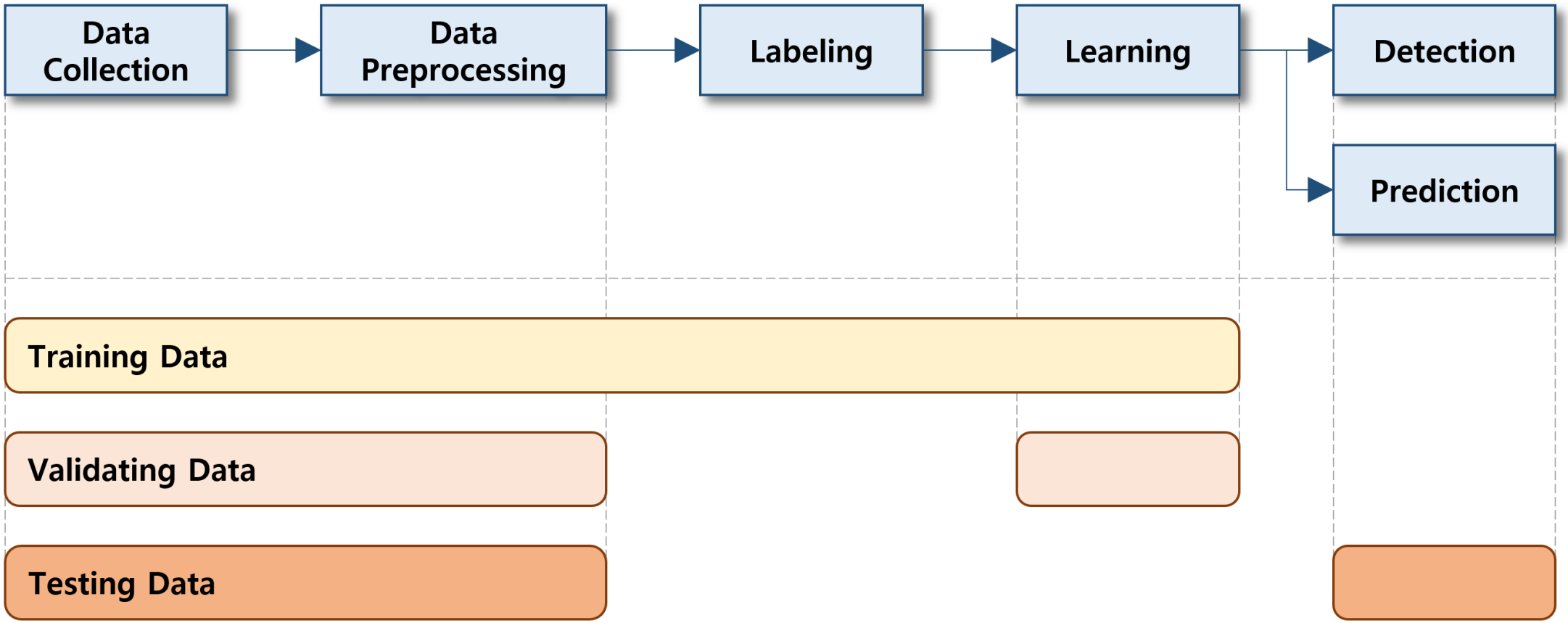
  → 코드의 변형없이 데이터만 교체해서 서로 다른 분야에 적용할 수 있음

- CNN을 비롯한 영상인식, 분류 등을 목적으로 하는 딥러닝 모델은

  → 학습의 결과가 옳은지 그른지 확인할 수 있도록 각각의 영역을 구분하고 표시를 해 둠

  → 이 작업 과정이 **레이블링** 작업 과정

이미지 인식 처리에서 데이터 셋의 적용 범위



AI 개발 최초 작업 시 문제점

정보가 없다

차량 번호판 인식용 학습 데이터셋 만들기 사례

- 참여자
  - 포항공대 2~3학년 인턴사원 3명
  - AI / 프로그래밍에 대한 기본적인 교육, 실습은 학교에서 이수
  - 데이터셋 개발 경험 없음
  - 실습 프로젝트 경험은 있으나 실무 관련 프로젝트 경험은 없음

