# End-to-End Object Detection with Transformers <DEtection TRansformer>

- Facebook AI <2020.05.26>

ComputerVision, transformer

백 대 환

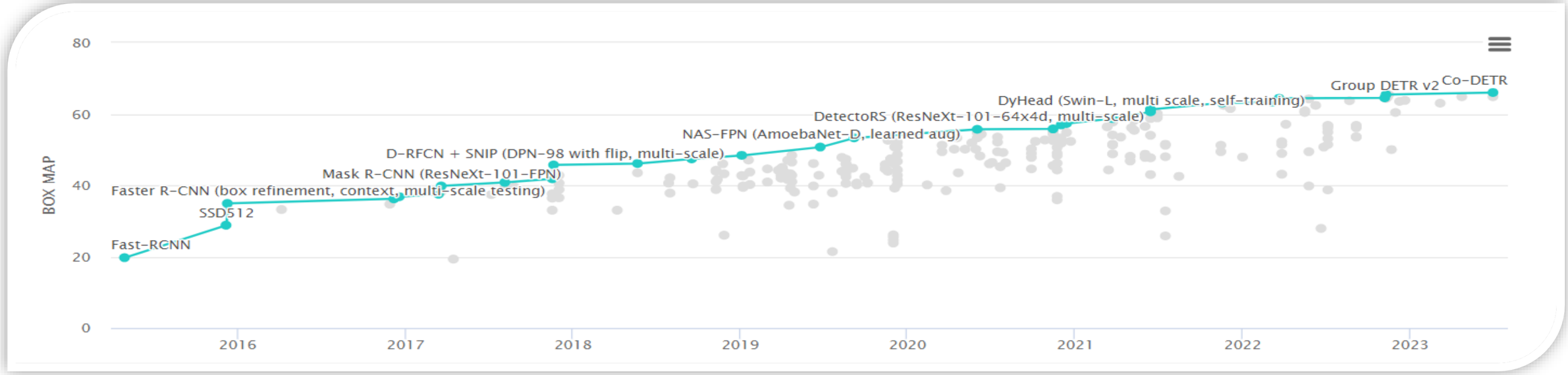
PaperReview

# OVERVIEW

- Computer Vision

➢ YOLO

➢ FASTER R-CNN



# 0. Abstract

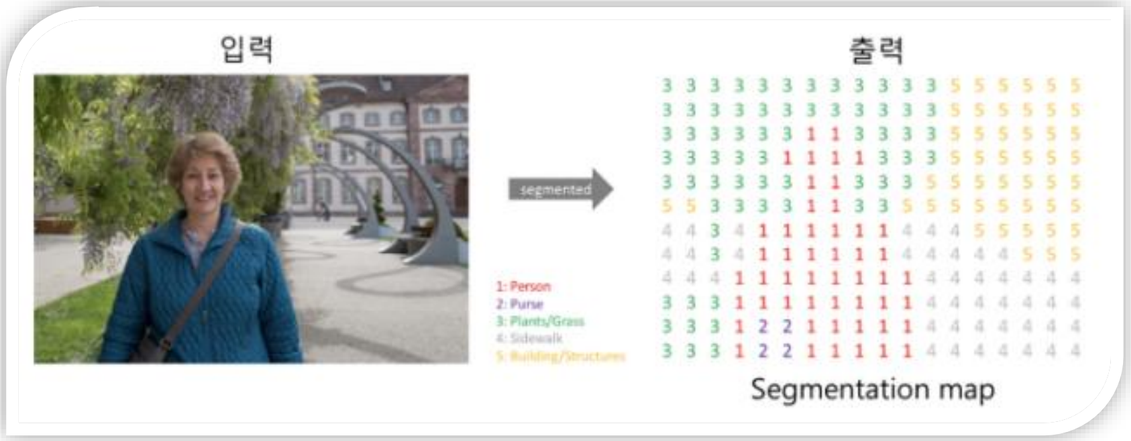
- European Conference on Computer Vision 학회 (ECCV 2020)

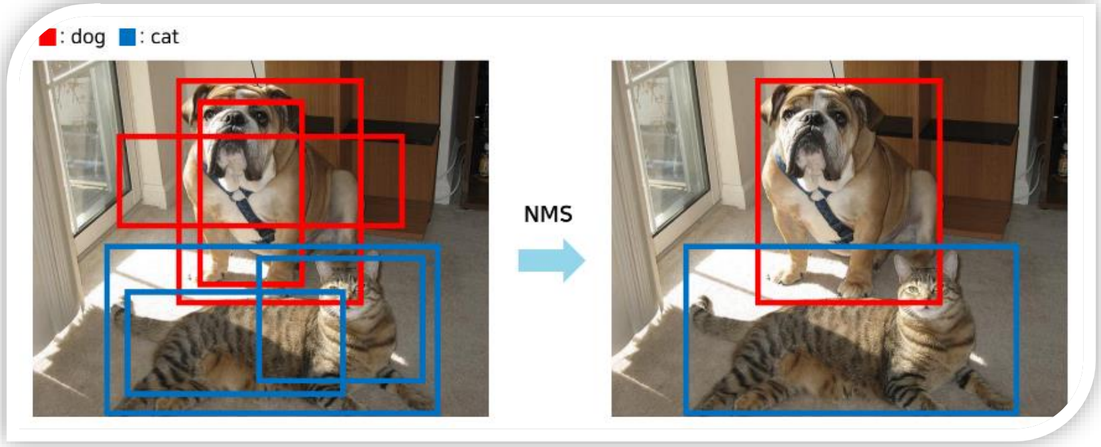
➢ Object detection(bounding box와 category label 집합을 찾는 과정)을 direct set prediction problem으로 간주하고 해결

