2025-11-25(End-to-End Object Detection with Transformer)

End-to-End Object Detection with Transformer:
https://arxiv.org/abs/2005.12872

https://github.com/facebookresearch/detr

End-to-End Object Detection with Transformers 논문 리뷰_김해찬.pdf

DETR 논문 리뷰 발표 참고

0. DETR

DETR는 “이미지 안의 모든 물체를 한 번에 집합(set) 형태로 예측하는 객체 탐지 모델”이다.

- Anchor, proposal, NMS 같은 복잡한 설계 없이
- Transformer encoder–decoder 구조와
- Hungarian matching(이분 매칭)을 이용한 set loss만으로
- End-to-end 학습이 가능하도록 만든 모델이다.

핵심 요약:

객체 탐지를 set prediction 문제로 정의하고, Transformer를 사용하여 end-to-end로 학습 가능한 최초 계열의 모델 중 하나이다.

---

1. 배경: 기존 객체 탐지의 한계

1-1. 기존 딥러닝 기반 Detector의 전형 구조

Faster R-CNN, RetinaNet, FCOS 등 대부분의 구조는 다음 흐름을 따른다.

1. CNN backbone으로 feature map 추출
2. Feature map 위에서
    - Anchor box를 촘촘히 배치하거나
    - Region proposal(후보 박스)을 생성
3. 각 anchor/proposal에 대해
    - Class 분류 (classification)
    - Bounding box 보정 (regression)
4. 마지막에 NMS(Non-Maximum Suppression)를 이용해 겹치는 박스를 제거

여기에는 사람이 설계해야 하는 요소가 많다.

- Anchor 크기 및 비율
- IoU 기준, positive/negative 매칭 규칙
- NMS 임계값
- FPN 구조, proposal 개수, sampling 비율 등

1-2. 논문이 지적하는 문제점

- 파이프라인이 복잡하고, 여러 단계로 분리되어 있음
- NMS 등 후처리가 필수
- “정답 박스들의 집합”을 end-to-end로 직접 예측하지 못함
- 사람이 정의한 다양한 heuristic(prior)에 크게 의존함

DETR의 기본 질문은 다음과 같다.
“모델이 애초에 ‘박스들의 집합’을 직접 예측하도록 만들면
anchor 설계나 NMS와 같은 복잡한 요소를 제거할 수 있는가?”

---

2. DETR의 핵심 아이디어 정리

DETR의 핵심 개념은 다음 세 가지로 요약된다.

1. Set prediction formulation
    - 출력 개수를 N개로 고정한다.

- 각 출력 슬롯은 다음 중 하나를 예측한다.
  - (클래스, 박스)
  - 또는 “no object (ø)”
- 전체 출력은 “순서 없는 집합(set)”으로 취급된다.

2. Hungarian matching + set-based loss
   - 예측 set과 GT set 사이에 1:1 매칭을 찾는다.
   - 그 매칭을 기준으로 classification + box loss를 계산한다.
   - 중복 박스 예측이 자연스럽게 패널티를 받아 NMS가 필요 없어진다.
3. Transformer encoder–decoder + object query
   - Encoder: 이미지 전체의 global context를 self-attention으로 학습한다.
   - Decoder: 학습 가능한 N개의 object query를 입력으로 받아,
     각 query가 이미지 내 하나의 객체 슬롯 역할을 수행한다.

---

3. Set prediction과 Hungarian loss

3-1. 출력 형식과 “no object” 클래스

한 이미지에 대해 DETR는 N개의 슬롯을 출력한다.

- 예측 세트:

$$\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$$

  각 원소는 다음과 같다.

$$\hat{y}_i = (\hat{p}_i(c), \hat{b}_i)$$

  여기서

$$\hat{p}_i(c) : 클래스확률분포(softmax결과, 모든클래스에대한확률)$$

$$\hat{b}_i \in [0,1]^4 : 정규화된 boundingbox(예 : (c_x, c_y, w, h))$$

- 정답 세트:

$$y = \{y_i\}_{i=1}^{N}$$

각 원소는 다음과 같다.

$$y_i = (c_i, b_i)$$

- 실제 객체 개수는 이미지마다 다르므로

  나머지 슬롯은 “no object(∅)” 클래스로 채운다.

직관적으로는 다음과 같다.

- N칸짜리 detection 슬롯을 미리 만들어 두고,
- 각 슬롯이 “어떤 물체를 담당할지, 혹은 비어 있을지(∅)”를 학습을 통해 결정한다.

