Senseable AI Lab

End-to-End Object Detection with Transformers

Nicolas Carion , Francisco Massa , Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko
Facebook AI

Dept. of AI and Bigdata, SOON CHUN HYANG Univ.

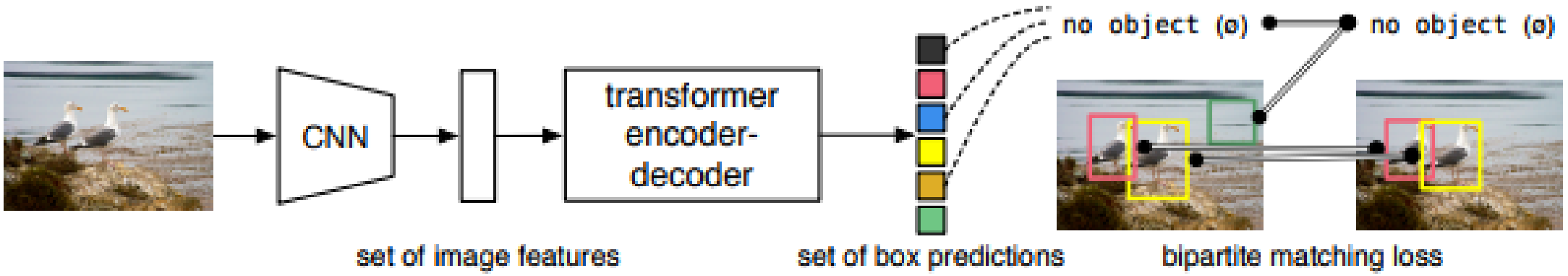
SENSEABLE AI LAB(SAIL)

민현식
minun001@sch.ac.kr

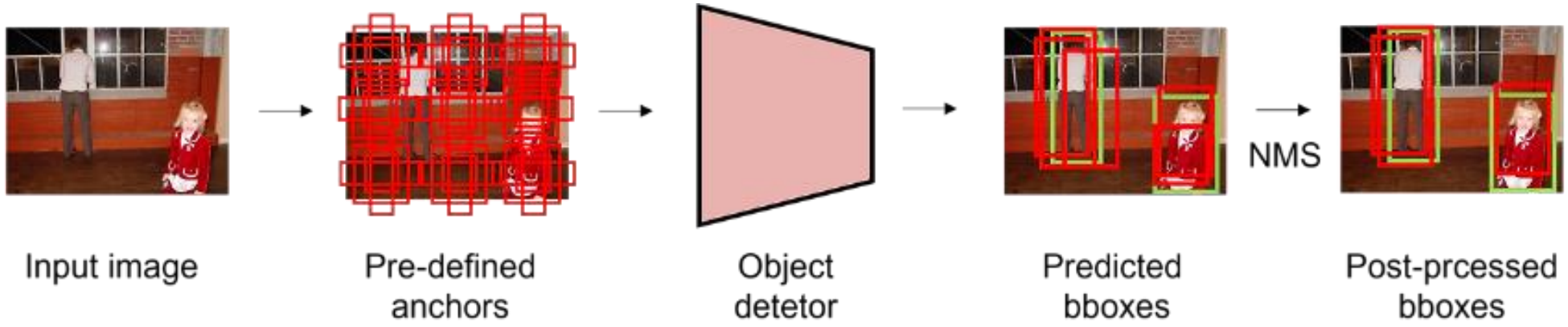
목차

1. Introduction

- 본 논문에서는 object detection을 bounding box와 category라는 set G={(B0,C0),(B1,C1),...,(Bn,Cn)}을 예측하는 task로 정의함
- 이 때 기존의 object detection 방법은 직접적으로 set을 예측하는 것이 아니라, 다수의 proposal, anchor, window center 등을 기반으로 set을 찾는 간접적인 방법에 기반을 두고 있음
- 반면 본 논문에서는 이러한 pipeline을 간소화하기 위해 set을 직접적으로 예측하는 접근법을 제안함.