➢ Prior knowledge(NMS)를 사용하지 않아 detection pipeline을 간소화

➢ 객체 쿼리를 고정해놓고, 객체간 관계와 Prediction을 병렬적으로 추론 가능

➢ SOTA baseline 모델 중 하나인 Faster R-CNN, YOLO 보다 매우 간단하고 나은 성능



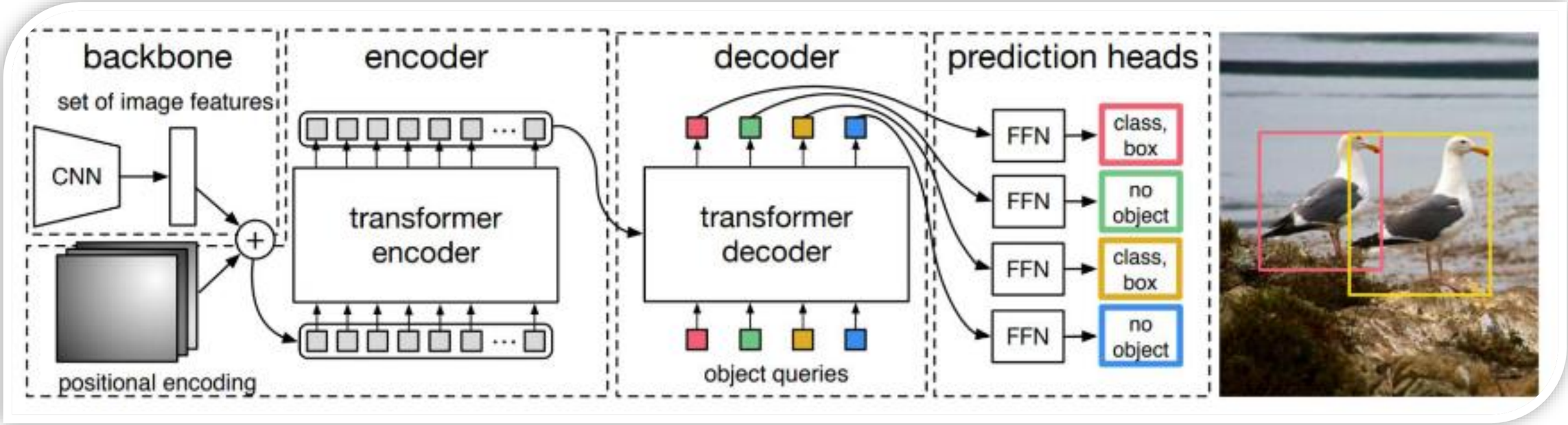
# 1. Introduction

- Introduction

➢ 특징 - 1. 이분 매칭 손실 함수 + 2. Transformer

➢ Transformer 기반의 encoder-decoder 구조

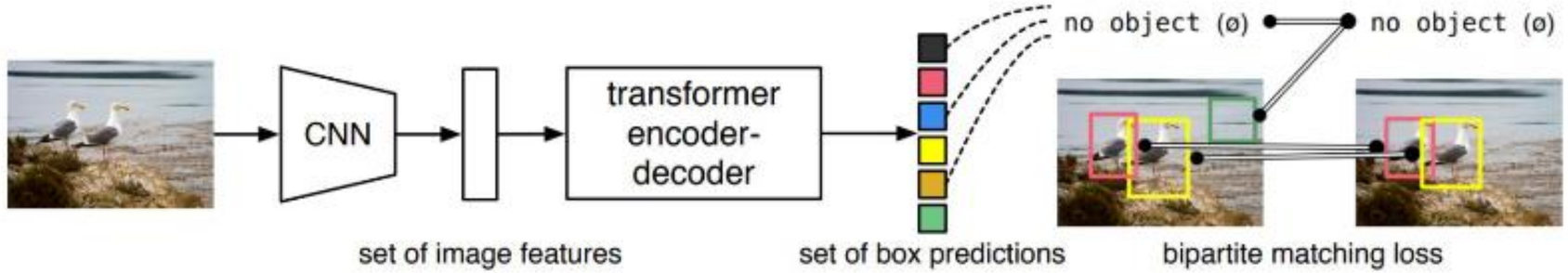
➢ Backbone - ResNet-50, ResNet-101, DETR-R101, DETR-DC5, DETR-DC5-R101



