

DETR: End-to-End Object Detection with Transformers (ECCV 2020)

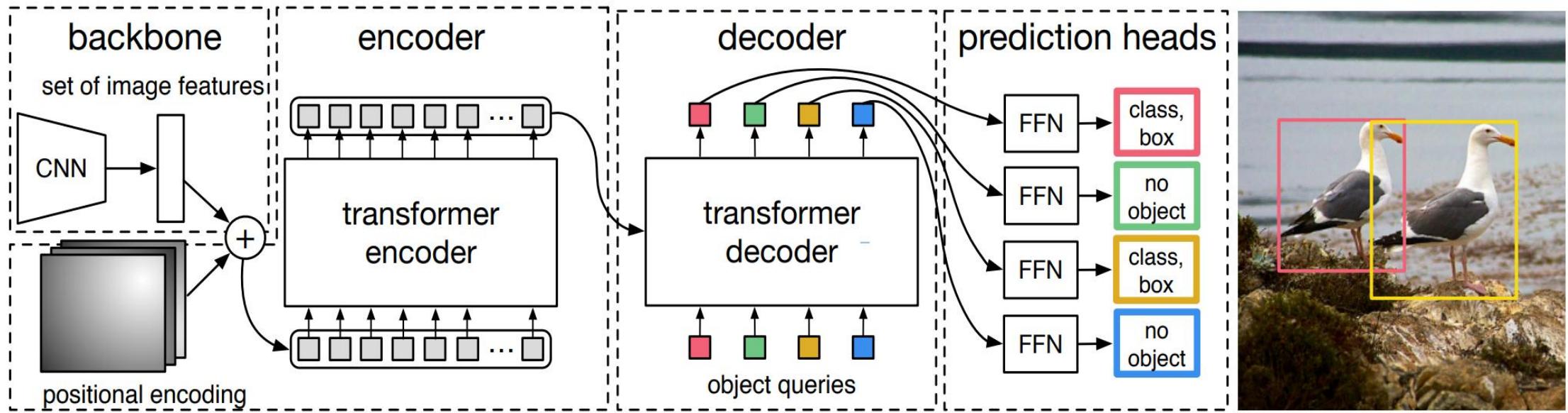


Table of Contents

I. Introduction

II. 핵심 아이디어

III. 실험 결과 및 한계

IV. Questions

Chapter I

Introduction

1.1 연구 배경: 기존 방법의 문제점

| 기존 객체 탐지(object detection) 방법들은 너무 **복잡**하며 다양한 라이브러리를 활용함

| 사전 지식(prior knowledge) 요구: bounding box의 형태, bounding box가 겹칠 때의 처리 방법, ..

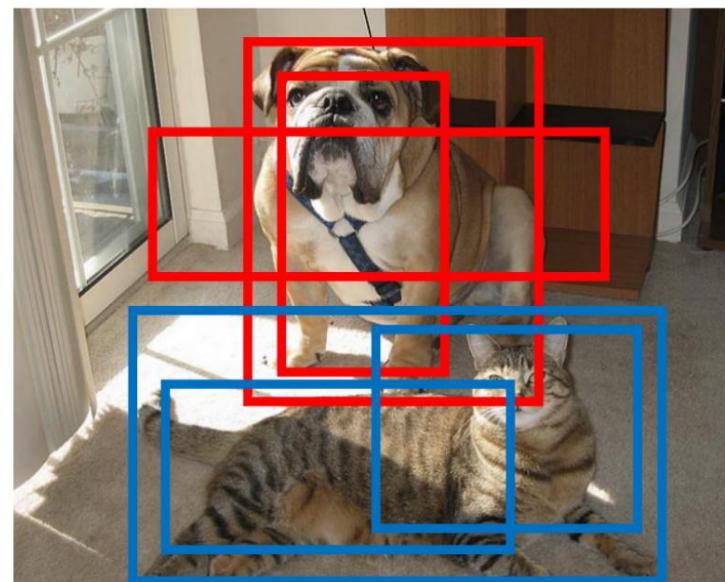
→

이렇게 해야 더 좋은 성능이 나오고 instance 구분을 잘 함

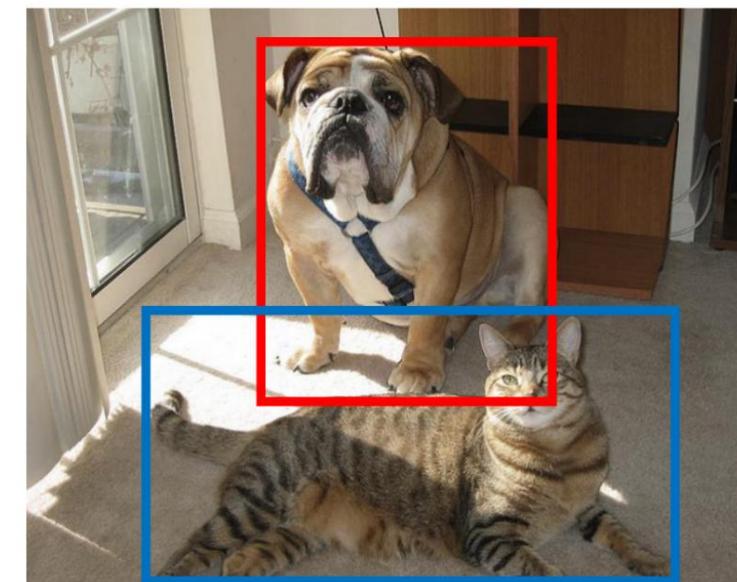
- prior knowledge: 모델링 하는 사람이 미리 알고 있으면 좋은 정보