1. Introduction

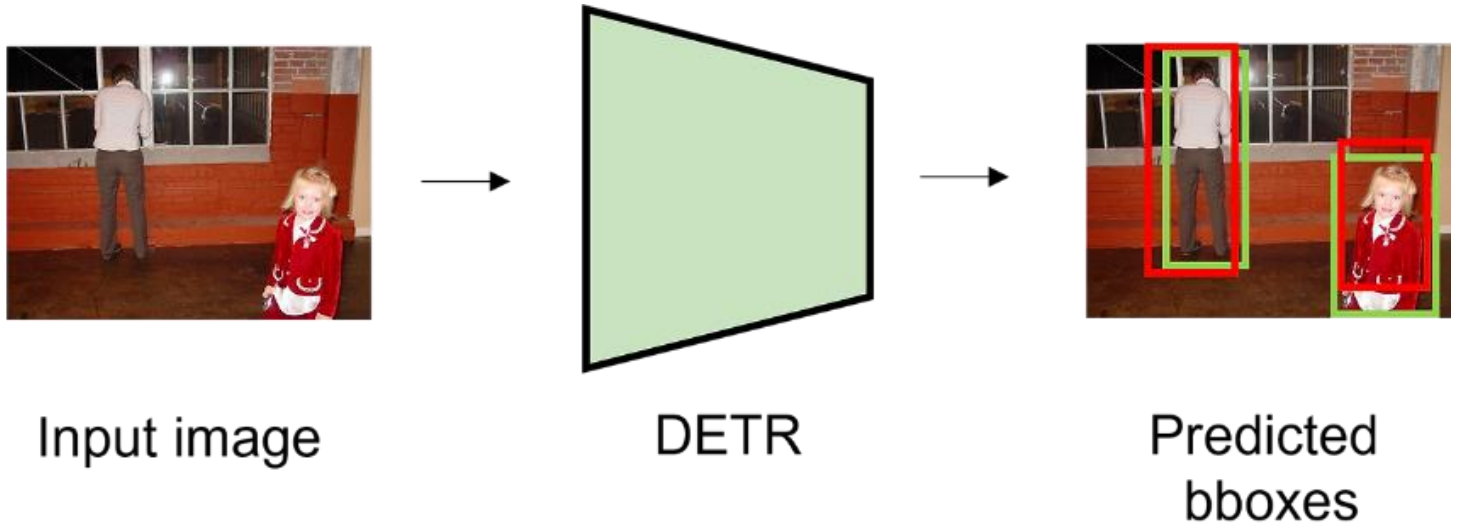


기존의 object detector의 detection 과정

- 기존 object detection 방법론은 pre-defined anchor를 사용함
- 이미지 내 고정된 지점마다 다양한 scale, aspect ratio를 가진 anchor를 생성함
- 이후 anchor를 기반으로 생성한 예측 bounding box와 ground truth를 match함
- 매칭 시, ground truth와의 IoU 값이 특정 임계값 이상일 경우 positive sample로, 이하일 경우 negative sample로 간주하며, positive sample에 대해서만 bounding box regression을 수행함
- 이처럼 임계값을 기준으로 독립적으로 예측을 수행하기 때문에 하나의 ground truth에 다수의 bounding box가 매칭됨
- 이로 인해 예측한 bounding box와 ground truth의 관계가 many-to-one이 됨

1. Introduction

- 기존 방법에서는 다양한 크기와 형태의 객체를 효과적으로 포착하기 위해 여러 개의 앵커를 생성함
- 그러나 하나의 ground truth를 예측하는 데 다수의 경계 상자(bounding box)가 존재하기 때문에, 중복된 예측을 제거하기 위해 NMS(Non Maximum Suppression)와 같은 후처리 과정이 필수적임



DETR의 detection 과정

- DETR은 여러 크기의 직접 정의된 앵커를 사용하지 않고, 하나의 ground truth 에 대해 하나의 예측된 경계 상자만 매칭함
- 이를 통해 예측된 경계 상자와 ground truth 사이의 관계가 일대일이 됨
- 하나의 ground truth 를 예측하는 경계 상자가 오직 하나만 있기 때문에 중복된 경계 상자가 없으며, 따라서 후처리 과정이 필요하지 않음

Contribution

본 논문에서 주장하는 contribution은 다음과 같음

1. 본 논문에서는 object detection을 direct set prediction으로 정의하여, transformer와 bipartite matching loss를 사용한**DETR(Detection with Transformer)**을 제안함

2. DETR은 COCO dataset에 대하여 **Faster R-CNN과 비슷한 수준의 성능**을 보임

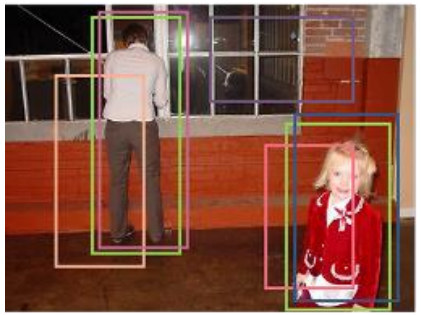
3. 추가적으로, self-attention을 통한 global information(전역 정보)를 활용함으로써 **크기가 큰 객체를 Faster R-CNN보다 훨씬 잘 포착**함