차량 번호판 인식용 학습 데이터셋 만들기

- 필요한 데이터
  - 번호판을 장착한 자동차의 사진
  - 자동차의 사진에서 번호판 영역에 적용된 레이블링 데이터
  - 다양한 형태의 번호판 사진
  - 기타 학습 조건들…

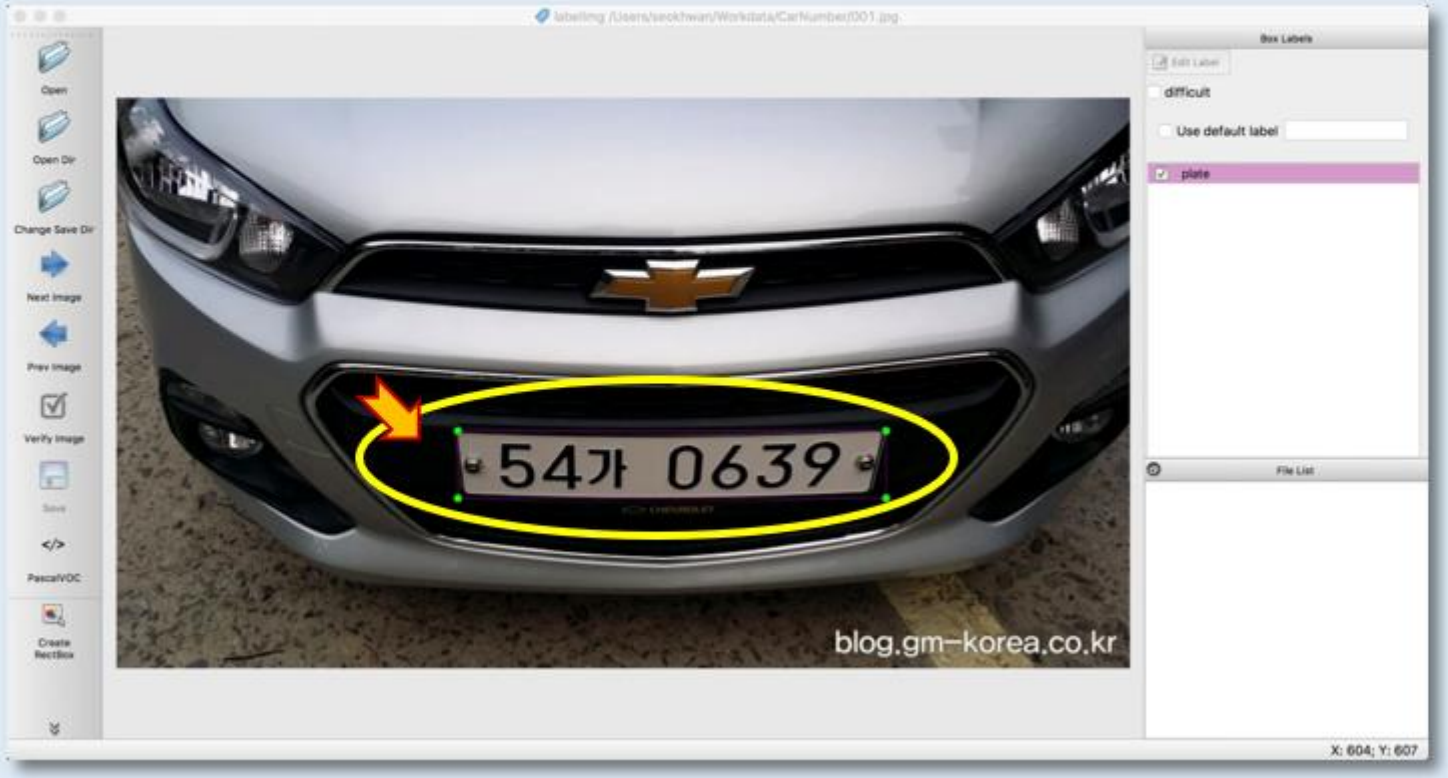
1차 시도: 이미지데이터 수집

- Google 검색을 통하여 약 50개의 차량 이미지 수집
- 전체 프로세스에 대한 과정을 처음 시도함
- 진행 방법이 올바른지에 대한 확인은 불가능한 상태