- Ex) 탐지하려는 객체가 기차 같이 긴 게 많은 경우 bbox를 길게 설계

■ : dog ■ : cat



NMS
→



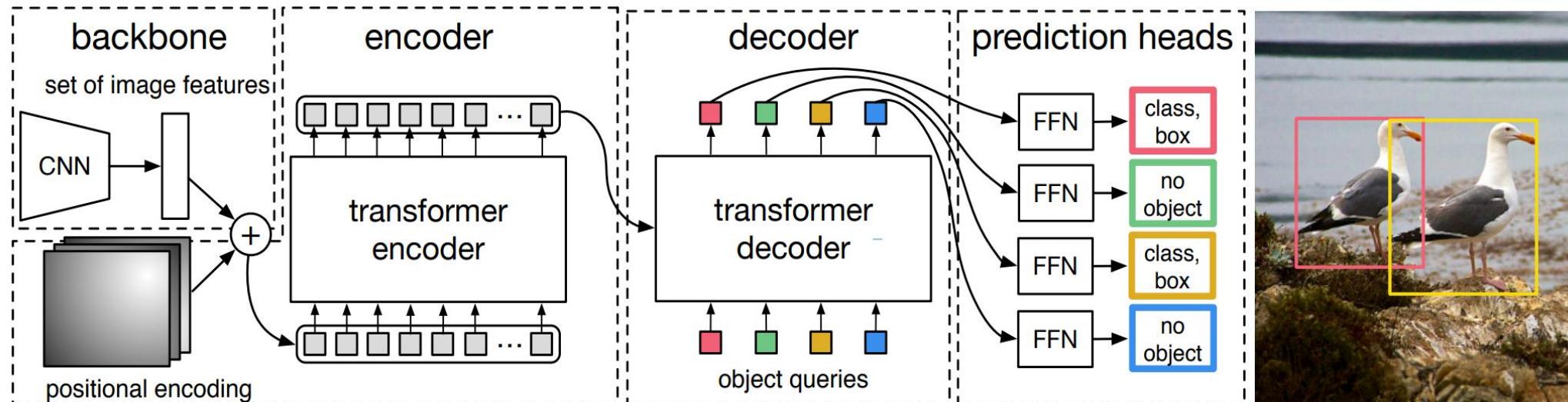
1.2 본 논문이 주목 받은 이유 (장점)

■ 본 논문의 장점

- 기존의 객체 탐지(Object detection) 기술과 비교했을 때 **매우 간단**하며 또한 **경쟁력 있는 성능**을 보임
- 어떤 Task에 대해서 Anchor 형태, bbox 설정 방법 등에 대한 정보를 사람이 일일이 설정할 필요 없이 data-driven(end to end)으로 모델이 자연스럽게 학습

■ 본 논문은 무엇을 제안?

- DETR (DEtection TRansformer): [1] 이분 매칭 손실 함수 + [2] Transformer



Chapter II

핵심 아이디어

2.1 핵심 아이디어: 이분 매칭 (Object detection을 set prediction problem으로 봄)

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

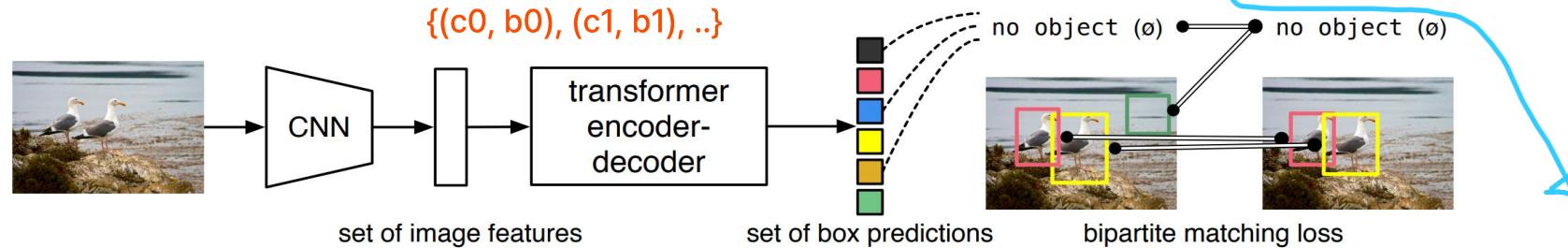


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a “no object” (\emptyset) class prediction.

수학에서의 집합을 의미

- 중복되는 원소 없음
- 순서 상관 없음 (객체 탐지에서 bbox 순서가 바뀌어도 상관 없음)

기존에는 Region Proposal → NMS 등 처럼 간접적으로 해결

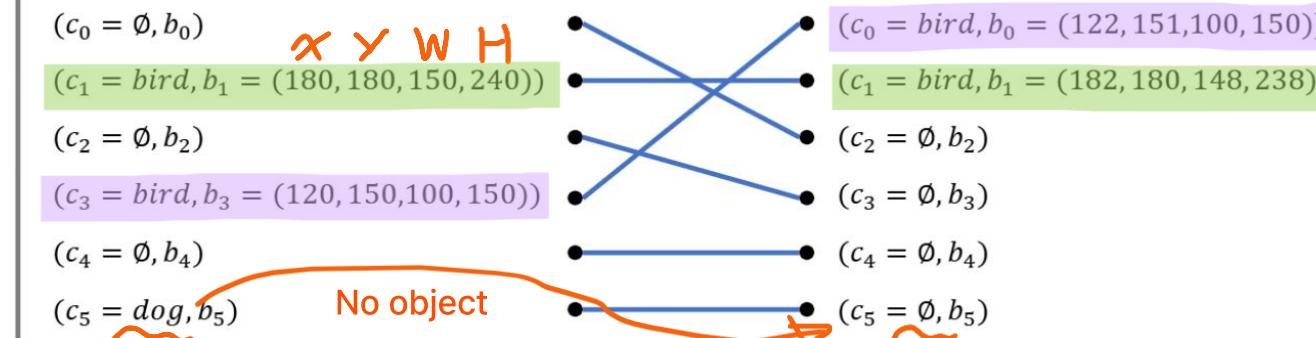
→ postprocessing에 따라서 성능이 달라졌기 때문에 본 논문은 이런 파이프라인을 간소화하고 directly하게 해결 (End-to-end)

| 학습 과정에서 이분 매칭을 수행함으로써 인스턴스가 중복되지 않도록 유도

출력 개수 고정: $N = 6$

이미지에 존재할 수 있는 instance 수를 고려해서 충분히 크게 설정하는 게 중요

예측 결과



클래스가 같고 bbox가 유사할 때 더 낮은 loss를 가지도록 매칭 loss를 구성

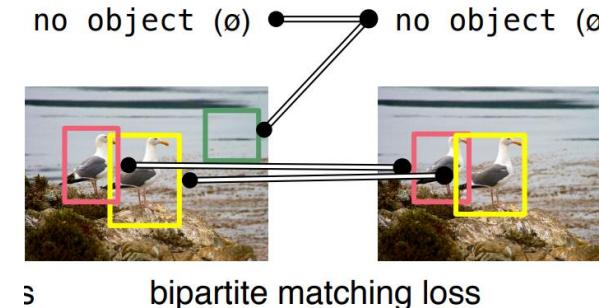
실제 값

2.1 핵심 아이디어: 이분 매칭 - Object detection set prediction loss

1-1. 최적 매칭 결과 찾기($\hat{\sigma}$)

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

- 가능한 모든 매칭 결과 중에서 로스가 가장 낮은 케이스의 매칭 결과(sigma)



1-2. 매칭 loss

prediction with index $\sigma(i)$ we define probability of class c_i as $\hat{p}_{\sigma(i)}(c_i)$
the predicted box as $\hat{b}_{\sigma(i)}$.

$$\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = -\mathbb{1}_{\{c_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \emptyset\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)}).$$

- 클래스가 잘 매칭 되고 bbox도 유사할 때 loss 값이 낮게 나오도록 구현

2. Hungarian loss: 이분 매칭 수행 후 그 결과를 통해 나온 output이 실제 클래스와 bbox에 유사하도록 만들기

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

결론: 이런식으로 이분 매칭을 하기 때문에 중복된 결과를 피할 수 있음!