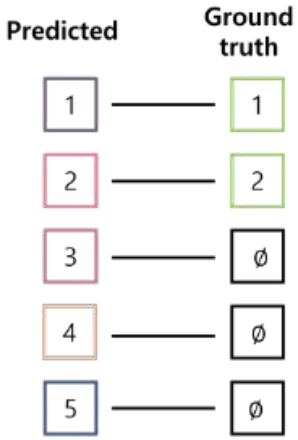
Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster 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**

2. Background knowledge

1. Hungarian algorithm

- DETR은 예측된 bounding box와 실제 객체를 일대일로 매칭함
- Loss를 최소화할 수 있는 매칭을 찾기 위해 가능한 모든 조합 경우의 수를 고려하는 brute force 방법을 사용하면 지나치게 많은 시간이 걸린다는 문제가 있음
- 따라서 본 논문에서는 **Hungarian algorithm**을 사용함
- **Hungarian algorithm은 두 집합 사이의 일대일 대응 시 가장 비용이 적게 드는 bipartite matching(이분 매칭)을 찾는 알고리즘임**
- Hungarian algorithm은 어떠한 집합 I와 matching 대상인 집합 J가 있으며, i$\in I$를 j$\in J$에 matching하는데 드는 비용을 $c(i,j)$라고 할 때, $\sigma: I \rightarrow J$로의 일대일 대응 중에서 **가장 적은 cost가 드는 matching에 대한 순열 σ을 찾는 것임**
- 여기서 말하는 순열은 매칭 시 최적의 순서에 대한 index를 의미함
- Hungarian algorith은 I,J에 대한 cost를 표현한 행렬에 대하여 동작함

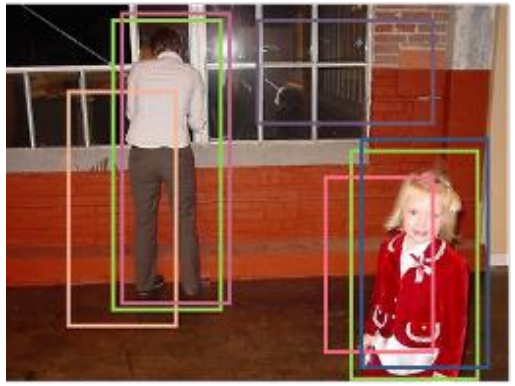


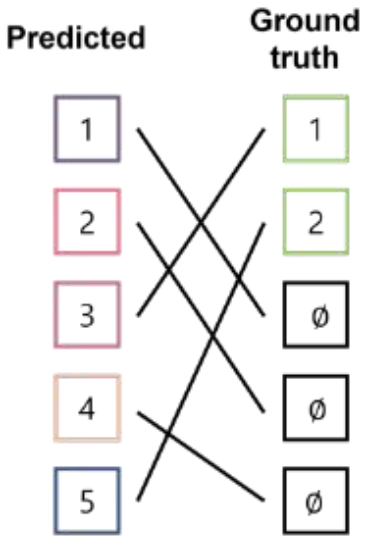


	1	2	ø	ø	ø
1	12	11	1	1	1
2	4	2	8	6	9
3	1	3	5	7	8
4	2	5	6	7	9
5	2	1	9	10	6

Permutation = [1, 2, 3, 4, 5]
Matching score = 12 + 2 + 5 + 7 + 6 = 32

- 이미지 내 두 개의 객체(두 명의 사람)가 있을 때, 예측된 bounding box에서 ground truth와 매칭을 하는 문제
- 이 때 cost에 대한 행렬은 위 그림과 같음
- 행렬의 행은 예측된 bounding box를, 열은 ground truth 를 의미하며, 행렬의 각 요소는 예측된 bounding box가 ground truth로 매칭되었을 때의 cost를 의미함
- 1번 예측된 bounding box가 2번 ground truth로 매칭되었을 때의 cost는 11
- 순열이 [1, 2, 3, 4, 5]인 경우, 즉 1번 bounding box가 1번 ground truth로 matching되고, 2번 bounding box가 2번 ground truth에 매칭되고...인 경우, cost가 32인 것을 확인할 수 있음