3-2. 1단계: Hungarian matching (이분 매칭)

$$예측 set \hat{y} 와 GT set y 가 있을 때,$$

두 집합 사이의 “최적 1:1 매칭”을 먼저 계산한다.

$$순열 \sigma 는 1, \ldots, N 인덱스의 재배열이다.$$

$$매칭 비용을 최소화하는 \hat{\sigma} 는 다음과 같이 정의된다.$$

$$\hat{\sigma} = \arg\min_{\sigma \in S_N} \sum_{i=1}^{N} L_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$$

$$여기서 S_N 은 길이 N 인 모든 순열의 집합이다.$$

이 최적화 문제는 Hungarian algorithm으로 해결하며,

$$시간복잡도는 O(N^3) 수준이고, N 을 약 100 정도로 두면 실용적인 수준이다.$$

3-3. 매칭 비용 L_match의 정의

정답 하나와 예측 하나를 다음과 같이 둔다.

- 정답:

$$y_i = (c_i, b_i)$$

- 예측:

$$\hat{y}_j = (\hat{p}_j, \hat{b}_j)$$

Hungarian 매칭에서 사용하는 비용 함수는 다음과 같이 정의된다.

$$L_{\text{match}}(y_i, \hat{y}_j) = -1_{\{c_i \neq \varnothing\}}\, \hat{p}_j(c_i) + 1_{\{c_i \neq \varnothing\}}\, L_{\text{box}}(b_i, \hat{b}_j)$$

설명:

c_i가실제객체($\emptyset$가아닌클래스)일때만비용을계산한다.

해당클래스를높게예측할수록($\hat{p}_j(c_i)$가클수록)비용이작아진다.

박스손실$L_{\text{box}}(b_i, \hat{b}_j)$가작을수록비용이작아진다.

$c_i = \emptyset$인경우(빈슬롯)는대체로일정한비용으로처리하여

매칭에 크게 영향을 주지 않도록 설계한다.

또한매칭단계에서는$cross-entropy$의 $-\log$ 대신

확률값$\hat{p}_j(c_i)$자체를사용하여

박스 손실과 스케일을 맞춘다.

3-4. 2단계: Hungarian loss (최종 학습 손실)

$Hungarian algorithm$으로최적매칭$\hat{\sigma}$를구한뒤,

각쌍$(y_i, \hat{y}_{\hat{\sigma}(i)})$에대해최종$loss$를계산한다.

전체 Hungarian loss는 다음과 같다.

$$L_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \Big[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + 1_{\{c_i \neq \varnothing\}}\, L_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \Big]$$

구성 요소:

- 첫 번째 항: 분류 손실 (classification cross-entropy)
- 두 번째 항: 박스 회귀 손실 (bounding box regression loss)

실제 구현에서는 배경 클래스(∅)에 대한 분류 손실에 작은 가중치(예: 0.1)를 부여하여
배경 샘플이 매우 많은 상황에서의 class imbalance를 완화한다.

3-5. 박스 손실 L_box: L1 + Generalized IoU

DETR는 anchor offset을 예측하는 대신,

이미지 크기에 정규화된 "절대 bounding box"를 직접 예측한다.

박스 손실은 L1 손실과 Generalized IoU(GIoU) 손실의 조합으로 정의된다.

$$L_{\text{box}}(b_i, \hat{b}_j) = \lambda_{\text{iou}} L_{\text{iou}}(b_i, \hat{b}_j) + \lambda_{L1} \|b_i - \hat{b}_j\|_1$$

$$L_{\text{iou}} : Generalized IoU loss$$

$$\|\cdot\|_1 : L1 norm$$

$$\lambda_{\text{iou}}, \lambda_{L1} : \text{두 손실항의 가중치}$$

역할:

- GIoU 항은 두 박스의 겹침 정도를 직접 반영하며, 크기에 무관한(scale-invariant) 특성을 가진다.
- L1 항은 박스 중심 및 크기 좌표의 차이를 직접 줄이는 데 기여한다.

---

4. DETR 전체 아키텍처 구조

DETR의 전체 구조는 다음과 같이 요약된다.

1. CNN backbone (ResNet-50 또는 ResNet-101)
2. Transformer encoder
3. Transformer decoder + N개의 object query
4. 각 query에 FFN(head)을 붙여 class + box를 출력
5. 전체 출력에 대해 Hungarian loss를 적용하여 end-to-end 학습

4-1. Backbone (ResNet)

입력 이미지와 feature map은 다음과 같이 표현할 수 있다.