2.1 핵심 아이디어: 이분 매칭 - Object detection set prediction loss

3. Bounding box loss

두 Bbox가 얼마나 유사한지를 평가

$$\mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$$

=

$$\lambda_{\text{iou}} \mathcal{L}_{\text{iou}}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{\text{L1}} \|b_i - \hat{b}_{\sigma(i)}\|_1$$

where $\lambda_{\text{iou}}, \lambda_{\text{L1}} \in \mathbb{R}$ are hyperparameters.

Bbox가 많이 겹치도록

두 Bbox가 유사해지도록

2.2 핵심 아이디어: Transformer

Transformer

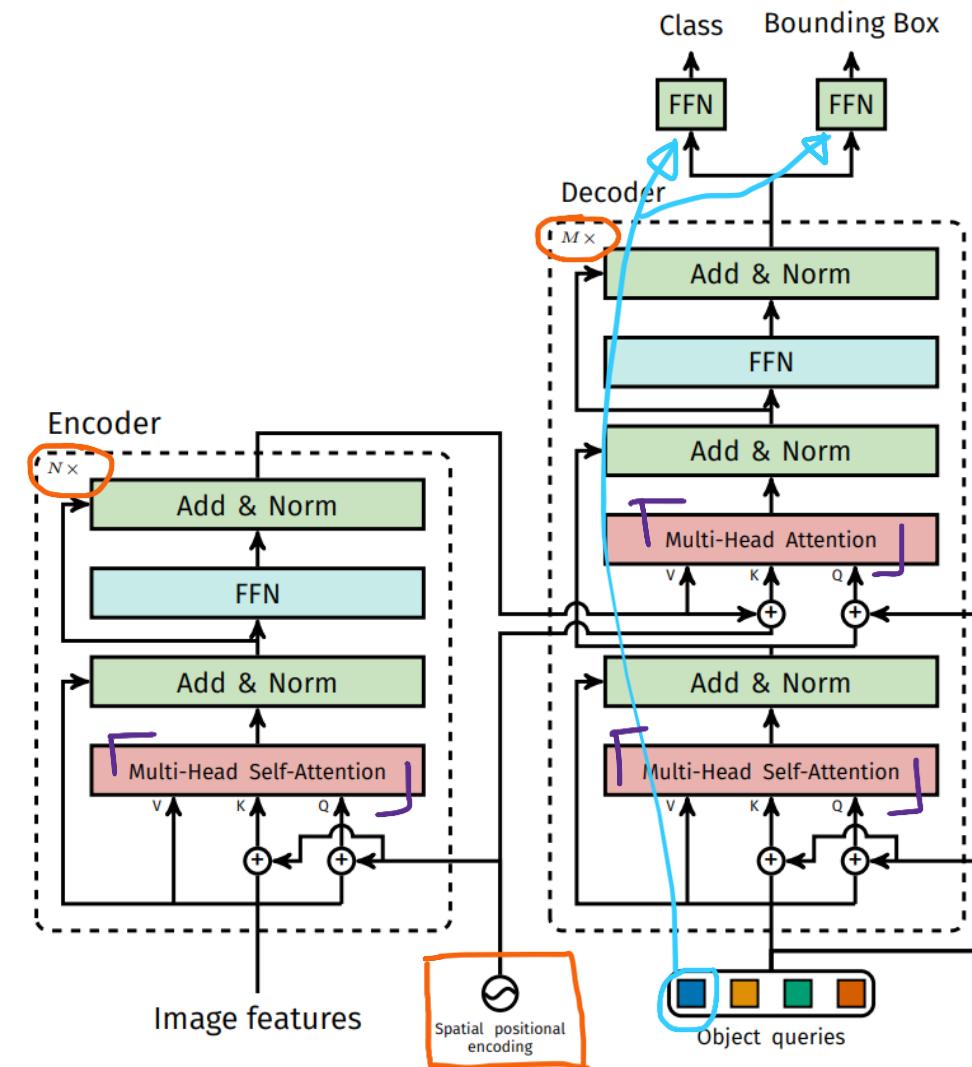
- Attention을 통해 전체 이미지의 문맥 정보를 이해
- 이미지 내 각 인스턴스의 상호작용(interaction) 파악 용이
 - Attention으로 각 픽셀들이 서로 score를 매기고 하는 과정에서 instance가 분리가 되고 각 instance에 대해 interaction값들을 파악
- 큰 bounding box에서의 거리가 먼 픽셀 간의 연관성 파악 용이

Encoder

- 이미지의 특징(feature) 정보를 포함하고 있는 각 픽셀 위치 데이터(feature map)를 입력 받아 인코딩 수행 – **임베딩 정보로 image feature 사용**
- Self-attention으로 feature map의 각 픽셀에 대한 연관성 정보 학습 (어떤 관계가 있는지 global하게 학습)

Decoder

- N개의 object query를 초기 입력으로 받으며 **인코딩된 정보를 활용**(전체 이미지에 대한 context 파악)
- 각 object query는 이미지 내 서로 다른 고유한 인스턴스를 구별
- object query:
 - output positional encodings (object queries)
 - learnable positional encodings
 - small fixed number of learned positional embeddings