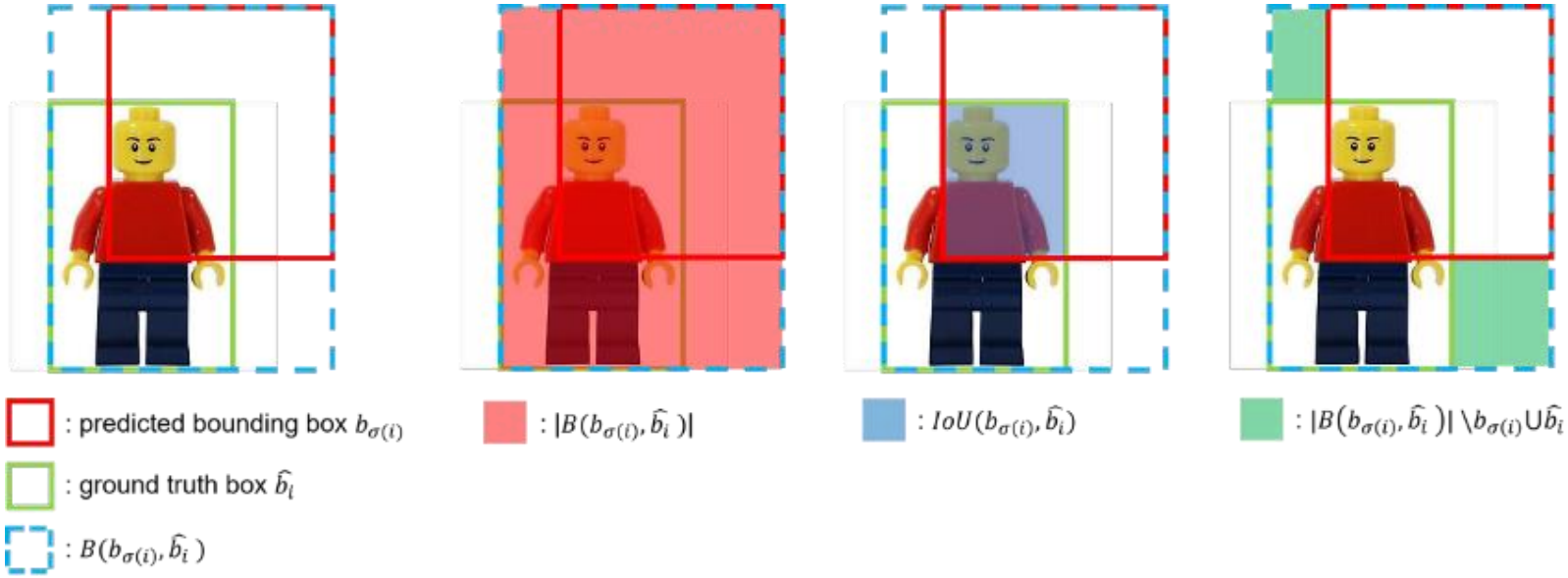
	1	2	ø	ø	ø
1	12	11	1	1	1
2	4	2	8	5	9
3	1	3	5	7	8
4	2	5	6	7	4
5	2	1	9	10	6

Permutation = [3, 4, 1, 5, 2]
Matching score = 1 + 5 + 1 + 4 + 1 = 12

- 순열이 [3, 4, 1, 5, 2]인 경우, cost가 12로 상대적으로 매우 낮으며 가장 바람직하게 matching된 것을 확인할 수 있음
- **Hungarian algorithm은 이처럼 cost에 대한 행렬을 입력 받아, matching cost가 최소인 순열을 출력함**

2. Bounding box loss

- 기존의 방법들은 앵커를 기준으로 경계 상자(bounding box)를 예측하기 때문에 예측된 경계 상자의 범위가 크게 벗어나지 않음
- 반면, DETR은 초기 추정값 없이 경계 상자를 예측하기 때문에 예측되는 값의 범위가 상대적으로 넓음
- 이 때문에 L1 loss만을 사용하면, 상대적인 오류는 비슷할 수 있지만, 크기가 큰 상자와 작은 상자는 서로 다른 범위의 손실을 가지게 됨(큰 상자는 큰 손실을, 작은 상자는 작은 손실을 가짐).
- 이 문제를 해결하기 위해, 본 논문에서는 L1 loss와 일반화된 IoU 손실인 generalized IoU(GIoU) loss를 함께 사용함