- 입력 이미지:

$$x \in \mathbb{R}^{3 \times H_0 \times W_0}$$

- 출력 feature map:

$$f \in \mathbb{R}^{C \times H \times W}$$

여기서

$$\text{일반적으로} C \approx 2048 (ResNet C5 stage \text{출력})$$

$$H = H_0/32, W = W_0/32 (stride 32)$$

추가 변형인 DC5("dilated C5")에서는

마지막 stage에 dilation을 적용하여 해상도를 2배로 증가시키고,

작은 객체에 대한 성능을 향상시킨다.

4-2. Transformer encoder

Encoder는 CNN feature를 sequence로 변환하고, self-attention을 수행한다.

$$1 \times 1 convolution \text{으로 채널수를} C \text{에서} d (Transformer hidden dim) \text{로 축소}$$

$$\text{공간차원} (H, W) \text{을} flatten \text{하여 길이} HW \text{인 시퀀스로 변환}$$

$$\text{각 위치에} 2D positional encoding \text{을 더하여 위치정보를 포함}$$

$$\text{여러개의} encoder layer \text{를} stack$$

- Multi-head self-attention
- Position-wise FFN
- Residual connection + LayerNorm

역할:

- 이미지 전체의 long-range dependency를 학습한다.
- 같은 객체에 속하는 부분들은 서로 유사하게,
  다른 객체는 분리되도록 feature를 형성한다.

4-3. Transformer decoder 및 object query

Decoder의 입력은 길이 N인 query 시퀀스이다.

- 각 query는 학습 가능한 positional embedding이며,
  "객체 슬롯(object slot)"을 의미한다.

각 decoder layer는 다음 연산을 수행한다.

1. Self-attention (query 간 상호작용)
   - 서로 같은 객체를 중복 예측하지 않도록 조정하는 역할을 수행한다.
2. Encoder–decoder attention
   - 각 query가 encoder 출력 전체를 보고
     자신이 담당할 객체 위치 주변의 정보를 집중적으로 추출한다.
3. FFN + residual connection + LayerNorm

특징:

- RNN처럼 순차적으로 한 개씩 예측하지 않고,
  N개의 query를 병렬로 동시에 업데이트한다.
- 최종 loss가 permutation-invariant한 set loss이므로,
  query의 순서는 의미를 갖지 않는다.

4-4. Prediction head (FFN)

각 decoder 출력 embedding에 대해 다음과 같은 head를 적용한다.

- 3-layer MLP → bounding box 좌표 (4차원)
- 1-layer linear + softmax → class distribution (K+1 차원; K개 클래스 + ∅)

결과적으로 각 query는 하나의 `(클래스, 박스)` 혹은 "no object(∅)"를 예측한다.

4-5. Auxiliary loss (보조 손실)

- 마지막 decoder layer뿐 아니라
  모든 decoder layer 출력에 head를 붙이고,
  각 layer별로 Hungarian loss를 추가하여 학습한다.

목적:

- 앞쪽 layer에서도 대략적인 detection을 수행하도록 유도하여
  gradient 전달을 돕는다.

- layer가 깊어질수록 예측이 점점 refine되는 구조를 형성한다.

---

5. 학습 설정과 COCO 실험

5-1. 학습 세팅 (COCO 2017 기준)

- Dataset: COCO 2017 (train 118k, val 5k)
- Optimizer: AdamW

$$Transformerlearningrate : 1e-4$$

$$Backbonelearningrate : 1e-5$$

$$Weightdecay : 1e-4$$

- Backbone: ResNet-50 / ResNet-101 (ImageNet 사전학습, BN freeze)
- Data augmentation:
  - 짧은 변: `[480, 800]` 범위에서 random resize
  - 긴 변: 최대 1333
  - 0.5 확률로 random crop 후 다시 resize
- Dropout: Transformer에 0.1 적용
- 기본 학습 스케줄:
  - 300 epochs, 200 epoch 이후 learning rate 1/10 감소
- 비교용 긴 스케줄:
  - 500 epochs, 400 epoch 이후 learning rate 1/10 감소
  - 약 +1.5 AP 향상

5-2. 추론 시 처리

- N개의 슬롯 중 “no object(ø)” 클래스로 예측된 것은 버린다.
- 배경(ø) 예측이 많기 때문에,

  AP 기준으로는 “ø 예측을 두 번째로 높은 클래스 라벨로 치환하는”

  후처리 트릭을 사용하면 약 2 AP 정도 향상이 보고된다.

핵심:

- NMS를 전혀 사용하지 않는 구조라는 점이 기존 detector와의 큰 차이이다.

5-3. Faster R-CNN과의 성능 비교 (개략)

COCO validation 기준 대략적인 비교 결과는 다음과 같다.