# 2. Related work

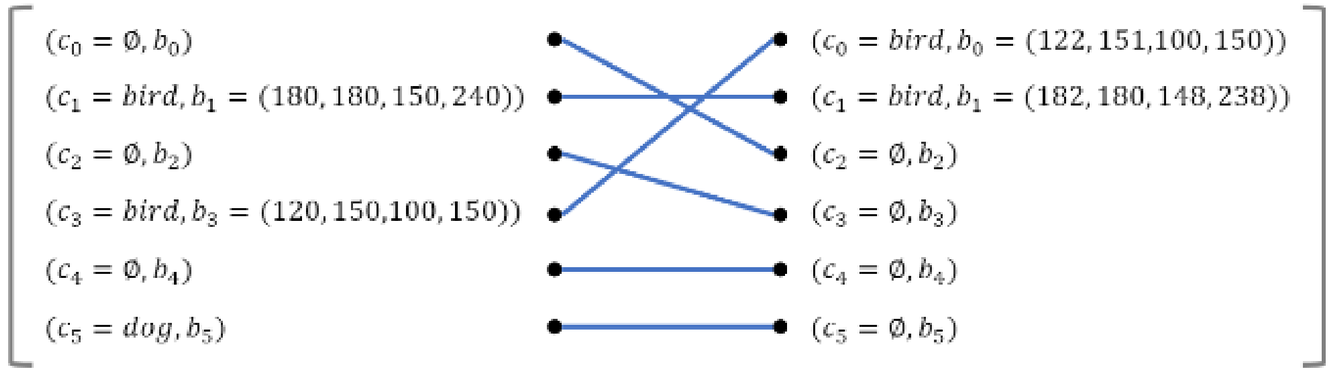
- 핵심 아이디어 <1> bipartite matching

➢ 이분 매칭(bipartite matching) : set prediction problem을 직접적으로 해결(directly)




➢ 학습 과정에서 이분 매칭을 수행함으로써 인스턴스가 중복되지 않도록 유도함

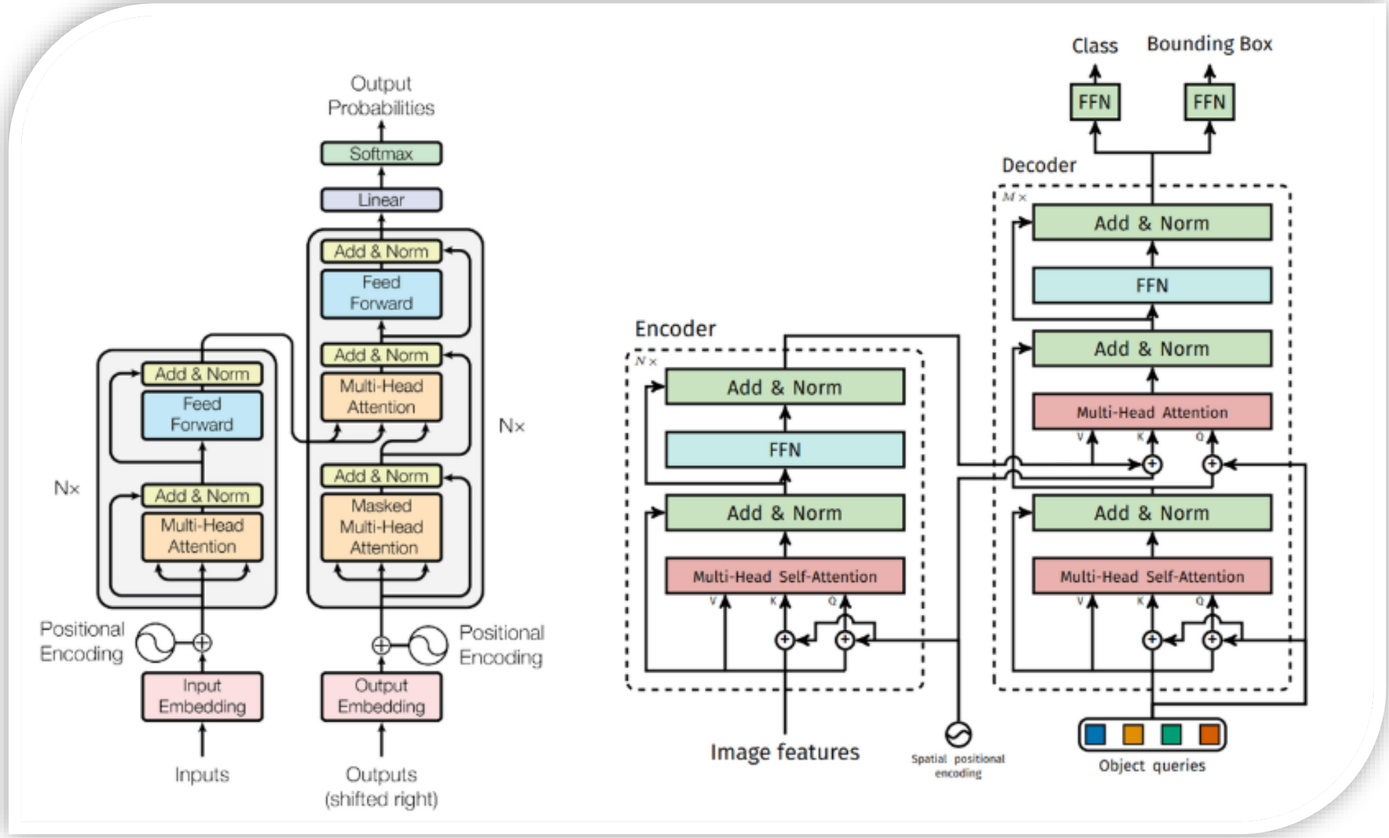
출력 개수 고정: $N = 6$



# 2. Related work

- 핵심 아이디어 <2> transformer

➢ NLP task vs DETR




1. 문장 vs 이미지(픽셀)