1차 시도: 이미지 레이블링 작업

1차

- 레이블링 도구: LabelImg

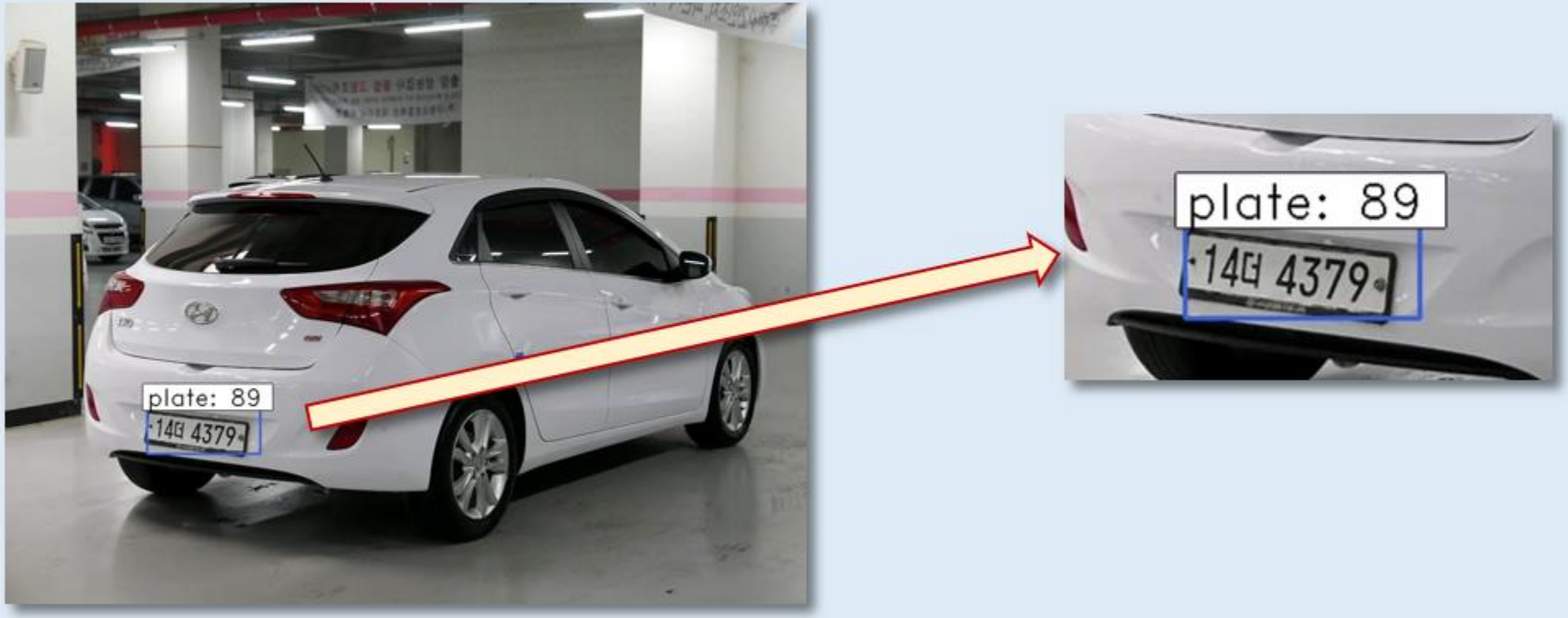



```
<annotation>
    <folder>CarNumber</folder>
    <filename>001.jpg</filename>
    <path>/training/001.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>980</width>
        <height>551</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>plate</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>330</xmin>
            <ymin>321</ymin>
            <xmax>744</xmax>
            <ymax>386</ymax>
        </bndbox>
    </object>
</annotation>
```

1차 시도: Fast RCNN 적용 → Detection

1차

- Faster R-CNN 적용 테스트 성공



1차 시도: YOLO 적용 → Detection

1차

- YOLO 적용 테스트 실패

1차 시도: 실패 원인 파악

- 데이터 부족을 의심함
- Fast RCNN 모델에서 인식을 성공하였으므로 작업의 방향성은 맞는 것으로 간주함

2차 시도: 데이터 추가 + 레이블링 + 학습

- 중고차 매매 사이트를 통해 100개의 고화질 이미지 확보
- 새로 추가된 각 이미지에 대하여 레이블링 작업 수행
- 100개의 이미지를 이용하여 1,000번의 학습 수행

