

# End-to-End Object Detection with Transformers (DETR)

학회: Facebook Research 팀 ECCV 2020

발표자: 김해찬

---

# DETR 배경

## 기존 방법론 발전

- R-CNN, Fast R-CNN, Faster R-CNN으로 발전
  - 2-stage 탐지기: 영역 제안 후 분류
  - 1-stage 탐지기: YOLO, SSD 등장
    - 정확도와 속도 간 균형 추구

## Anchor 기반 접근법

- 다양한 크기와 비율의 anchor box 필요
- 수작업 설계에 의존하는 휴리스틱 방식
  - 하이퍼파라미터 튜닝 복잡성
  - 객체 크기/비율에 따른 성능 편차

## Post-processing

- Non-Maximum Suppression(NMS) 필수
  - 중복 탐지 제거를 위한 추가 단계
  - 속도 저하 및 파이프라인 복잡화
  - End-to-End 학습 불가능한 구조

## End-to-End 필요성

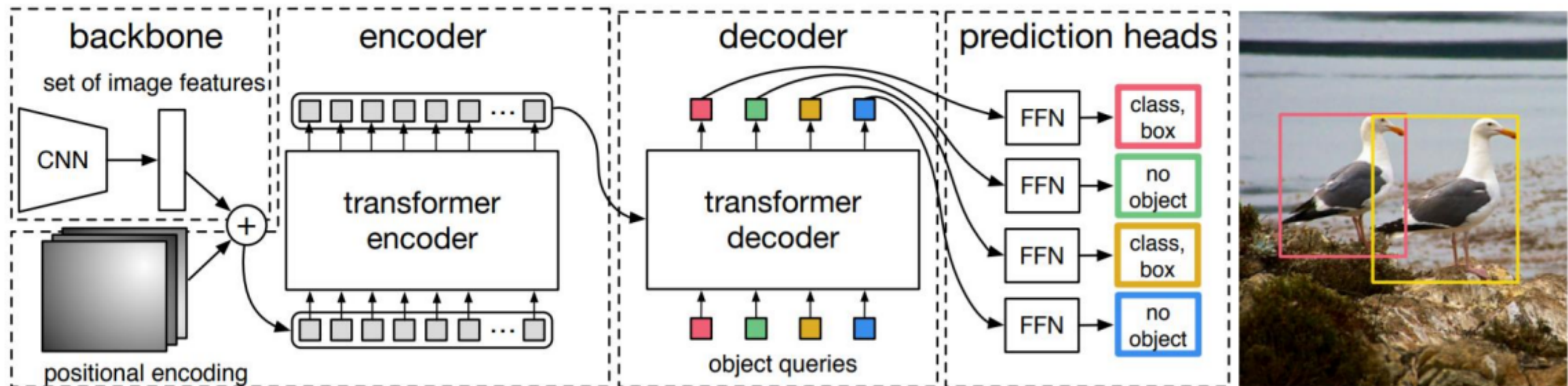
- 수작업 설계 요소 최소화 필요
- 전체 파이프라인의 통합 학습 요구
  - 간소화된 아키텍처 추구
- Transformer의 병렬 처리 활용 가능성

# DETR의 등장 배경

|                |                                    |
|----------------|------------------------------------|
| 연구 동기          | Facebook AI Research의 객체 탐지 단순화 목표 |
| Transformer 적용 | 자연어 처리 성공 모델의 비전 태스크 적용            |
| Set Prediction | 객체 집합을 직접 예측                       |
| 기존 방법과 차별점     | NMS와 anchor box 설계 없는 간결한 접근법      |
| 핵심             | End-to-End 학습 가능한 완전한 객체 탐지 시스템    |