2. positional encoding

Embedding의 순서 상관 없이 동일 값 출력

vs x, y axis가 있는 2D 크기의 feature map 입력

3. Decoder

Masked multi attention

vs Multi-head self-attention

4. decoder 이후 하나의 head vs 두 개의 head

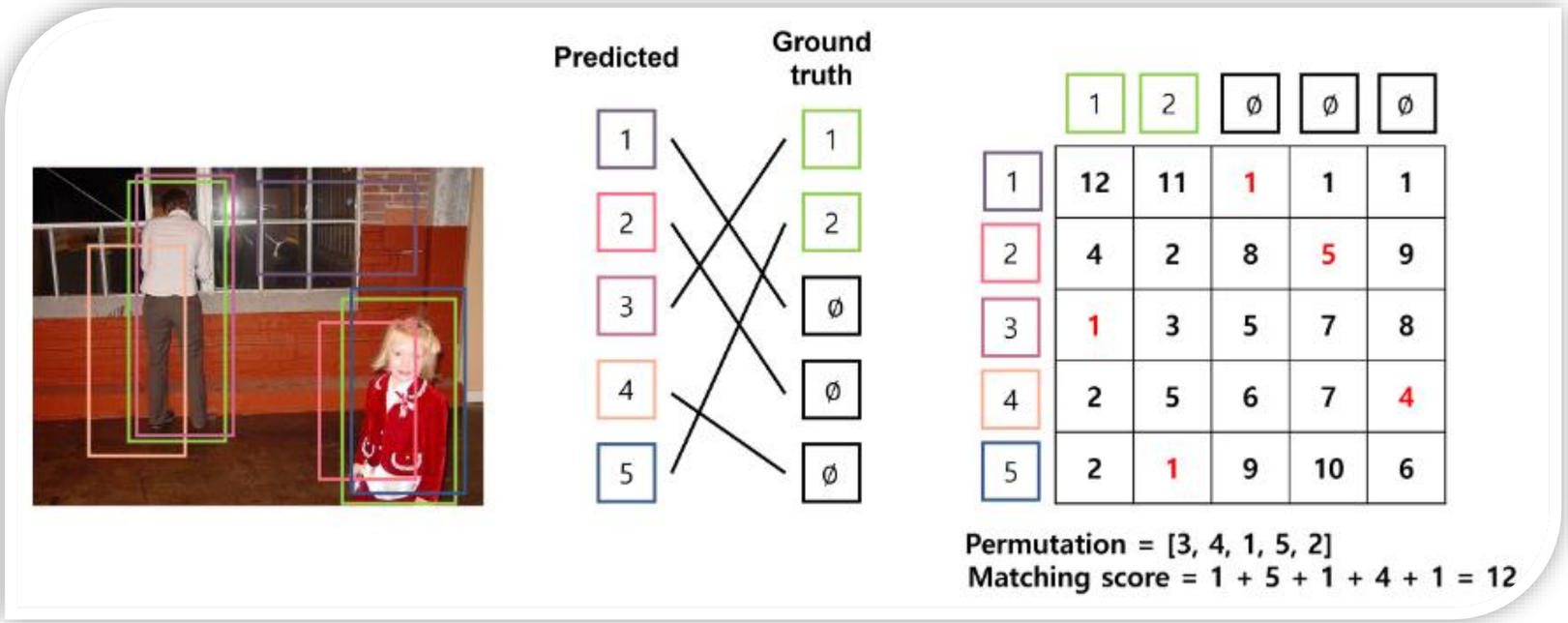
# 3. DETR Model

- 모델 설명 들어가기 전,

➢ Hungarian algorithm -> 이분 매칭 기반

➢ 두 집합 사이의 일대일 대응 시 가장 비용이 적게 드는 bipartite matching(이분 매칭)을 찾는 알고리즘

➢ Loss를 최소화 하는 값을 효율적으로 찾기 위해 사용



# 3. DETR Model

- 모델 설명

➢ Object detection set prediction loss (Find optimal matching)

σ: Ground truth의 object set의 순열
σ_hat: L_match를 최소로 하는 예측 bounding box set의 순열
y: Ground truth의 object set // y_hat: 예측한 N object set
c: class label // p(c): 해당 class에 속할 확률
b: bounding box의 위치와 크기 (x, y, w, h)

### 3.1 Object detection set prediction loss

DETR infers a fixed-size set of $N$ predictions, in a single pass through the decoder, where $N$ is set to be significantly larger than the typical number of objects in an image. One of the main difficulties of training is to score predicted objects (class, position, size) with respect to the ground truth. Our loss produces an optimal bipartite matching between predicted and ground truth objects, and then optimize object-specific (bounding box) losses.

Let us denote by $y$ the ground truth set of objects, and $\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$ the set of $N$ predictions. Assuming $N$ is larger than the number of objects in the image, we consider $y$ also as a set of size $N$ padded with $\varnothing$ (no object). To find a bipartite matching between these two sets we search for a permutation of $N$ elements $\sigma \in \mathfrak{S}_N$ with the lowest cost:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}), \tag{1}$$

$$\hat{\sigma} = argmin_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma}(i))$$

where $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ is a pair-wise *matching cost* between ground truth $y_i$ and a prediction with index $\sigma(i)$. This optimal assignment is computed efficiently with the Hungarian algorithm, following prior work (*e.g.* [43]).