2차 시도: YOLO 적용 → Detection

- 번호판 인식은 되지만 성능이 열악함
- 인식 영역이 조금씩, 또는 꽤 어긋나거나 잘못 그려지는 경우 발생

2차 시도: 실패 원인 파악

- 데이터 부족이 원인일 가능성 제시
- 또는 학습 횟수의 부족이 원인일 가능성 제시
- 기존 모델(CNN)에서 사용하는 레이블링 데이터와 YOLO에서 사용하는 데이터의 형태가 다름을 확인

2차 시도: 실패 원인 파악

2차

- YOLO 적용 테스트 실패

LabelImg 용 XML

```
<annotation>
    <folder>CarNumber</folder>
    <filename>001.jpg</filename>
    <path>/training/001.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>980</width>
        <height>551</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>plate</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>330</xmin>
            <ymin>321</ymin>
            <xmax>744</xmax>
            <ymax>386</ymax>
        </bndbox>
    </object>
</annotation>
```

Yolo 용 TXT

```
0 0.611328 0.700521 0.400391 0.640625
```

- 다수의 사이트에서 변경없이 사용할 수 있는 것으로 설명하거나 관련 설명 자체가 누락됨

2차 시도: 실패 원인 파악

2차

- YOLO 적용 테스트 실패

LabelImg 용 XML

```
    </source>
    <size>
        <width>980</width>
        <height>551</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>plate</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>330</xmin>
            <ymin>321</ymin>
            <xmax>744</xmax>
            <ymax>386</ymax>
        </bndbox>
    </object>
</annotation>
```

Yolo 용 TXT

```
0 0.611328 0.700521 0.400391 0.640625
```

이미지 크기를 기준으로 하여
0~1 값으로 정규화, 파일 변환

3차 시도: 데이터 추가, 학습횟수 증가

- 1,675개의 이미지를 확보하여 2,000번의 학습 수행
- 기존의 이미지 레이블링 데이터 → YOLO 용 데이터로 변환

3차 시도: YOLO 적용 → Detection

2차

- YOLO 적용 테스트 실패

변환된 데이터는
제대로 적용되지 않음

3차 시도: 실패 원인 파악

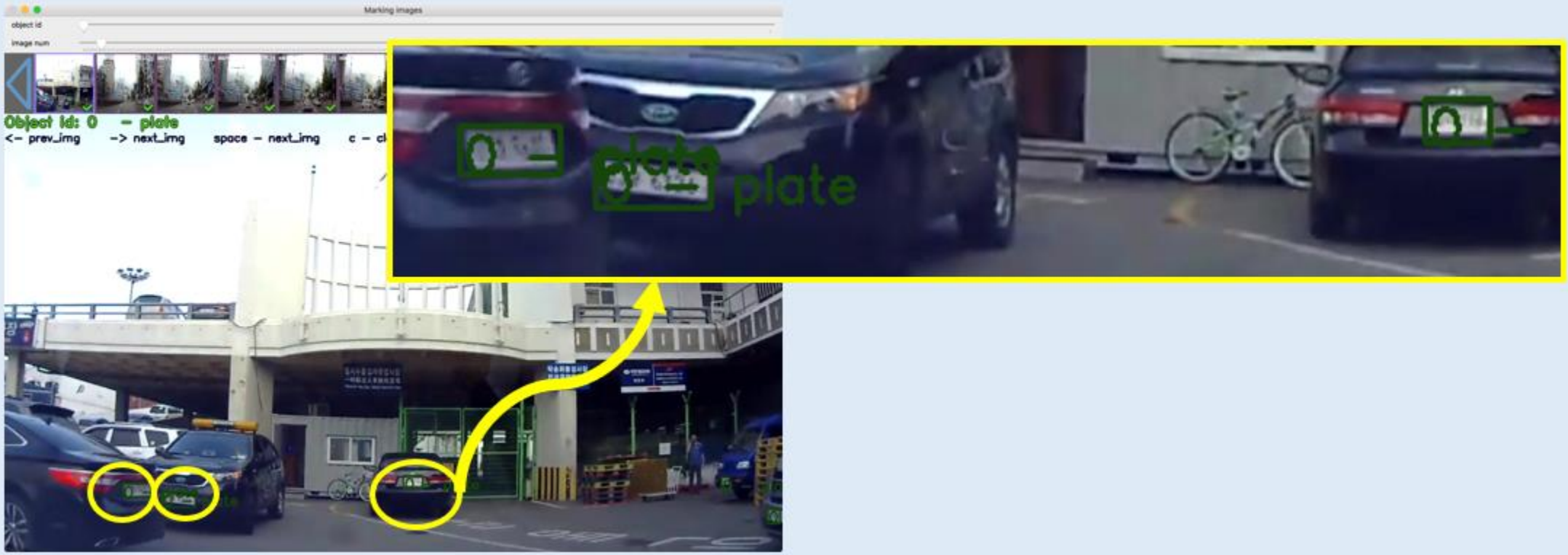
- 기존 레이블링 데이터를 YOLO 형태로 변환한 것이 제대로 동작하지 않았을 가능성 제시
- 레이블링 작업을 처음부터 새로 하기로 결정
- 인터넷 검색을 통하여 YOLO 전용 레이블링 도구 검색