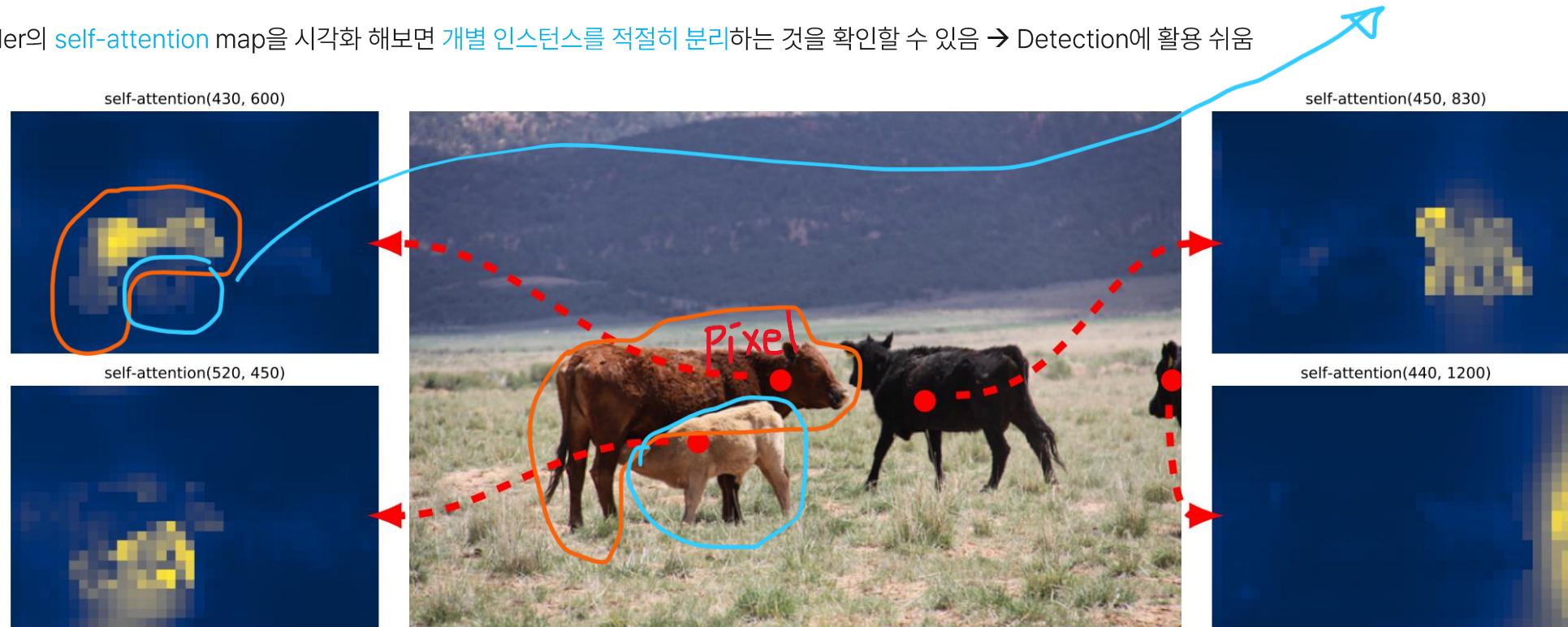


2.2 핵심 아이디어: Transformer (Encoder)

Encoder

- Encoder는 $d \times HW$ 크기의 연속성을 띠는 **feature map**을 입력으로 받음
 - 이때 d 는 image feature(각 픽셀에 대한 image feature 임베딩)를 의미하고 **HW는 각각의 픽셀 위치 정보**를 담고 있음
- Encoder의 **self-attention** map을 시각화 해보면 **개별 인스턴스를 적절히 분리**하는 것을 확인할 수 있음 → Detection에 활용 쉬움

Occluding 문제도 잘 해결:
뒤쪽에 있는 소만 진짜 instance임을
잘 구별

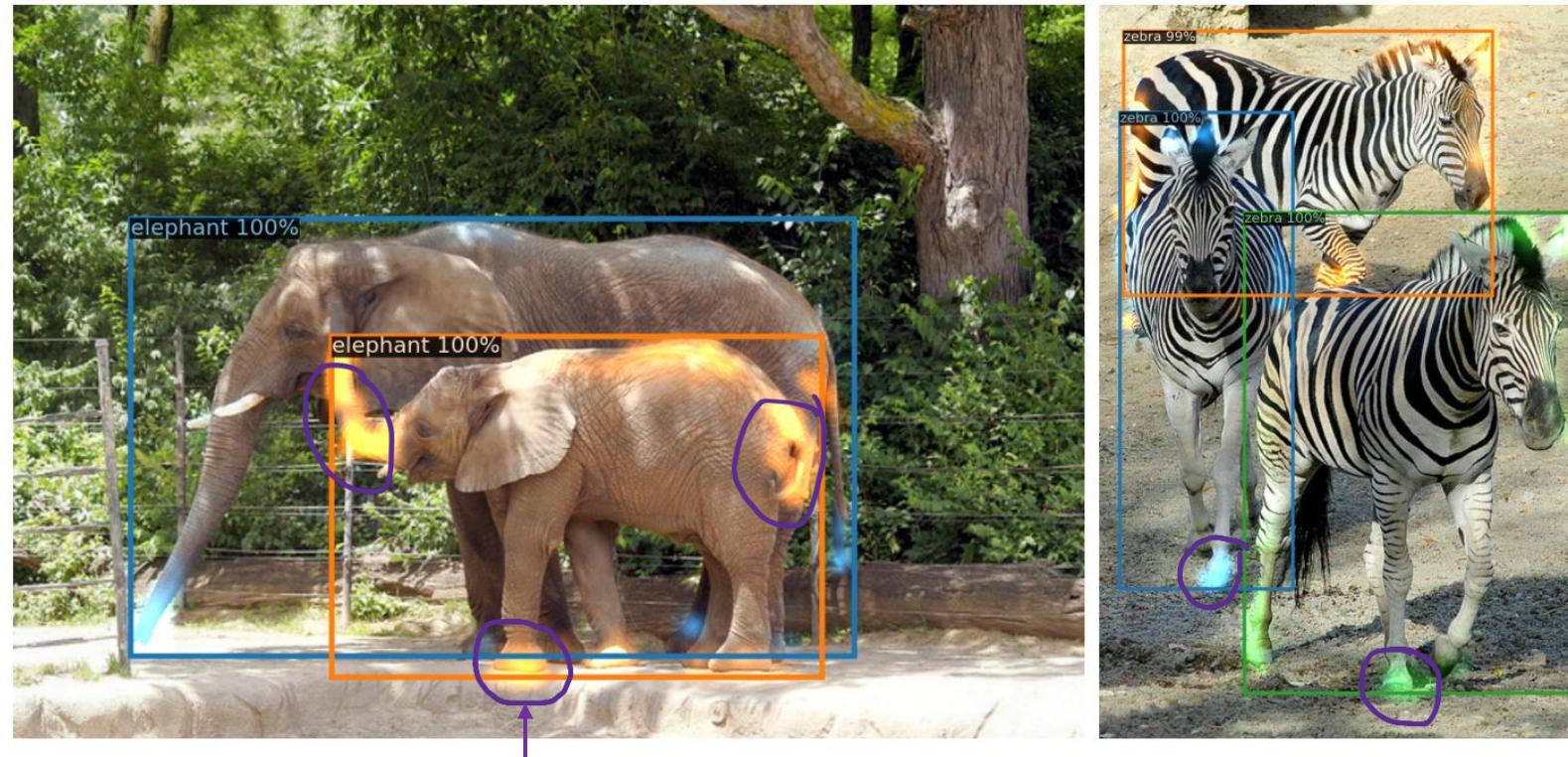


2.2 핵심 아이디어: Transformer (Decoder)

Decoder

- Since the decoder is also **permutation-invariant**, the N **input** embeddings must be **different** to produce **different results(물체 예측)** (순서가 있다면 입력 순서에 따라 출력을 구분할 수 있음)
--> N개의 object query(학습된 위치 임베딩)를 초기 입력으로 이용
- 인코더가 global attention을 통해 인스턴스를 분리한 뒤에 디코더는 각 인스턴스의 클래스와 경계선을 추출