# DETR

- 이 논문은 무엇을 제안했나요?
- DETR (DEtection TRansformer): ① 이분 매칭 손실 함수 + ② Transformer



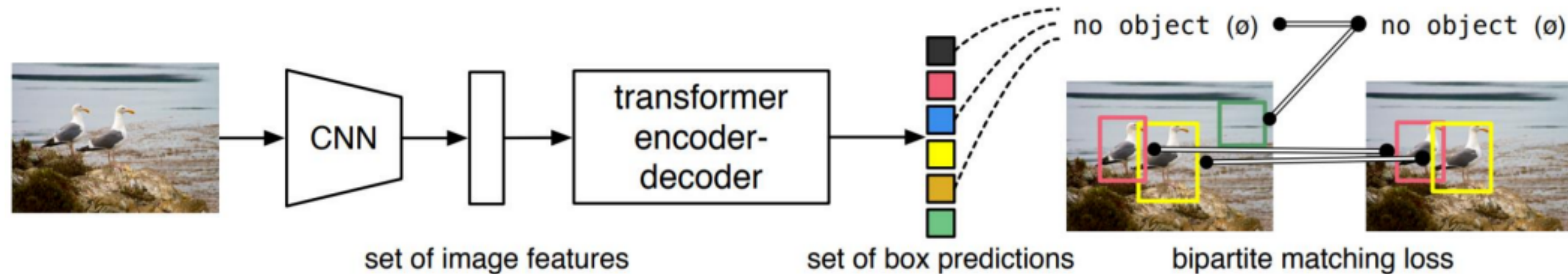
# Set Prediction 문제 정의

| 접근법   | 일대일 매칭  | 클래스 불균형  | No-object 처리  |
|---|---|--|---|
| <ul style="list-style-type: none"><li>- 객체 탐지를 고정된 크기의 집합 예측 문제로 재정의</li><li>- 순서에 무관한 예측 집합 생성</li></ul> | <ul style="list-style-type: none"><li>- 예측과 GT 객체 간의 일대일 매칭</li><li>- Hungarian 알고리즘 활용</li><li>- 최적 이분 매칭 수행</li></ul> | <ul style="list-style-type: none"><li>- 배경이 대부분인 이미지 특성 반영</li><li>- 클래스별 가중치 조정</li></ul> | <ul style="list-style-type: none"><li>- 고정된 수의 객체 쿼리 사용</li><li>- 객체 없음 클래스 별도 처리</li><li>- 배경 예측의 효율적 학습</li></ul> |

DETR은 객체 탐지를 Set Prediction 문제로 재정의 → Hungarian 알고리즘을 통한 일대일 매칭으로 예측 이는 NMS와 같은 후처리 과정 없이 End-to-End 학습

# 본 논문의 핵심 아이디어: 이분 매칭

- 이분 매칭(bipartite matching)을 통해 set prediction problem을 직접적으로(directly) 해결



- 학습 과정에서 이분 매칭을 수행함으로써 인스턴스가 중복되지 않도록 유도

출력 개수 고정:  $N = 6$

예측 결과

$(c_0 = \emptyset, b_0)$

$(c_1 = bird, b_1 = (180, 180, 150, 240))$

$(c_2 = \emptyset, b_2)$

$(c_3 = bird, b_3 = (120, 150, 100, 150))$

$(c_4 = \emptyset, b_4)$

$(c_5 = dog, b_5)$

$(c_0 = bird, b_0 = (122, 151, 100, 150))$   
 $(c_1 = bird, b_1 = (182, 180, 148, 238))$   
 $(c_2 = \emptyset, b_2)$   
 $(c_3 = \emptyset, b_3)$   
 $(c_4 = \emptyset, b_4)$   
 $(c_5 = \emptyset, b_5)$

실제 값

# 본 논문의 핵심 아이디어: Transformer

- DETR는 CNN이 추출한 이미지 특징 위에 Transformer를 적용하여 객체 탐지를 수행
  - Transformer의 self-attention을 이용해 이미지 전체 영역의 문맥 정보를 한 번에 고려
  - Encoder는 모든 픽셀 위치(또는 셀) 간의 관계를 학습하여 개별 인스턴스를 분리된 표현으로 인코딩
  - Decoder는 N개의 학습 가능한 object query를 입력받아, 각 query가 한 개의 객체(또는 배경)를 담당하도록 학습
  - 각 query의 출력은 클래스(label)와 정규화된 bounding box(cx, cy, w, h)이며, 이 출력 집합을 정답 객체 집합과 1:1로 매칭한다.
  - Hungarian 알고리즘을 이용해 NMS나 anchor 설계 없이도 End-to-End 학습이 가능하도록 한다.
-