- DETR (ResNet-50)
  - AP ≈ 42
  - 큰 객체(AP_L): Faster R-CNN보다 상당히 우수
  - 작은 객체(AP_S): 다소 성능 열세
- DETR-DC5 (해상도 2배)
  - AP ≈ 43.3
  - 작은 객체 성능이 일부 개선

요약:

- “잘 튜닝된 Faster R-CNN + FPN”과 유사한 수준의 AP를 보이면서도
- 구조는 훨씬 단순하고,
- 특히 큰 객체에 대한 성능이 강점이다.

---

6. Ablation Study에서의 주요 관찰

6-1. Encoder layer 수 변화

Encoder layer 개수를 0, 3, 6, 12 등으로 변경하여 실험한 결과:

- 0 layer: AP가 크게 감소, 특히 large object 성능이 크게 떨어짐
- 6 layer: 기본 설정, 성능이 충분히 양호
- 12 layer: 약간의 추가 향상

해석:

- 전역 self-attention을 통한 long-range dependency 학습이
  객체 분리(disentangling)에 중요한 역할을 한다.

6-2. Decoder layer와 NMS 효과

각 decoder layer의 출력에 대해 AP와 NMS 효과를 비교하면 다음과 같다.

- 초반 layer:
  - 동일 객체에 대한 중복 예측이 많이 발생
  - NMS 적용 시 AP가 오히려 증가하는 경향
- 후반 layer:
  - query 간 self-attention을 통해 중복이 점차 조정됨
  - 마지막 layer에서는 NMS 적용 시
    true positive가 제거되어 AP가 감소하는 현상이 나타난다.

결론:

- 충분한 깊이의 decoder와 set loss 조합만으로
  NMS 없이도 중복 제거가 가능하다.

6-3. FFN의 역할

Transformer layer에서 FFN을 제거하고 attention만 사용할 경우:

- 전체 파라미터 수는 감소하나
- AP가 약 2~3 정도 감소

이는 multi-head attention만으로는 표현력이 부족하며,
FFN이 비선형 변환을 통해 중요한 기능을 수행함을 시사한다.

6-4. Positional encoding의 영향

Positional encoding 구성에 대한 실험 결과:

- Spatial positional encoding을 완전히 제거하면 AP가 크게 감소
- Encoder에서만 제거, Decoder에서만 사용하는 등의 조합에 따라
  성능 차이가 부분적으로 존재
- Sine 기반 positional encoding과 학습 가능한 positional encoding 모두
  유사한 성능을 보이며, 세부 차이는 크지 않다.

핵심:

- Object query(출력 쪽 positional encoding)는 필수적이다.
- Spatial positional encoding을 사용하는 편이 전반적인 성능 향상에 중요하다.

6-5. Loss 구성: L1 vs GIoU

박스 손실 조합에 따른 결과는 다음과 같다.

- Class + GIoU만 사용: baseline과 비슷한 AP
- Class + L1만 사용: AP가 눈에 띄게 감소
- Class + L1 + GIoU (기본 설정): 가장 좋은 성능

GIoU 손실이 핵심적인 기여를 하며,

L1 손실은 특히 중형 및 대형 객체에서 추가적인 이득을 제공한다.

---

7. Object query 동작 방식 분석

7-1. 슬롯별 박스 분포

각 query 슬롯이 예측한 박스 중심을 COCO 전체에서 시각화하면 다음과 같은 경향이 나타난다.

- 일부 슬롯은 화면 중앙에 주로 분포
- 일부 슬롯은 상단 또는 하단 등 특정 공간 영역에 편중
- 많은 슬롯이 "전체를 덮는 큰 박스 모드"를 함께 가지고 있음

  (COCO 데이터 특성상 큰 배경·큰 객체가 많기 때문으로 해석 가능)

7-2. "보기 드문 경우"에 대한 generalization

예시:

- 학습 데이터에는 "기린 13마리 이상이 있는 이미지"가 존재하지 않음
- 그러나 합성 이미지로 기린 24마리가 있는 입력을 제공하면

  DETR가 24개 개체를 상당히 잘 찾아낸 사례가 보고됨

이는 특정 query가 특정 클래스(예: "기린 전용 슬롯")로 고정되는 것이 아니라,

이미지 상황에 따라 query들이 유연하게 역할을 분담한다는 점을 보여준다.

---

8. Panoptic segmentation 확장

8-1. Panoptic segmentation 정의

Panoptic segmentation은 다음을 동시에 예측하는 문제이다.