$$GIoU = IoU(b_{\sigma(i)}, \hat{b}) - \frac{|B(b_{\sigma(i)}, \hat{b})| \setminus b_{\sigma(i)} \cup \hat{b_i}}{|B(b_{\sigma(i)}, \hat{b})|}$$

$$\mathcal{L}_{iou}(b_{\sigma(i)}, \hat{b}) = 1 - GIoU$$

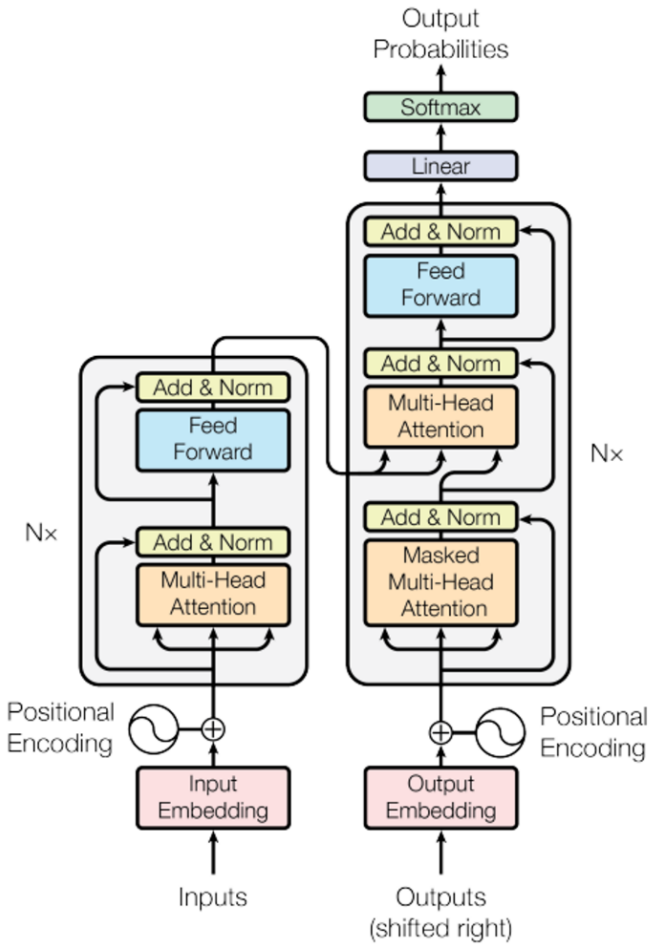
- **Generalized IoU(*GIoU*) loss**: 두 box 사이의 IoU(Intersection over Union) 값을 활용한 loss로 **scale을 동일하게 만든다는** 특징이 있음