## 본 논문의 핵심 아이디어: Transformer (Encoder)

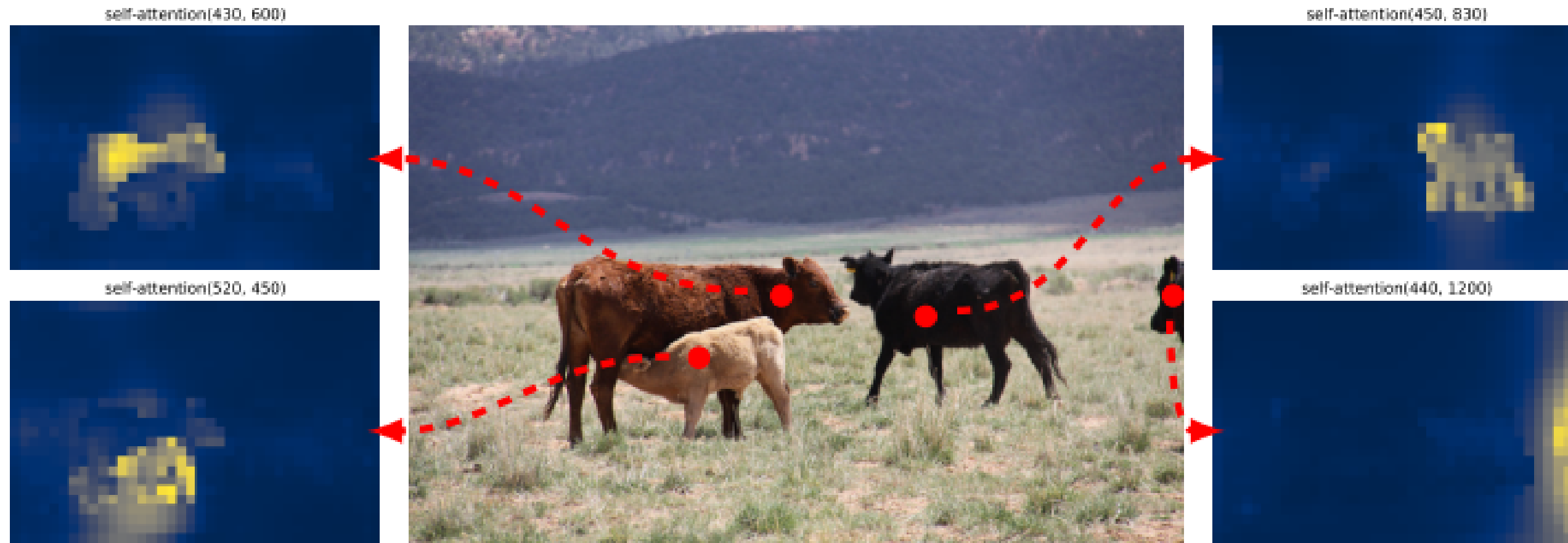


Fig. 3: Encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image.

Encoder는 모든 픽셀 위치(또는 셀) 간의 관계를 학습하여 개별 인스턴스를 분리된 표현으로 인코딩한다.