- Things: 사람, 자동차 등 개별 인스턴스를 갖는 객체

- Stuff: 하늘, 도로, 잔디 등 연속적인 배경 영역

DETR는 이미 "모든 객체를 set으로 예측"하고 있으므로,

각 객체 슬롯마다 마스크(mask)를 예측하도록 head를 추가하면

동일한 프레임워크로 panoptic segmentation을 수행할 수 있다.

8-2. Mask head 구조

Mask head의 개략적인 구조는 다음과 같다.

- 입력:
  - 각 객체에 대한 decoder output embedding
  - Encoder feature map (multi-scale, FPN 스타일 구성)
- 절차:
  1. Multi-head attention
     - 객체 embedding을 query로 사용
     - encoder feature를 key/value로 사용
     - 객체별 low-resolution mask feature를 산출
  2. FPN-style CNN을 통한 upsampling
     - ResNet 여러 stage의 feature를 결합
     - 최종적으로 stride 4 해상도 수준의 mask를 예측

       (크기 예: `N × (H/4) × (W/4)`)
- Loss:
  - Dice(F1) loss
  - Focal loss (양/음 픽셀 불균형 문제 대응)

최종 panoptic 결과는 각 픽셀 위치에 대해

N개의 mask logits 중 argmax를 취하여

어느 객체(또는 배경)에 속하는지를 결정함으로써 얻어진다.

장점:

- Mask끼리의 겹침 문제를 자연스럽게 해결할 수 있고
- 별도의 복잡한 merging rule 없이 panoptic 출력을 구성할 수 있다.

8-3. 학습 방식과 성능

학습 방식의 한 예는 다음과 같다.

1. DETR를 bounding box 기반으로 먼저 학습한다.
2. Backbone과 Transformer를 freeze한다.
3. Mask head만 몇 epoch 추가로 학습한다.

위와 같은 2단계 학습 방식은 joint 학습보다 wall-clock time이 짧고,

성능도 유사하거나 더 좋은 것으로 보고된다.

성능 측면에서 DETR 기반 panoptic 모델은

- PanopticFPN, UPSNet 등 기존 SOTA와 경쟁 가능한 PQ 성능을 보이며
- 특히 stuff PQ에서 우수한 성능을 보이고
- things PQ 역시 준수한 수준을 유지한다.

---

9. Multi-head attention 수식 정리 (간단 버전)

DETR에서 사용하는 attention은 표준 Transformer의 multi-head attention 구조와 동일하다.

9-1. Single-head attention

입력:

- Query 행렬: `Q`
- Key 행렬: `K`
- Value 행렬: `V`

Single-head attention은 다음과 같이 정의된다.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

여기서 `d_k`는 각 head의 key 차원 크기이다.

9-2. Multi-head attention

Multi-head attention은 다음 단계로 구성된다.

1. Q, K, V를 여러 head로 분할하거나 서로 다른 projection을 적용한다.

2. 각 head에서 single-head attention을 계산한다.
3. 모든 head의 결과를 concat 한 뒤, 최종 linear projection을 적용한다.

Encoder에서는 self-attention (`Q = K = V`) 형태로 사용되며,

Decoder에서는

- Self-attention (query 간 관계 학습)
- Encoder–decoder attention (query와 encoder feature 간 상호작용)

두 종류의 attention을 모두 사용한다.

---

10. 전체 정리 및 논문 리뷰 포인트

DETR를 요약하면 다음과 같다.

1. 객체 탐지를 “set prediction 문제”로 재정의한다.
    - 출력 슬롯 수 N을 고정한다.
    - 각 슬롯이 (클래스, 박스) 또는 ∅를 예측하도록 한다.
2. Hungarian matching 기반 set loss를 사용한다.
    - 예측 set과 GT set 사이에 1:1 매칭을 강제한다.
    - 이 매칭을 기준으로 classification + box loss를 계산한다.
    - 이로 인해 중복 박스 예측이 억제되어 NMS가 불필요해진다.
3. Transformer encoder–decoder와 object query를 사용한다.
    - Encoder에서 global context를 학습한다.
    - Decoder의 N개 object query가 각각 한 객체 슬롯을 담당한다.
    - Self-attention을 통해 query 간 중복과 상호 관계를 조정한다.
4. 성능 측면에서
    - 튜닝된 Faster R-CNN + FPN과 비슷한 수준의 COCO AP를 달성한다.
    - 구조가 단순하며, 특히 큰 객체에 강한 성능을 보인다.
5. 확장성 측면에서
    - Mask head를 추가하여 panoptic segmentation으로 자연스럽게 확장할 수 있다.
    - 이후 Deformable DETR 등 다양한 후속 연구의 기반이 되었다.