3차 시도: 레이블링 재작업

3차

- 레이블링 도구: Yolo-Mark

인터넷 검색을 통해 새로운 도구 발견



3차 시도: YOLO 적용 → Detection

3차

- YOLO 기반 차량번호판 인식 성공

3차 시도: YOLO 적용 → 동영상

실패

3차 시도: 실패 내용

- 1,675개의 이미지를 이용하여 2,000번의 학습 수행
  - 임계 값: 80% → 동영상에 포함된 차량번호판의 약 10%정도 인식
  - 차량번호판이 아닌 유사 패턴 발견 시 → 차량번호판으로 오인식
  - 인식 성공률은 낮지만 대체로 안정적 (꾸준히 10%대…)
  - 정지영상(이미지)에 대한 인식 성공률은 약 90% → 안정적
  - 증가한 학습대상 이미지에 비하여 지나치게 적은 학습 횟수가 원인일 수 있다고 판단 → 학습 횟수를 늘리기로 결정

4차 시도: 학습 횟수 증가

- 1,675개의 이미지를 이용하여 45,000번의 학습 수행
  - 임계값: 80% → 정지영상 인식 성공 (성공률 약 90%), 동영상 인식 실패
  - 임계값: 30% → 동영상에서 일부 차량번호판 인식 성공
  - 인식 기대값이 너무 낮게 계산됨 → 처음부터 인식을 시도하지 않음

  - 너무 선명한 이미지만을 학습대상으로 사용한 것이 원인일 가능성 제시

    → 동영상에서 캡처한 이미지를 일부 학습에 적용하기로 결정

4차 시도: 데이터 추가 수집(동영상에서)

- 2,013개의 이미지를 이용하여 2,000번의 학습 수행
  - 동영상에서 캡처한 338장의 이미지 대상 → 레이블링 작업 추가 진행
  - 임계값: 50% → 인식 실패
  - 임계값: 30% →
    - **시도(4)보다 인식 수량 증가 확인**
    - 중복 인식 증가
    - **유사패턴**에 대한 오인식 증가
    - **어긋난 위치**에 대한 인식 표시 증가

학습 횟수의 증가가 필요하다고 판단

4차 시도: 학습 횟수 증가

- 2,013개의 이미지를 이용하여 45,000번의 학습 수행
  - 임계값: 60% → 동영상에서도 왠만한 경우는 인식 성공(기대값 평균 65%~86%)
  - 차량번호판이 아닌 유사 패턴 발견 시 → 오인식 거의 없음
  - 거의 정확한 번호판의 위치 표시