## 본 논문의 핵심 아이디어: Transformer (Decoder)

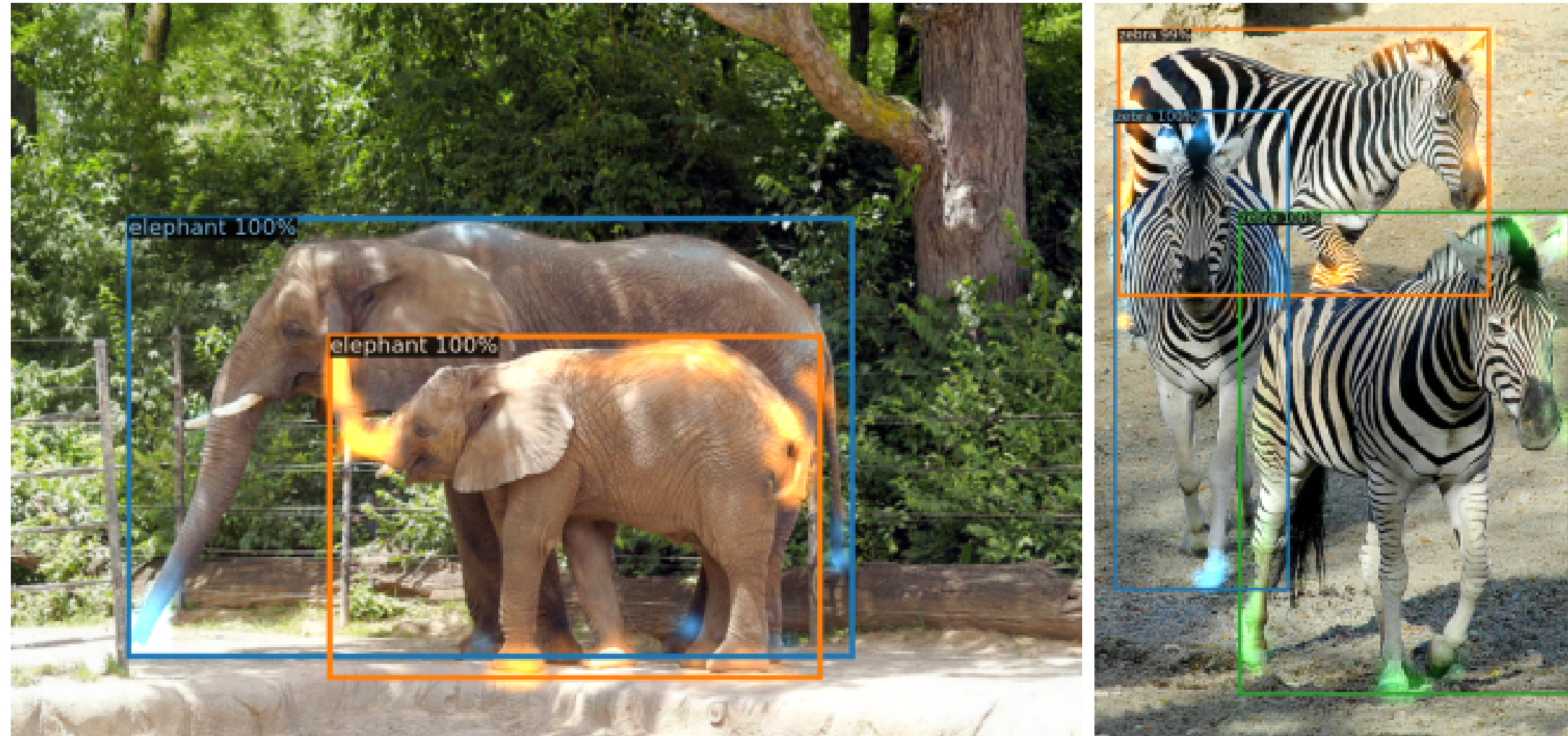


Fig. 6: Visualizing decoder attention for every predicted object (images from COCO val set). Predictions are made with DETR-DC5 model. Attention scores are coded with different colors for different objects. Decoder typically attends to object extremities, such as legs and heads. Best viewed in color.