The matching cost takes into account both the class prediction and the similarity of predicted and ground truth boxes. Each element $i$ of the ground truth set can be seen as a $y_i = (c_i, b_i)$ where $c_i$ is the target class label (which may be $\varnothing$) and $b_i \in [0, 1]^4$ is a vector that defines ground truth box center coordinates and its height and width relative to the image size. For the prediction with index $\sigma(i)$ we define probability of class $c_i$ as $\hat{p}_{\sigma(i)}(c_i)$ and the predicted box as $\hat{b}_{\sigma(i)}$. With these notations we define $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ as $-\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$.

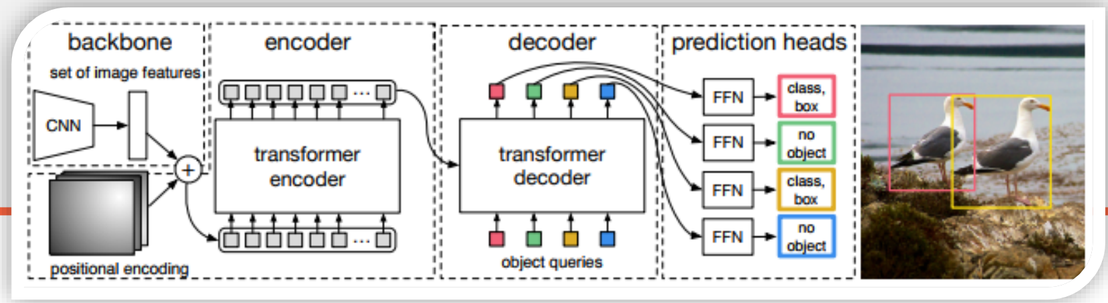
This procedure of finding matching plays the same role as the heuristic assignment rules used to match proposal [37] or anchors [22] to ground truth objects in modern detectors. The main difference is that we need to find one-to-one matching for direct set prediction without duplicates.

The second step is to compute the loss function, the *Hungarian loss* for all pairs matched in the previous step. We define the loss similarly to the losses of common object detectors, *i.e.* a linear combination of a negative log-likelihood for class prediction and a box loss defined later:

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[ -\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right], \tag{2}$$

$$\mathcal{L}_{match} = -1_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + 1_{\{c_i \neq \varnothing\}} \mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)})$$

# 3. DETR Model



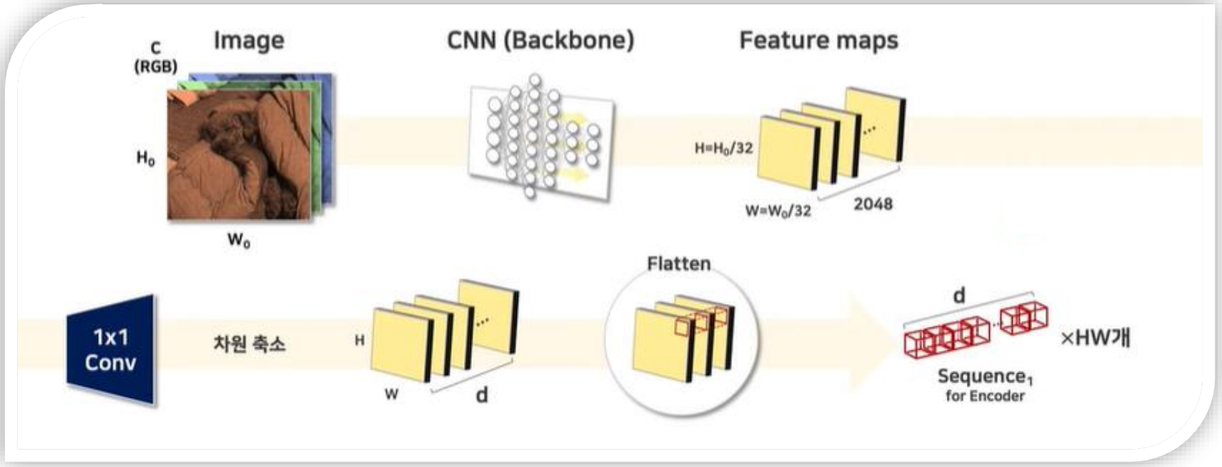

- 모델 설명 - DETR architecture

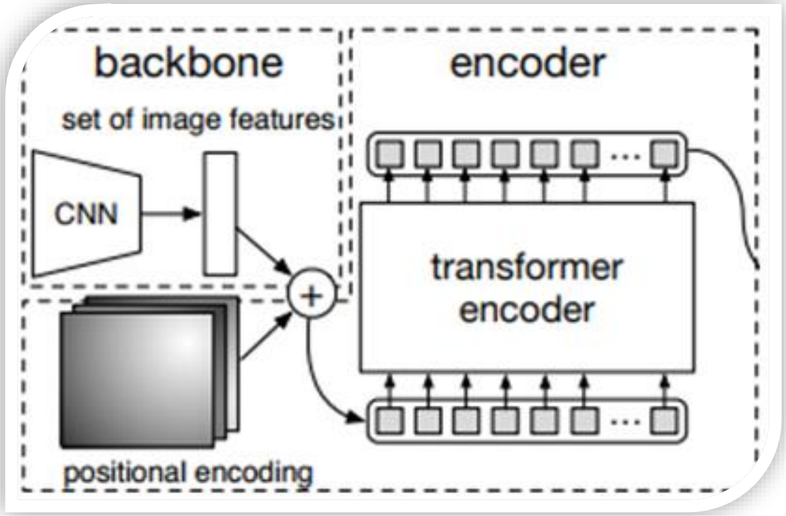
➢ **Backbone:** 어떤 feature를 추출하기 위해서 사용되는 구조