정지영상에 대한 인식 결과

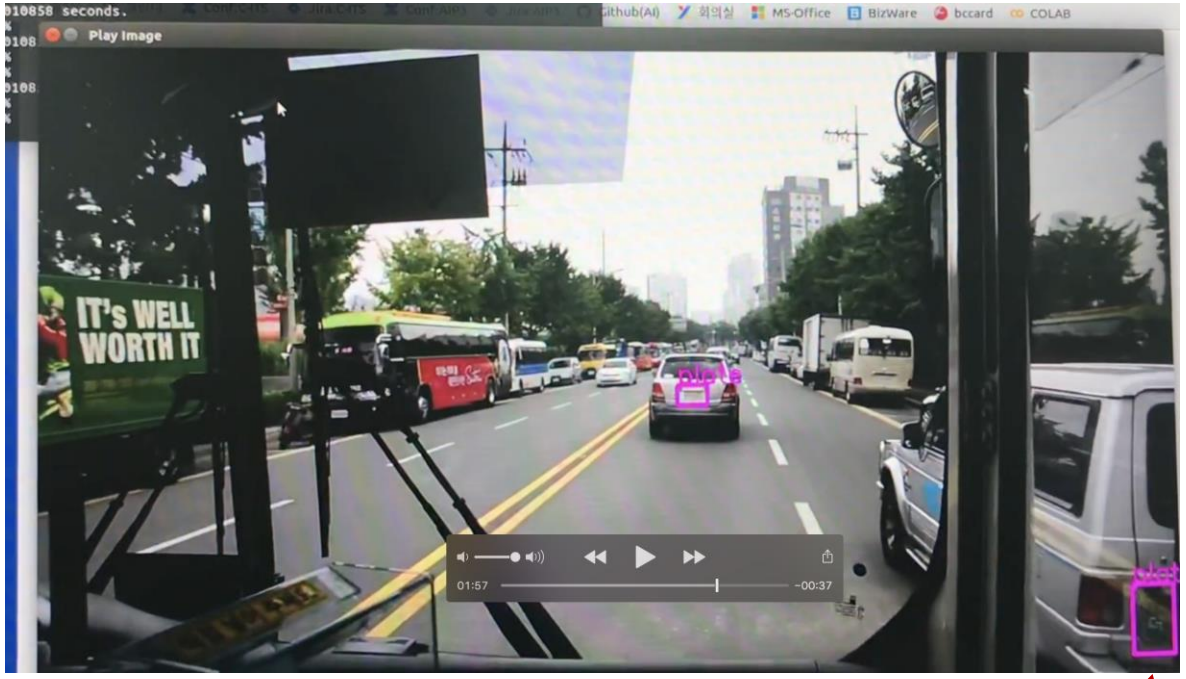
(92%)

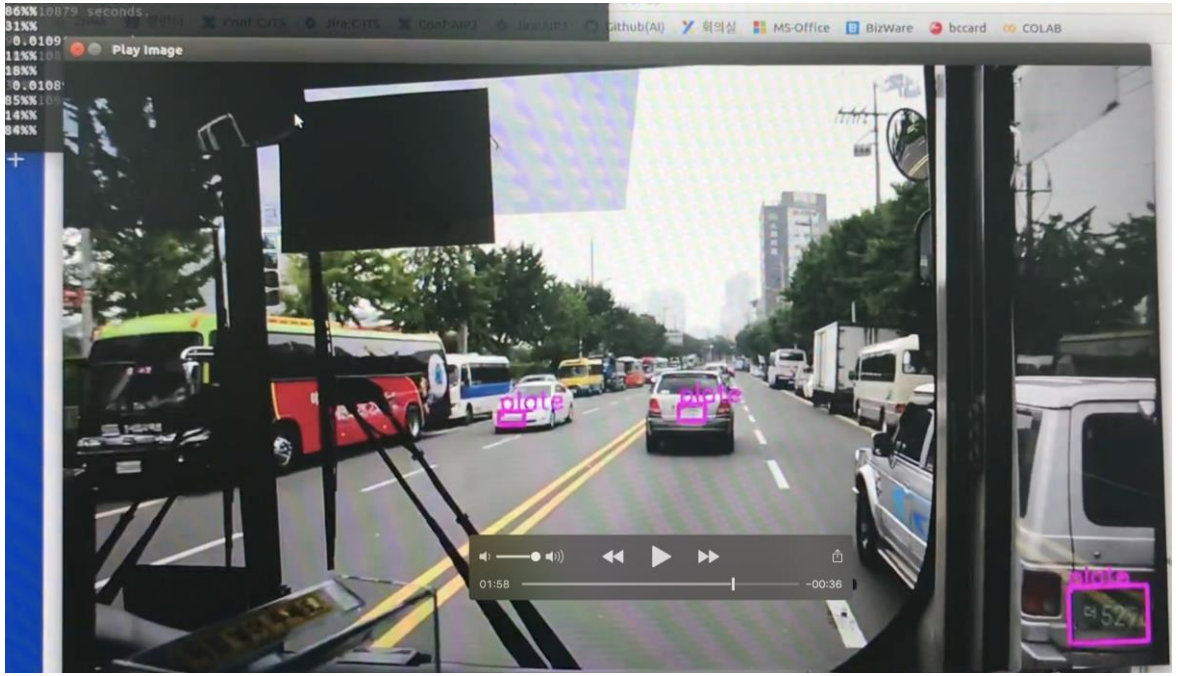
(90%)