# DETR의 Loss 함수

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

$L_{\text{iou}} : \text{GeneralizedIoUloss}$

$\|\cdot\|_1 : L1norm$

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

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

## Hungarian Loss의 구성 요소

- 분류(Classification) Loss: 클래스 예측을 위한 cross-entropy 손실 함수 사용
- 바운딩 박스(Bounding Box) Loss: L1 손실과 IoU 기반 손실의 조합

## Loss 함수의 균형과 가중치 설정

- 클래스 불균형 문제 해결을 위한 가중치 조정
- No-object 클래스에 낮은 가중치 부여로 학습 안정성 확보

# DETR의 성능 평가

| Model                 | GFLOPS/FPS | #params | AP   | AP <sub>50</sub> | AP <sub>75</sub> | AP <sub>S</sub> | AP <sub>M</sub> | AP <sub>L</sub> |
|-----------------------|------------|---------|------|------------------|------------------|-----------------|-----------------|-----------------|
| Faster RCNN-DC5       | 320/16     | 166M    | 39.0 | 60.5             | 42.3             | 21.4            | 43.5            | 52.5            |
| Faster RCNN-FPN       | 180/26     | 42M     | 40.2 | 61.0             | 43.8             | 24.2            | 43.5            | 52.0            |
| Faster RCNN-R101-FPN  | 246/20     | 60M     | 42.0 | 62.5             | 45.9             | 25.2            | 45.6            | 54.6            |
| Faster RCNN-DC5+      | 320/16     | 166M    | 41.1 | 61.4             | 44.3             | 22.9            | 45.9            | 55.0            |
| Faster RCNN-FPN+      | 180/26     | 42M     | 42.0 | 62.1             | 45.5             | 26.6            | 45.4            | 53.4            |
| Faster RCNN-R101-FPN+ | 246/20     | 60M     | 44.0 | 63.9             | 47.8             | 27.2            | 48.1            | 56.0            |
| DETR                  | 86/28      | 41M     | 42.0 | 62.4             | 44.2             | 20.5            | 45.8            | 61.1            |
| DETR-DC5              | 187/12     | 41M     | 43.3 | 63.1             | 45.9             | 22.5            | 47.3            | 61.1            |
| DETR-R101             | 152/20     | 60M     | 43.5 | 63.8             | 46.4             | 21.9            | 48.0            | 61.8            |
| DETR-DC5-R101         | 253/10     | 60M     | 44.9 | 64.7             | 47.7             | 23.7            | 49.5            | 62.3            |

## COCO 데이터셋 성능 비교

- Faster R-CNN과 유사한 AP 성능 달성(42.0% AP)
- 대형 객체 탐지에서 우수한 성능(AP\_L 61.1%로 Faster R-CNN 보다 높음)

## 정성적 결과 및 계산 복잡도

- 복잡한 장면에서 우수한 객체 구분 능력
- 추론 속도는 Faster R-CNN-FPN과 비슷한 수준이지만, 구조는 훨씬 단순한 파이프라인

# DETR의 장점

아키텍처 단순성

복잡한 파이프라인 없이 직관적인 구조 설계

NMS 제거

후처리 과정 없이 직접 객체 예측 가능

Global Context

전체 이미지 컨텍스트를 활용한 객체 관계 파악

학습 용이성

End-to-End 방식으로 통합된 학습 프로세스

# DETR의 한계점

| 소형 객체 탐지  | 느린 수렴 속도   | 계산 복잡도  | 메모리 사용량  |
|---|--|---|--|
| <ul style="list-style-type: none"><li>- 작은 객체 탐지 성능이 상대적으로 저조</li><li>- 특히 밀집된 환경에서 더 취약함</li><li>- 해상도 문제 존재</li></ul> | <ul style="list-style-type: none"><li>- 학습 시간이 기존 모델보다 오래 걸림</li><li>- 수렴까지 많은 에포크 필요</li><li>- 초기 학습이 불안정</li></ul> | <ul style="list-style-type: none"><li>- Transformer의 Self-attention 연산 복잡도 높음</li><li>- <math>O(n^2)</math> 시간 복잡도</li><li>- 추론 시간 증가</li></ul> | <ul style="list-style-type: none"><li>- 많은 파라미터로 인한 높은 메모리 요구량</li><li>- 배치 크기 제한</li><li>- 학습 환경 제약</li></ul> |