- 각 인스턴스에서 **말단 (Extremities)** 부분의 Attention Score 값이 높게 형성
- 각각의 **경계선**을 잘 구분할 수 있도록 만들어 주는 요인, bbox가 잘 쳐지도록 decoder가 학습

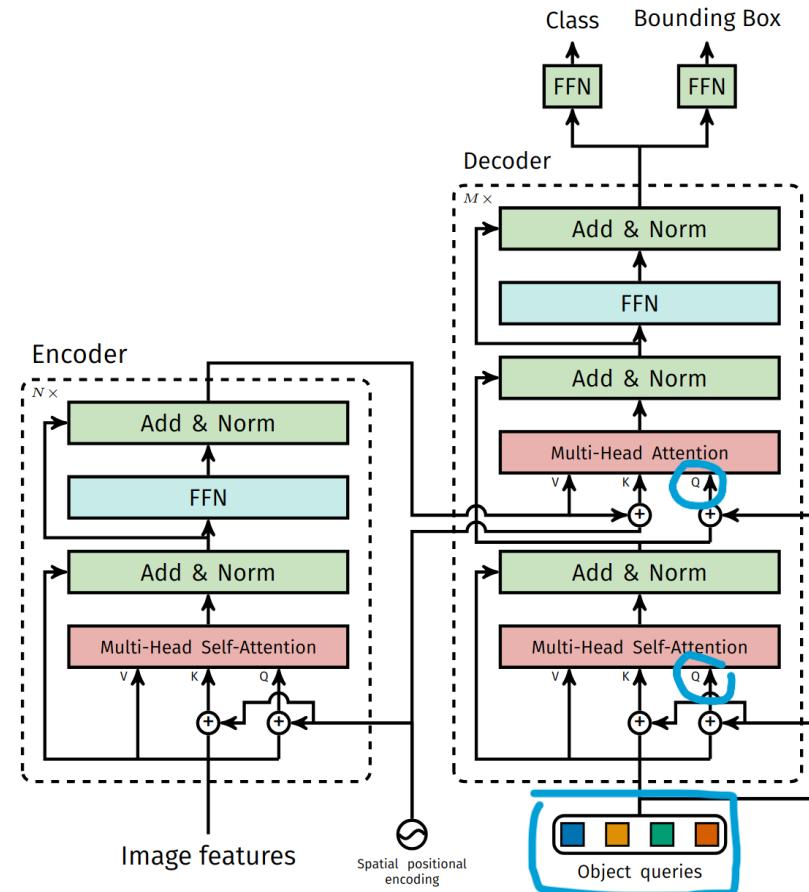


[Visualizing decoder attention for every predicted object]

2.2 핵심 아이디어: Transformer (Decoder)

Decoder

- Using **self- and encoder-decoder attention(cross attention)** over these embeddings, the model **globally reasons** about all objects together using pair-wise relations between them, while being able to use the **whole image as context**.
- Enc는 position에 따라 픽셀끼리 attention & dec도 N개의 query가 attention하기 때문에 instance가 겹치지 않음. 각 query가 서로 정보를 주고 받으면서 관심 가지는 영역이 분리 가능
- Query 자체가 개별적으로 다 다른 학습된 값들을 가짐(**initially set to zero**). 매 layer마다 enc로부터 attention을 거쳐서 전체 query 가 마지막 decoder 과정까지 거칠 때 각 query가 디테일한 데이터들을 잡을 수 있음
 - Queries + object queries(output positional encodings)**
 - 처음에는 아무 정보도 없는(random-initialized?) 벡터들이고, 학습을 통해 의미 있는 object detector로 발전하게 됨
- Decoder layer도 적게 사용하면 instance 분리가 잘 안됨. 따라서 **dec layer 도 적절히 많은 layer를 구성해야 함.** N개의 query 또한 서로 self-attention을 통해 feature를 구분하는 능력을 키워서 전체 네트워크가 잘 동작하도록 함.
- These input embeddings are learnt positional encodings that we refer to as object queries, and similarly to the encoder, we add them to the input of each attention layer.
- "**The first self-attention layer in the first decoder layer can be skipped.**"
 - DETR의 디코더는 학습 가능한 object queries를 입력으로 받음. 이 object query들은 처음에는 아무 정보도 없는 (즉, random initialized?) 벡터들이고, 학습을 통해 의미 있는 object detector로 발전하게 됨.
 - 그런데 디코더의 첫 번째 layer는 object queries가 처음으로 transformer에 들어가는 지점이기 때문에, 이 시점에서는 self-attention을 해봤자 의미 있는 관계를 찾기 어려움.
 - 모든 object query에 의미 있는 정보가 아직 없기 때문.
 - Query 자체도 **initially set to zero**
 - 그래서 논문에서는 첫 번째 디코더 layer의 self-attention은 생략해도 학습에 큰 영향을 주지 않거나, 오히려 효율적으로 작동할 수 있다고 말하고 있음