(89%. 81%)

(90%)

동영상에 대한 인식 결과





- 일부만 표시된 차량번호판도 인식 성공

결론: Labeling 작업에 대한 사전 고려사항

- **Labeling 영역의 지정에 대한 규칙 필요**
  - 인식하고자 하는 대상과의 거리
  - 시선의 초첨으로부터 허용되는 좌우 범위(각도)
  - 시선의 초점으로부터 허용되는 상하 범위(각도)
  - 인식 대상에 대한 선명도(흐릿함)의 허용 범위
  - 인식 대상에 적용되는 밝기의 허용 범위
  - 인식 대상의 크기는 다양하게 적용 가능하도록 이미지 선택

- 정확한 수치 규정일 필요는 없음
- 대략적인 느낌을 통해 적용

결론: Labeling 작업에 대한 사전 고려사항

- 예 1

결론: Labeling 작업에 대한 사전 고려사항

- 예 2

결론: Labeling 작업에 대한 사전 고려사항

- 예 3

결론: Labeling 작업에 대한 사전 고려사항

- 예 4

이런 이미지만 학습하면

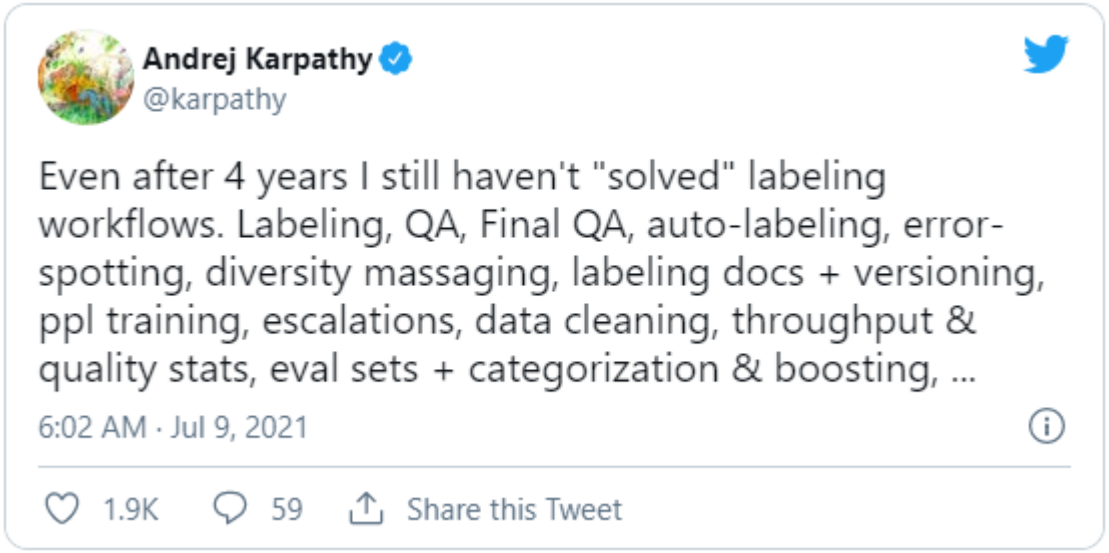
이런 크기의 대상은 인식하지 못하는 경우 증가

결론: Labeling 작업에 대한 사전 고려사항

- 영상의 전처리 범위에 대한 규칙 필요
  - 입력 영상의 주변 환경 (실내, 실외, 맑음, 흐림 등)을 기준으로 입력 영상에 대한 표준 밝기
  - 입력 영상의 해상도를 기준으로 하는 이미지의 표준 크기
  - 그 외, 특정 환경에서의 인식이 필요한 경우, 해당 환경에 맞는 밝기, 크기 등의 사전 규칙 결정