➢ **Transformer encoder:** d×HW의 feature matrix에 Positional encoding 정보를 더한 matrix를 multi-head self-attention에 통과시킵니다. Transformer의 특성 상 입력 matrix와 출력 matrix의 크기는 동일

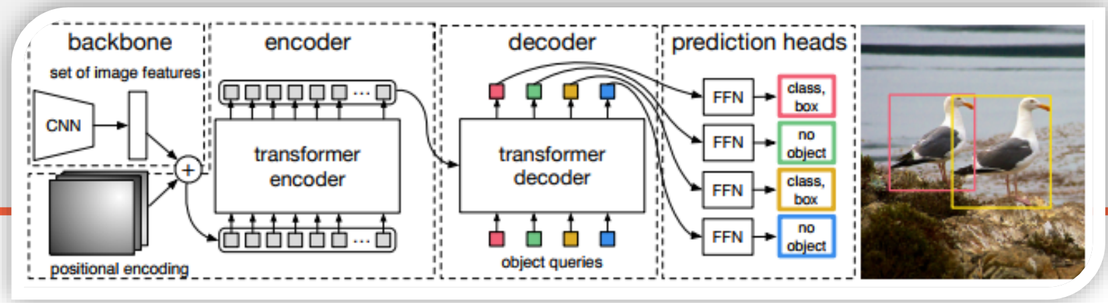
각 encoder layer는 multi-head self-atttention module과 feed forward network(FFN)으로 구성되어. Transformer 구조는 입력 embedding의 순서와 상관없이 같은 출력값을 생성하는 permutation-invariant 속성이 있기 때문에 encoder layer 입력 전에 입력 embedding에 positional encoding을 더해줌

➢ Object query는 그릇, 슬롯: 이미지의 어느 부분을 위주로 봐야할 지, 일대일 매칭을 어떻게 수행할지



# 3. DETR Model




- 모델 설명 - DETR architecture

➢ Transformer Decoder

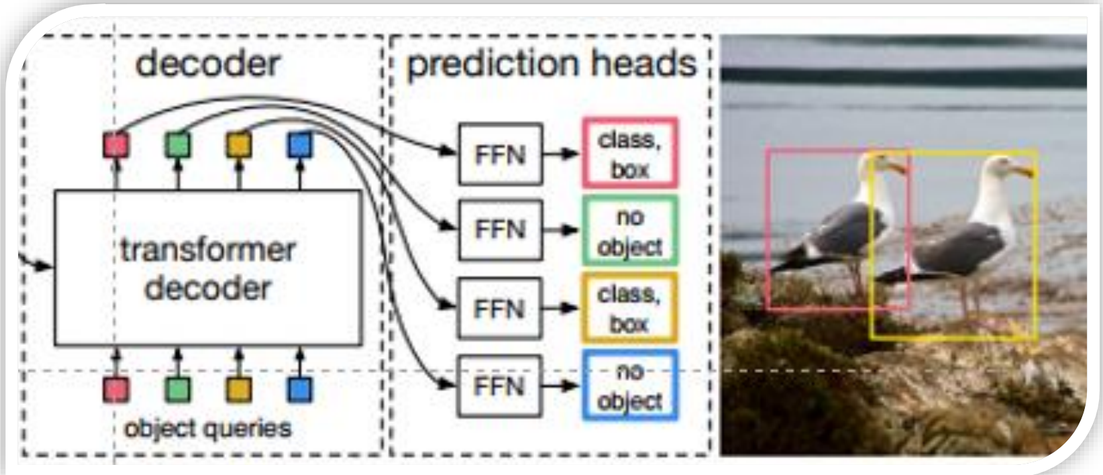
- N개의 bounding box에 대해 N개의 object query를 생성합니다. 초기 object query는 0으로 설정되어있습니다.
- Decoder는 앞서 설명한 N개의 object query를 입력받아 multi-head self-attention을 거쳐 가공된 N개의 unit을 출력합니다.
- 이 N개의 unit들이 Query로, 그리고 encoder의 출력 unit들이 Key와 Value로 작동하여 multi-head attention을 수행합니다.

➢ Prediction feed-forward networks (FFNs)

- 이때 bi-partite matching을 통해 각 bounding box가 겹치지 않도록 함.
- Decoding을 거쳐 최종적으로 예측되는 정보는 중심 정보(x, y), 높이, 너비와 클래스
- 만약 클래스가 없다면 "no object"로 결과가 산출되고 이미지 상 Background에 해당

➢ Auxiliary decoding losses

- 학습 시, 각 decoder layer마다 FFN을 추가하여 auxiliary loss를 구함.
- 이러한 보조적인 loss를 사용할 경우 모델이
- 각 class별로 올바른 수의 객체를 예측하도록 학습시키는데 도움을 줌.