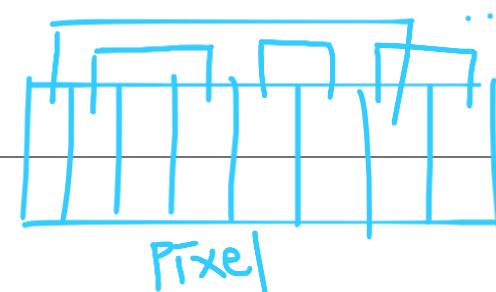


2.3 DETR architecture 정리

| Backbone-enc-dec-prediction heads

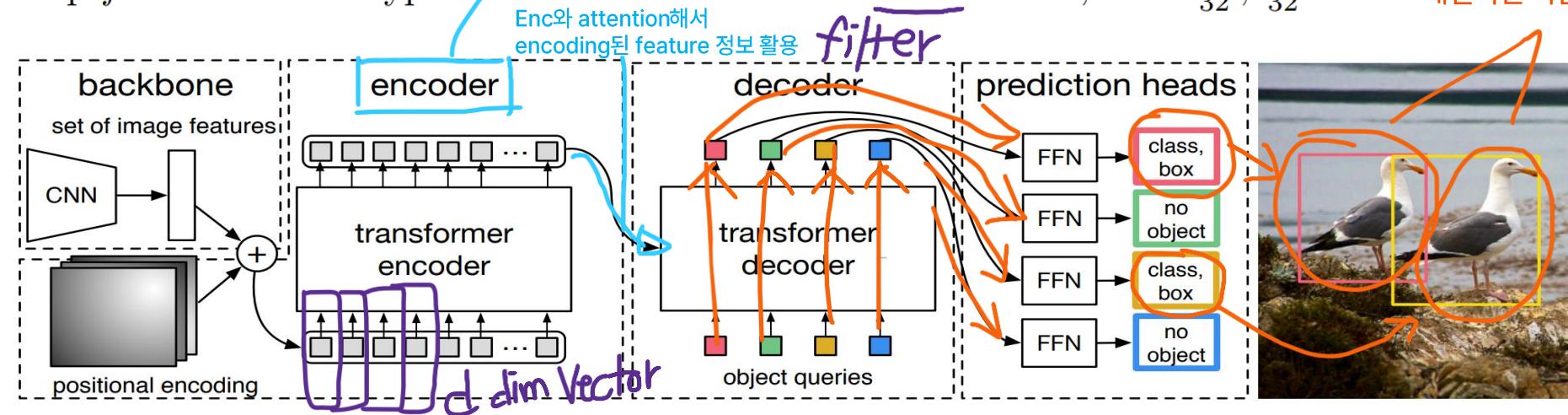
Backbone. Starting from the initial image $x_{\text{img}} \in \mathbb{R}^{3 \times H_0 \times W_0}$ (with 3 color channels²), a conventional CNN backbone generates a lower-resolution activation map $f \in \mathbb{R}^{C \times H \times W}$. Typical values we use are $C = 2048$ and $H, W = \frac{H_0}{32}, \frac{W_0}{32}$.

Self-attention

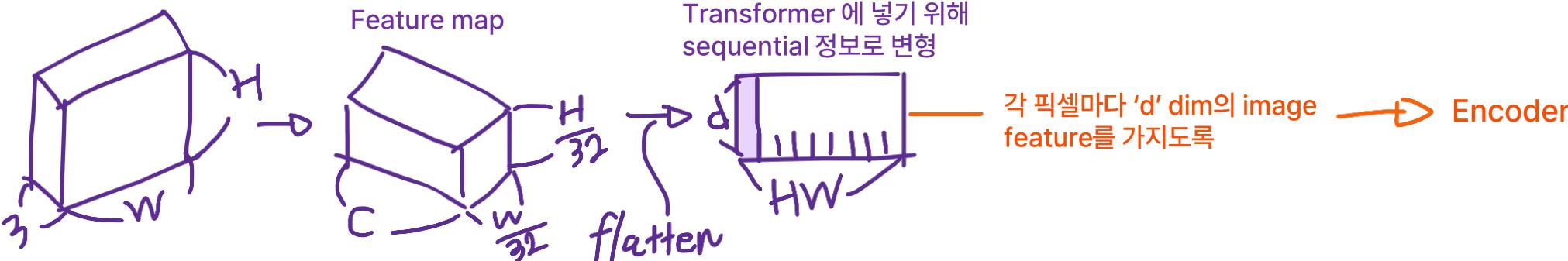


각 픽셀은 서로에 대한 attention score 계산
→ 이미지의 어떤 위치의 픽셀이 다른 픽셀들과 어떤 연관성(attention)을 가지는지 계산

서로 다른 instance: 클래스가 같더라도 개별적인 사물



Using self- and encoder-decoder attention over these embeddings, the model globally reasons about all objects together using pair-wise relations between them, while being able to use the whole image as context.



Chapter III

실험 결과 및 한계

실험 결과

Comparison with Faster R-CNN

Table 1: Comparison with Faster R-CNN with a ResNet-50 and ResNet-101 backbones on the COCO validation set. The top section shows results for Faster R-CNN models in Detectron2 [50], the middle section shows results for Faster R-CNN models with GIoU [38], random crops train-time augmentation, and the long 9x training schedule. DETR models achieve comparable results to heavily tuned Faster R-CNN baselines, having lower AP_S but greatly improved AP_L. We use torchscript Faster R-CNN and DETR models to measure FLOPS and FPS. Results without R101 in the name correspond to ResNet-50.

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	<u>42M</u>	42.0	62.1	45.5	<u>26.6</u>	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	<u>41M</u>	43.3	63.1	45.9	<u>22.5</u>	47.3	61.1
DETR-R101	152/20	<u>60M</u>	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

Training the baseline model for 300 epochs on 16 V100 GPUs takes 3 days, with 4 images per GPU (hence a total batch size of 64).

실험 결과

Ablations - Number of encoder layers

Table 2: Effect of encoder size. Each row corresponds to a model with varied number of encoder layers and fixed number of decoder layers. Performance gradually improves with more encoder layers.

#layers	GFLOPS/FPS	#params	AP	AP ₅₀	AP _S	AP _M	AP _L
0	76/28	33.4M	36.7	57.4	16.8	39.6	54.2
3	81/25	37.4M	40.1	60.6	18.5	43.8	58.6
6	86/23	41.3M	40.6	61.6	19.9	44.3	60.2
12	95/20	49.2M	41.6	62.1	19.8	44.9	61.9

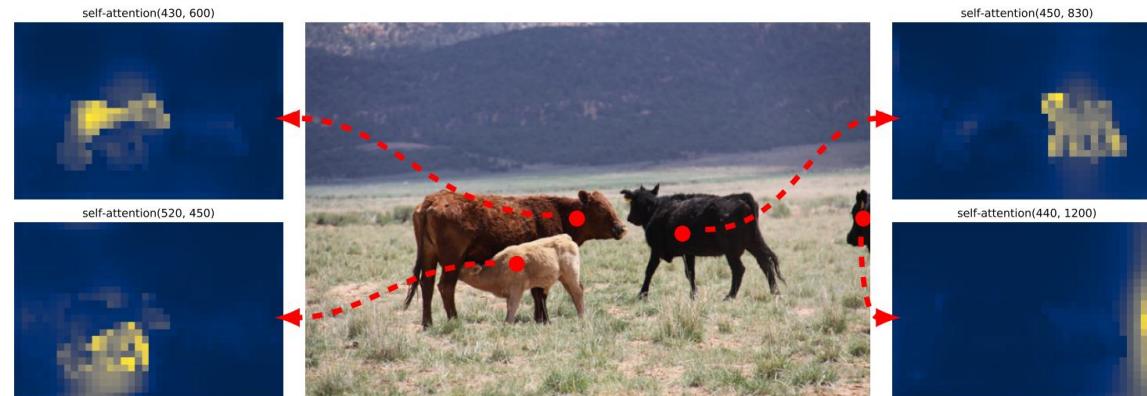


Fig. 3: Encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image.

We hypothesize that, by using global scene reasoning, the encoder is important for disentangling objects.

<attention maps of the last encoder layer of a trained model>

The encoder seems to separate instances already, which likely simplifies object extraction and localization for the decoder.