테슬라 AI 디렉터, 안드레아 카파시

- **컴퓨터 비전 데이터에 대한 모든 것 (superb-ai.com)**



4년이 지났는데도 레이블링 작업흐름이 아직 "해결"되지 않았습니다. 레이블링, QA, 최종 QA, 자동 레이블링, 오류 발견, 다양성 마사지, 레이블링 문서 + 버전 지정, PPL 교육, 에스컬레이션, 데이터 정리, 처리량 & 품질 통계, 평가 세트 + 분류 & 부스팅, ...

- 이미지나 영상 내의 객체를 레이블링 하는 것이 레이블링 작업의 전부라고 단순히 생각하기 쉽다.
- 그러나 안드레아가 나열한 항목들을 보면 레이블링 워크플로우(labeling workflows)는 그보다 더 많은 작업을 수반하고 있음을 알 수 있다

레이블링 워크플로우

- 레이블링 워크플로우를 구성하는 작업들
  - 데이터에 대한 레이블링 유형 기준 마련
  - 레이블링 작업 결과물에 대한 품질관리(QA), 피드백 수집 과정 구축
  - 레이블링 작업자의 훈련과 퍼포먼스 측정 등을 포함하는 인력관리
  - 레이블링 과정 중 이슈 발생시 커뮤니케이션 비용 관리
  - 라벨링을 통해 구축된 데이터셋의 버전관리 등

→ 나열된 것은 레이블링 작업 중 예상되는 워크플로우 범위 중 일부일 뿐...

레이블링 작업의 중요성

- 이미지, 영상 데이터의 레이블링 작업은 가장 덜 중요한 것으로 인식
- 그러나 컴퓨터 비전 문제를 푸는 데 있어서 가장 선도적인 테슬라팀이 여전히 난항을 겪고 있는 부분이 라벨링 워크플로우라고 지적
- 상용화 단계에서 컴퓨터 비전 문제를 해결하기 위해 우리가 주목해야할 부분은, **사실은 데이터**였다.

컴퓨터 비전이 데이터를 다루기 어려운 이유

- **사람은 이미지를 보지만 컴퓨터는 숫자를 본다**
  - **저명한 컴퓨터 비전 전문가, UC버클리 교수 Jitendra Malik(지텐드라 말릭)**
    - 사람이 이미지를 인식하는 과정이 무의식적으로 혹은 잠재 의식적으로 진행되기 때문에 흔히 컴퓨터가 비전을 처리하는 방식도 매우 쉬울 거라고 착각할 수 있다고 지적
    - 실제 사람의 경우에도, 대뇌피질의 상당 부분은 이미지 프로세싱 처리에 집중
    - 컴퓨터가 이미지를 처리하는 방식은 인간과 다름
    - 인간이 매우 시각적이고 직관적인 방식으로 이미지를 인식하는 반면, 컴퓨터는 이미지의 모든 부분을 개별 픽셀로 환산하여 숫자로 인식
    - 이미지를 인식할 때마다 처리해야 하는 데이터 양이 많다는 것을 의미
      → 복잡한 시각적 작업을 수행하기 위한 계산 및 데이터 리소스 또한 방대해짐을 의미

컴퓨터 비전이 데이터를 다루기 어려운 이유

- 사람보다 더 많은 데이터를 필요로 하는 컴퓨터 비전 시스템
  - 현재 컴퓨터 비전 분야를 이끌어 가는 가장 고도화된 기법은 지도학습
  - 지도학습은 '정답'이라고 여겨지는 피쳐를 레이블링한 데이터를 기반으로 학습을 이행
  - 학습에 사용된 데이터를 기반으로 모델의 성능이 결정되기 때문에 모델이 인지하려는 모든 다양한 상황에 대해 데이터로 준비해서 학습시켜야 하는 것