# 4. Experiments

- Experiments

➢ **Backbone -** ResNet-50, ResNet-101, DETR-R101, DETR-DC5, DETR-DC5-R101

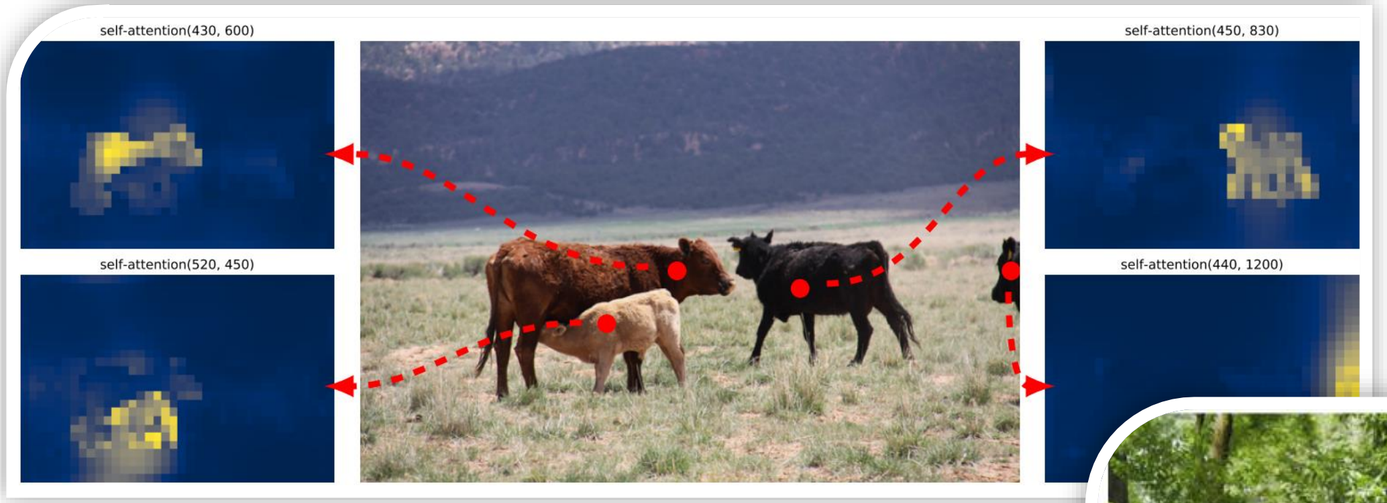
➢ **COCO Dataset -** 120만개 training, 5천개 validation, 이미지에 존재하는 인스턴스는 7개, 최대는 63개

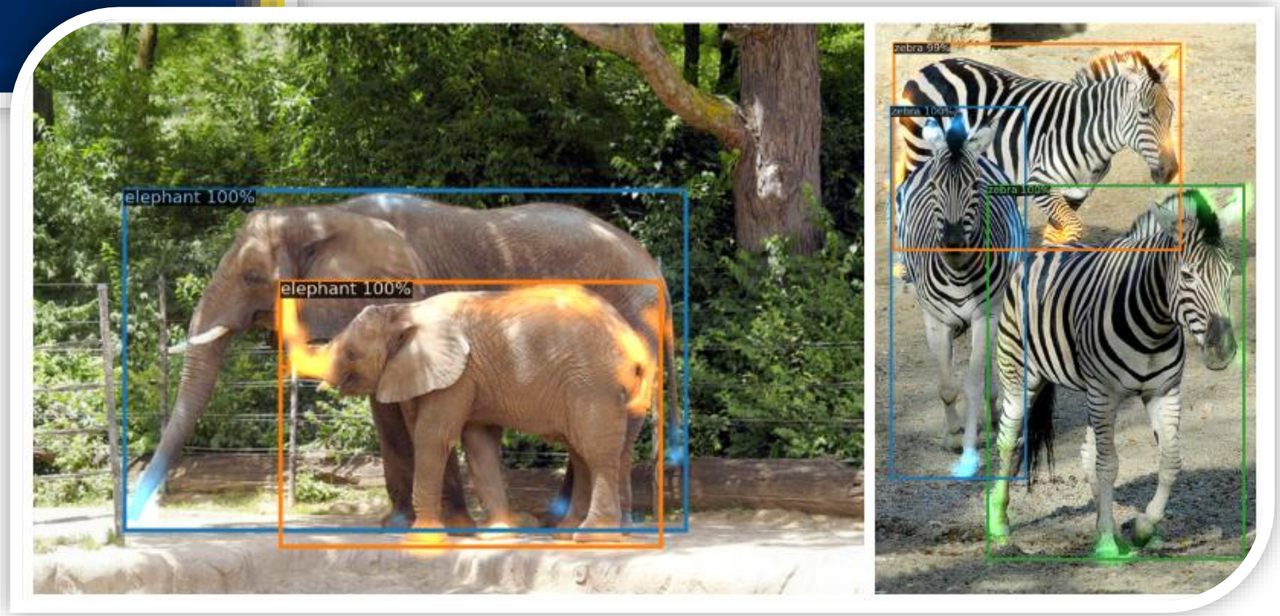
➢ architecture와 loss에 관한 **Ablation study**를 수행.

| Model | GFLOPS/FPS | #params | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|
| 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** |