실험 결과

Ablations - Number of decoder layers

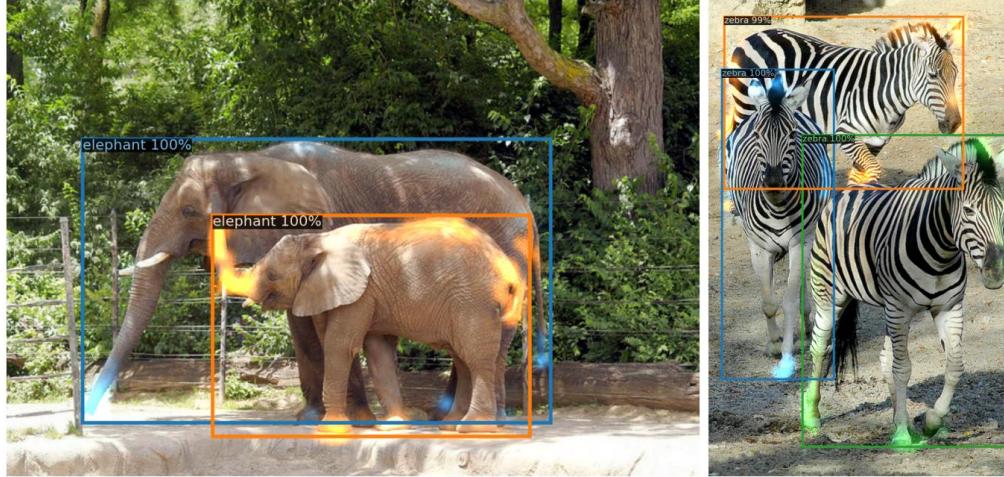
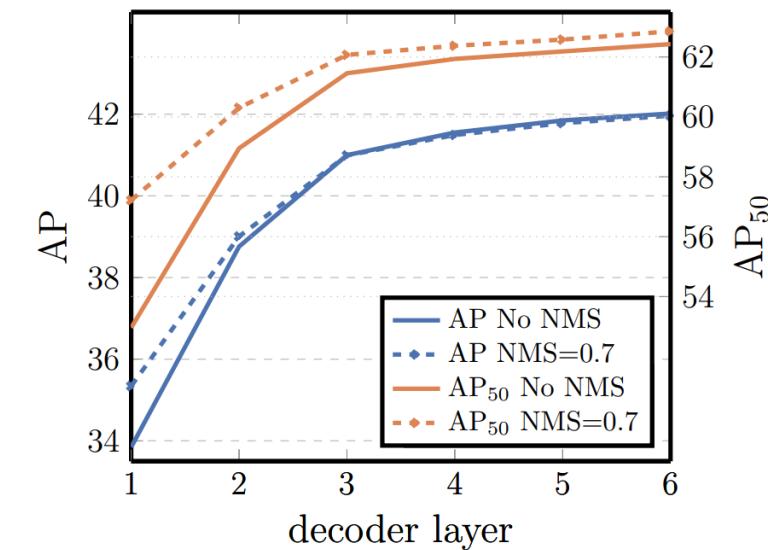


Fig. 6: Visualizing decoder attention for every predicted object (images from COCO val set). Predictions are made with DETR-DC5 model. Attention scores are coded with different colors for different objects. Decoder typically attends to object extremities, such as legs and heads. Best viewed in color.

We hypothesise that after the encoder has separated instances via global attention, the decoder only needs to attend to the extremities to extract the class and object boundaries.

This can be explained by the fact that a single decoding layer of the transformer is not able to compute any cross-correlations between the output elements, and thus it is prone to making multiple predictions for the same object. In the second and subsequent layers, the self-attention mechanism over the activations allows the model to inhibit duplicate predictions.

즉, enc, dec 모두 높은 capacity(표현력)이 있을수록 좋은 성능을 보임



실험 결과

Decoder output slot analysis

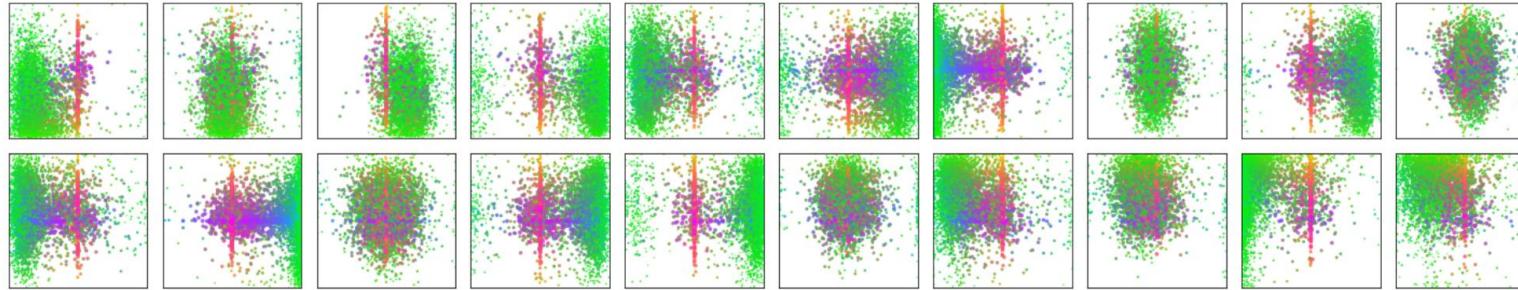


Fig. 7: Visualization of all box predictions on all images from COCO 2017 val set for 20 out of total $N = 100$ prediction slots in DETR decoder. Each box prediction is represented as a point with the coordinates of its center in the 1-by-1 square normalized by each image size. The points are color-coded so that green color corresponds to small boxes, red to large horizontal boxes and blue to large vertical boxes. We observe that each slot learns to specialize on certain areas and box sizes with several operating modes. We note that almost all slots have a mode of predicting large image-wide boxes that are common in COCO dataset.

DETR learns **different specialization for each query slot**. We observe that each slot has several modes of operation focusing on **different areas and box sizes**.

- Feature 값의 어느 부분에 가중치를 두어서 인식하는지에 대한 정보가 포함
- 인코더로부터 넘어온 인코딩 정보를 활용해서 N개의 서로 다른 instance를 구별할 수 있도록 디코딩이 이루어짐

Summary

Summary

- Given these ablations, we conclude that **transformer components**:
the global self-attention in encoder, FFN, multiple decoder layers, and positional encodings, **all significantly contribute to the final object detection performance**.
- 학습 시간이 매우 오래 걸림 (v100 gpu 여러 개 썼음에도,,)
- Transformer 입각한 아키텍쳐를 이용했을 때 Object detection 분야에서도 충분히 높은 성능을 보일 수 있다~!



Fig. 5: Out of distribution generalization for rare classes. Even though no image in the training set has more than 13 giraffes, DETR has no difficulty generalizing to 24 and more instances of the same class.

