# 241213 CV

- 우리가 했던 분류
  - Classification
  - 공간 개념 존재하지 않음

▼ Object Detection : 객체 탐지

- 내가 이미지에서 라벨링한 구역 == 실측값
- 모델이 예측한 구역 == 예측값
- 이 두개의 IOU값
  - IOU = 예측값 / 실측값
- 종류:
  - R-CNN
    1. 이미지 입력
    2. RoI (Region of Interest) : Selective Search 알고리즘 사용, 후보 영역 추출
       a. **시간 소모 多**
    3. RoI 동일 사이즈로 Resizing - CNN에 입력 가능하도록 조정
    4. 각 독립적 ConvNet 통과 (질감, edge 등 특징을 추출)
    5. SVM 분류 통한 클래스 예측
       a. Support Vector Machine
       b. 각 RoI의 객체 클래스를 예측함
    6. Bounding-Box Regressor (새로운 물체 위치 좌표값)
       a. 바운딩 박스의 중심위치, 너비, 높이 조정
  - Fast R-CNN
    1. 이미지 입력
    2. 한 번의 ConvNet 연산 → Feature map 출력
       a. 이미지 전체를 한 번만 CNN에 통과

    b. 연산 비용 ↓
3. RoI
4. RoI Pooling
    a. RoI Pooling: RoI(관심영역)를 일정한 그리드로 나눈 후 Max Pooling
5. FC Layer (Fully Connected Layer)

- One Stage vs Two Stage
    - Two Stage
        - R-CNN 계열: Region Proposal 단계 & CNN 예측
        - 정밀도, 재현율 ↑ / 속도 ↓
    - One Stage
        - YOLO: 이미지 전체 한 번의 CNN 연산, Bounding Box 좌표값/클래스 확률/신뢰도 동시 예측
        - 속도 ↑ / 정확도 ↓ 위험 (작은 객체, 복잡한 장면...)
            - RoI 빠져서 빠른 속도, but, 물체 작다하면 R-CNN 계열 유리. → 정밀도
- YOLO
    1. 이미지 입력, 그리드로 나눠 탐지 ← 해당 셀의 중심에 있는 객체를 탐지함
    2. 한 번의 CNN 연산, Bounding Box 좌표값, 클래스 확률, 신뢰도 동시 예측
    3. 손실 함수
        a. 바운딩 박스 손실, 클래스 확률 손실, 신뢰도 손실 요소를 고려한 손실 함수
        b. 손실 함수로 모델 학습
    4. 참고: YOLO: Real-Time Object Detection
- Object Detection 관련 링크들
    - roboflow: Roboflow
    - labelimg 깃허브: HumanSignal/labelImg
    - models 깃허브: tensorflow/models

        - detectron2 깃허브: facebookresearch/detectron2
        - HUMAN 깃허브링크: 로컬에서 라벨링? 한 번 만져봐라

▼ Image Segmentation : 물체의 실루엣을 만들고 시각화

- Semantic Segmentation : 픽셀 단위로 간소화
    - 단순 pixel 변화에 따른 것
    - 사람인지 고양인지 하늘인지 등등...
- Instance Segmentation : 인스턴스 세분화
    - 인간으로 인식한 객체 중에서도 인간1, 인간2... 로 따로 봄
- Mask R-CNN ← 인스턴스 세분화
    1. 이미지 입력
    2. CNN : 특징맵으로 변환
    3. RPN : 물체가 있을 가능성이 높은 영역(Region Proposal) 생성
        a. Faster R-CNN 에서 도입된 RPN을 사용.
    4. Region Proposal (물체 후보 영역 좌표 & 신뢰도 출력 값)
    5. RoI Pooling
    - FCC Layer 최종출력: 물체 클래스, Bounding Box 좌표값, 마스크(픽셀단위)
        - RPN: selective Search의 진보된 버전
- Pose Estimation
    1. 이미지 입력
    2. 관절 위치 detection(픽셀 단위의 세부 단위)
        - 이미지 특정 영역에서 관절이 존재할 확률 계산
    3. Heatmap 추출 (어깨, 무릎 위치 등)
        - → 해당 관절 위치 가능성 높을수록 밝은 색
    4. 각 관절 정보 결합 (스켈레톤 구조 형성)
    5. 회귀 Network 거쳐서 관절의 정확한 위치 추출
        - → Detection 결과를 Regression이 정밀하게 조정
    - 대표적 Pose Estimation 오픈소스 라이브러리:
        - OpenPose

        - MediaPipe
        - DensePose

- 지금까지의 CNN 계층 이해
    - Input_data = 32×32×3
        - 3×3 필터 수 = 64
            - 연산량: Input_data × (3×3) × 64
        - 5×5 필터 수 = 32
            - 연산량: Input_data × (5×5) × 32
        - 1×1 필터 수 = 16
            - 연산량: Input_data × (1×1) × 16
        - + 1×1 적용 후 n×n conv 연산량: 32×32 × 16 × (n×n) × {n×n 필터 수}

- Vision Transformer (ViT)
    1. 이미지 입력
    2. 작은 조각들(Patch)로 쪼개기
    3. 선형 변환(동일 차원의 벡터)
        - 각 패치를 Flatten하여 벡터로 변환, 고정된 차원의 벡터로 매핑
    4. 각각의 위치정보 추가 & Token 부여
    5. Transformer Encoder ← 패치 벡터와 클래스 토큰을 입력
    6. Class token(최종분류 정보 학습) → MLP Head(분류작업 수행)
    7. 최종 클래스 예측

---

과제:

1. 수업 내용 정리
2. 분류 모델 만들기
    a. 이미지 데이터셋 구축
    b. 관절 추출 (분류 정확도 최대한 높일 수 있는 좌표값)

c. CSV 파일 추출

d. 머신러닝 분류기 사용 후 정확도까지 볼 것

e. !!! 모든 좌표는 쓰지 말 것 !!!

3. 제출 후 PR 코멘트에 자신이 제출한 코드에 대한 설명을 남길 것

a. 데이터셋은 무엇인지?

b. 추출한 좌표는?

c. 이 좌표를 추출한 이유는?