# 4. Experiments

- Experiments



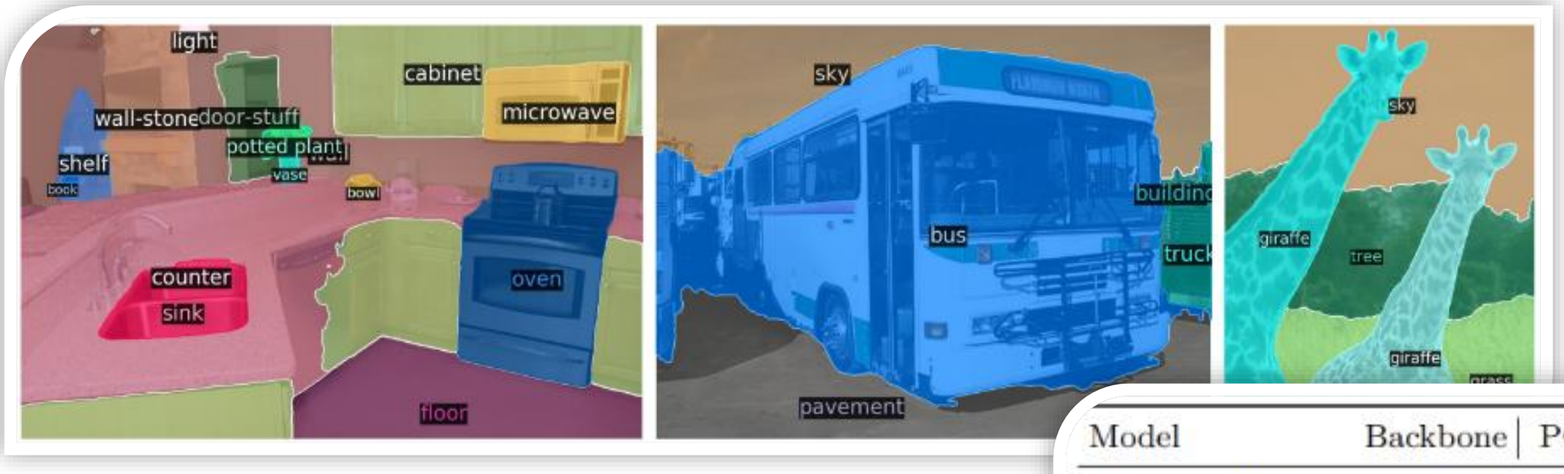



# 4. DETR for panoptic segmentation

- Segmentation

➢ 프로세스를 서로 크기가 다른 논리적인 블록 단위인 Segment로 분할하여 메모리에 할당

➢ segment들의 크기가 서로 다르기 때문에 프로세스가 메모리에 적재될 때 빈 공간을 찾아 할당하는 기법




| Model | Backbone | PQ | SQ | RQ | $PQ^{th}$ | $SQ^{th}$ | $RQ^{th}$ | $PQ^{st}$ | $SQ^{st}$ | $RQ^{st}$ | AP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PanopticFPN++ | R50 | 42.4 | 79.3 | 51.6 | 49.2 | 82.4 | 58.8 | 32.3 | 74.8 | 40.6 | 37.7 |
| UPSnet | R50 | 42.5 | 78.0 | 52.5 | 48.6 | 79.4 | 59.6 | 33.4 | 75.9 | 41.7 | 34.3 |
| UPSnet-M | R50 | 43.0 | 79.1 | 52.8 | 48.9 | 79.7 | 59.7 | 34.1 | 78.2 | 42.3 | 34.3 |
| PanopticFPN++ | R101 | 44.1 | 79.5 | 53.3 | **51.0** | **83.2** | 60.6 | 33.6 | 74.0 | 42.1 | **39.7** |
| DETR | R50 | 43.4 | 79.3 | 53.8 | 48.2 | 79.8 | 59.5 | 36.3 | 78.5 | 45.3 | 31.1 |
| DETR-DC5 | R50 | 44.6 | 79.8 | 55.0 | 49.4 | 80.5 | 60.6 | **37.3** | **78.7** | **46.5** | 31.9 |
| DETR-R101 | R101 | **45.1** | **79.9** | **55.5** | 50.5 | 80.9 | **61.7** | 37.0 | 78.5 | 46.0 | 33.0 |

# 5. Conclusion

- Conclusion

➢ Transformer를 object detection에 최초로 적용, 간단한 코딩

➢ object detection을 set prediction task로 정의하여 prediction과 ground truth 사이의 일대일 matching을 수행

➢ near duplicate prediction을 효과적으로 감소시켜 post-processing 과정이 없는 end-to-end 프레임워크를 제안

➢ **한계점**

- 여러 크기의 anchor를 사용하지 않기 때문에 다양한 크기, 형태의 객체를 포착하지 못함
- 하나의 예측 bounding box를 converge하는데 훨씬 긴 학습시간을 필요로 합니다.

➢ 제안한 DETR은 Faster R-CNN보다 나은 성능을 보이면서도 heuristic한 과정이나 post-processing을 필요하지 않다는 점에서 큰 의의를 가짐

➢ 단점을 보완한 Faster R-CNN과 YOLO 보다 나은 속도와 정확도를 가지는 RT-DETR이 만들어짐

➢ https://arxiv.org/pdf/2304.08069.pdf < DETRs Beat YOLOs on Real-time Object Detection > - 2023. 4. 17.