Summary

Summary

- 큰 object에 대해서는 높은 성능 but 작은 object에 대해서는 낮은 성능
 - This new design for detectors also comes with **new challenges**, in particular regarding training, optimization and performances on **small objects**. Current detectors required several years of improvements to cope with similar issues, and we expect future work to successfully address them for DETR.
 - In addition, it achieves significantly **better performance on large objects** than Faster R-CNN, likely thanks to the **processing of global information performed by the self-attention**.
 - Transformer 구조 상 attention의 global context는 잘 포착하지만 local 영역에 대한 특징은 추출하기 어렵기 때문에
 - CNN(backbone, 예시로 ResNet-50/101)을 사용해 feature map을 뽑을 때, 가장 마지막 stage, 즉 **가장 마지막 feature map**(C5 출력을 1×1 conv로 projection한 것)을 Transformer의 입력으로 사용. C5는 일반적으로 channel이 2048개이고, 해상도는 입력 대비 $1/32$ 로 줄어든 map
--> 따라서 오히려 global한 영역을 보는데 특화. input feature map도 low resolution이니까 작은 물체 검출은 더욱 힘든 상황

Summary

Deformable DETR

- CNN기반 detection 모델은 locality 특성 상 이미지의 큰 물체는 반영하기 어려움
- DETR은 반대로 Attention 특성 상 global한 특징을 잘 보기 때문에 큰 물체에 대한 성능은 높지만 작은 물체에 대한 성능은 낮음, 또한 [한 스케일 정보에 대해서만 global attention 연산](#)을 수행하므로 더욱 작은 물체에 대한 성능이 낮아진다고 볼 수 있음
--> 저자들도 fpn 같은 구조를 사용하면 개선할 수 있을 것이라고 함.
- Deformable DETR : attention 연산을 수행할 때 key가 모든 픽셀이 되는 것이 아니라, 특정 layer를 통해 예측된 sampling points들에 대해서만 attention 연산을 수행 (=Deformable convolution)
-

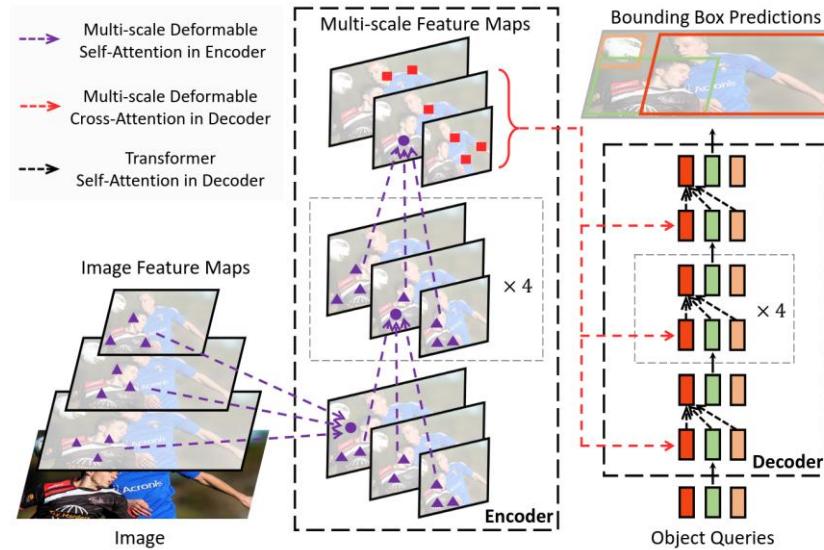


Figure 1: Illustration of the proposed Deformable DETR object detector.

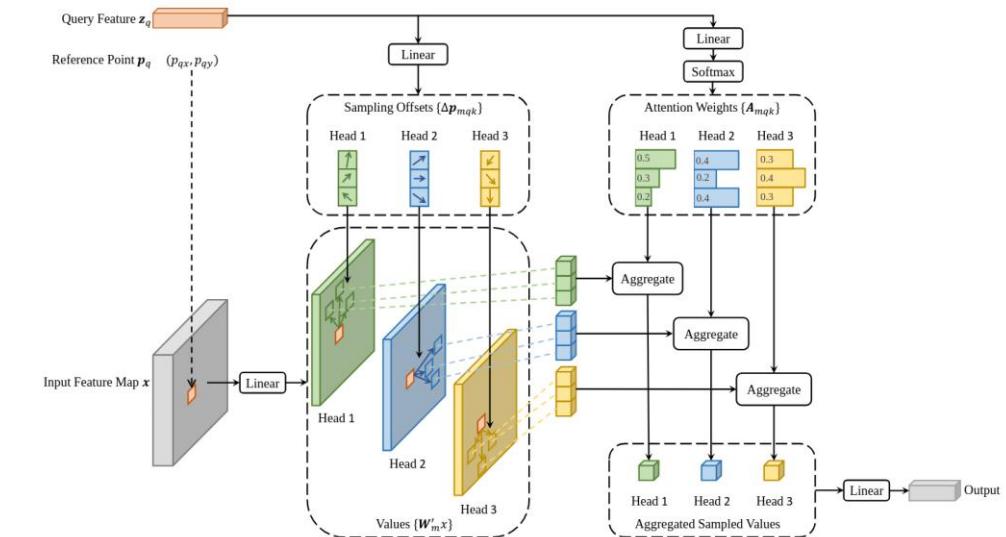


Figure 2: Illustration of the proposed deformable attention module.

Chapter IV

Questions

ViT와 DETR

ViT와 DETR

- ViT와 DETR 모두 트랜스포머를 사용하지만, 설계 목적은 다르다.
- ViT는 이미지 패치 간의 관계를 self-attention으로 학습하여 이미지 분류를 수행하며,
- DETR은 트랜스포머를 이용해 **바운딩 박스(boxes)**를 **세트 예측(set prediction)**하는 데 사용된다.
- DETR의 트랜스포머는 3가지 attention으로 구성된다:
 - 인코더의 self-attention: ViT와 유사하게 글로벌 관계를 학습함.
 - 디코더의 self-attention: 중복 바운딩 박스를 억제(NMS-like) 하기 위한 메커니즘.
 - 디코더의 cross-attention: 특징 맵과 쿼리 간의 관계를 학습하여 위치 추정을 수행.
- 따라서, 기존의 백본 기반 pre-training task([10], [11], [37])는 DETR의 트랜스포머에 직접 적용하기 어렵다.
- ---> UP-DETR 논문에 나오는 내용임.

Reference

Reference

- DETR 유튜브 강의 : <https://www.youtube.com/watch?v=hCWUTvVrG7E&t=63s>
- DETR Paper : <https://arxiv.org/pdf/2005.12872.pdf>
- https://github.com/ndb796/Deep-Learning-Paper-Review-and-Practice/blob/master/lecture_notes/DETR.pdf
- <https://kimjy99.github.io/%EB%85%BC%EB%AC%B8%EB%A6%AC%EB%B7%B0/detr/>