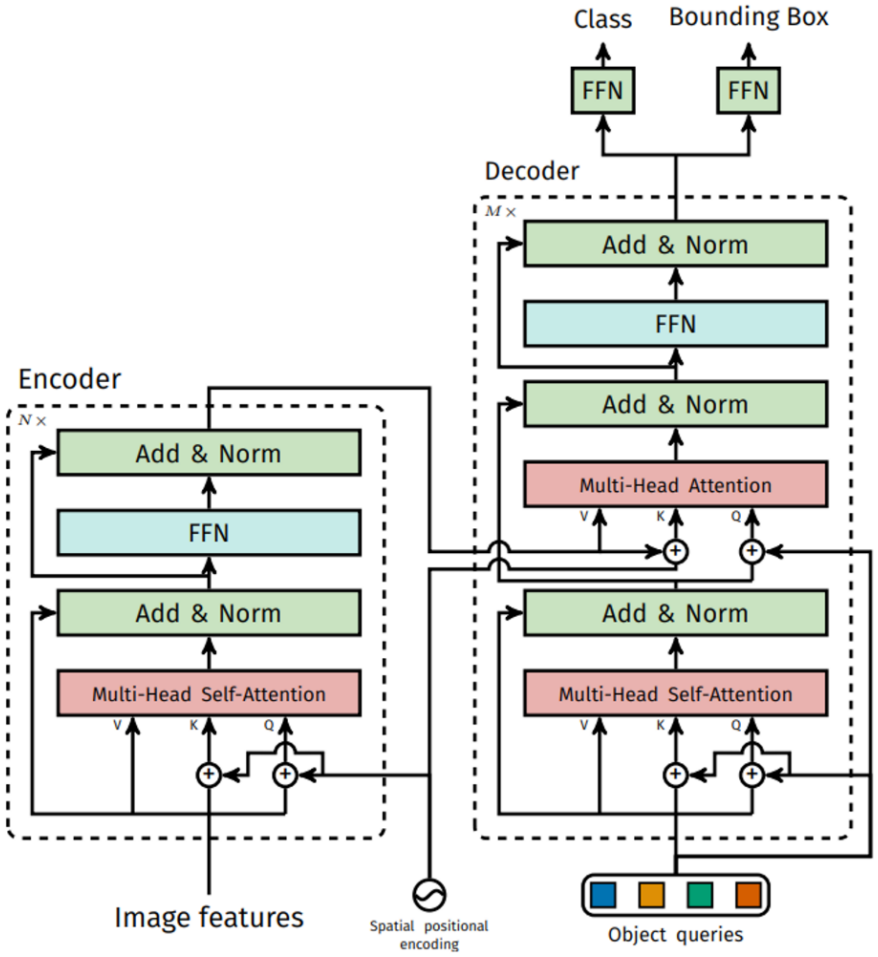
- $GIoU$ 를 구하기 위해서는 predicted box $b_{\sigma(i)}$와 ground truth box $\hat{b}_i$를 둘러싸는 가장 작은 box $B(b_{\sigma(i)}, \hat{b})$를 구함
- 이때 predicted box와 ground truth가 많이 겹칠 수록 $B(b_{\sigma(i)}, \hat{b})$가 작아지며, 두 box가 멀어질 수록 $B(b_{\sigma(i)}, \hat{b})$가 커짐
- $IoU(b_{\sigma(i)}, \hat{b})$는 두 box 사이의 IoU를 의미하며 $\frac{|B(b_{\sigma(i)}, \hat{b})| \setminus b_{\sigma(i)} \cup \hat{b_i}}{|B(b_{\sigma(i)}, \hat{b})|}$는 $B(b_{\sigma(i)}, \hat{b})$에서 predicted box와 ground truth를 합한 영역을 뺀 영역에 해당함
- $GIoU$는 -1~1 사이의 값을 가지며, $GIoU$ 를 loss로 사용할 때 $1-GIoU$ 형태로 사용하여 loss의 최대값은 2, 최소값은 0이 됨

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

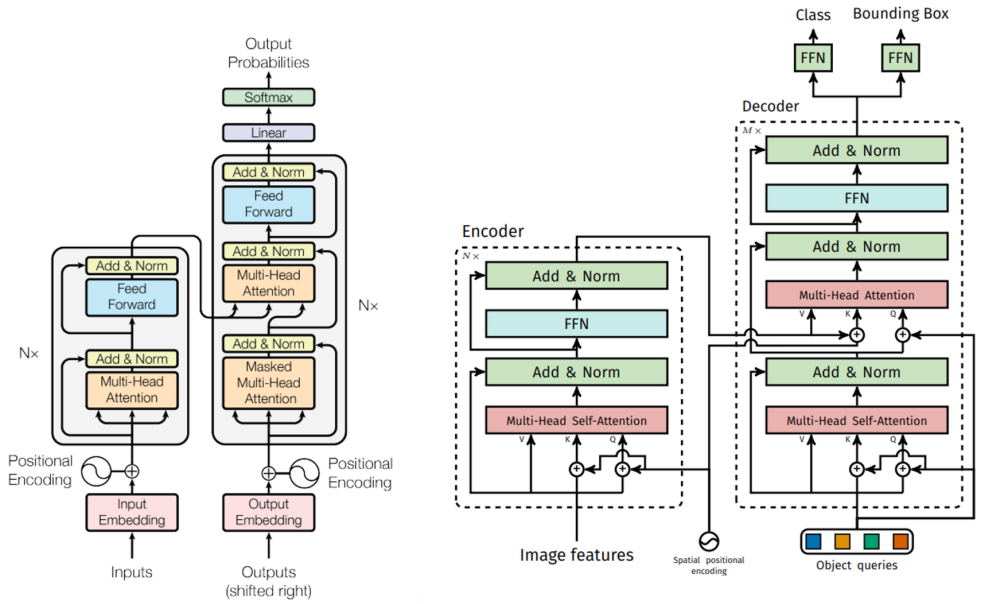
- L1 loss와 $GIoU$ loss를 사용한 전체 bounding box loss
- λ_{iou}, λ_{L1}는 두 term 사이를 조정하는 scalar hyperparameter
- 두 loss는 mini-batch 내 객체의 수에 따라 normalize됨

3. Transformer for NLP task vs DETR Transformer



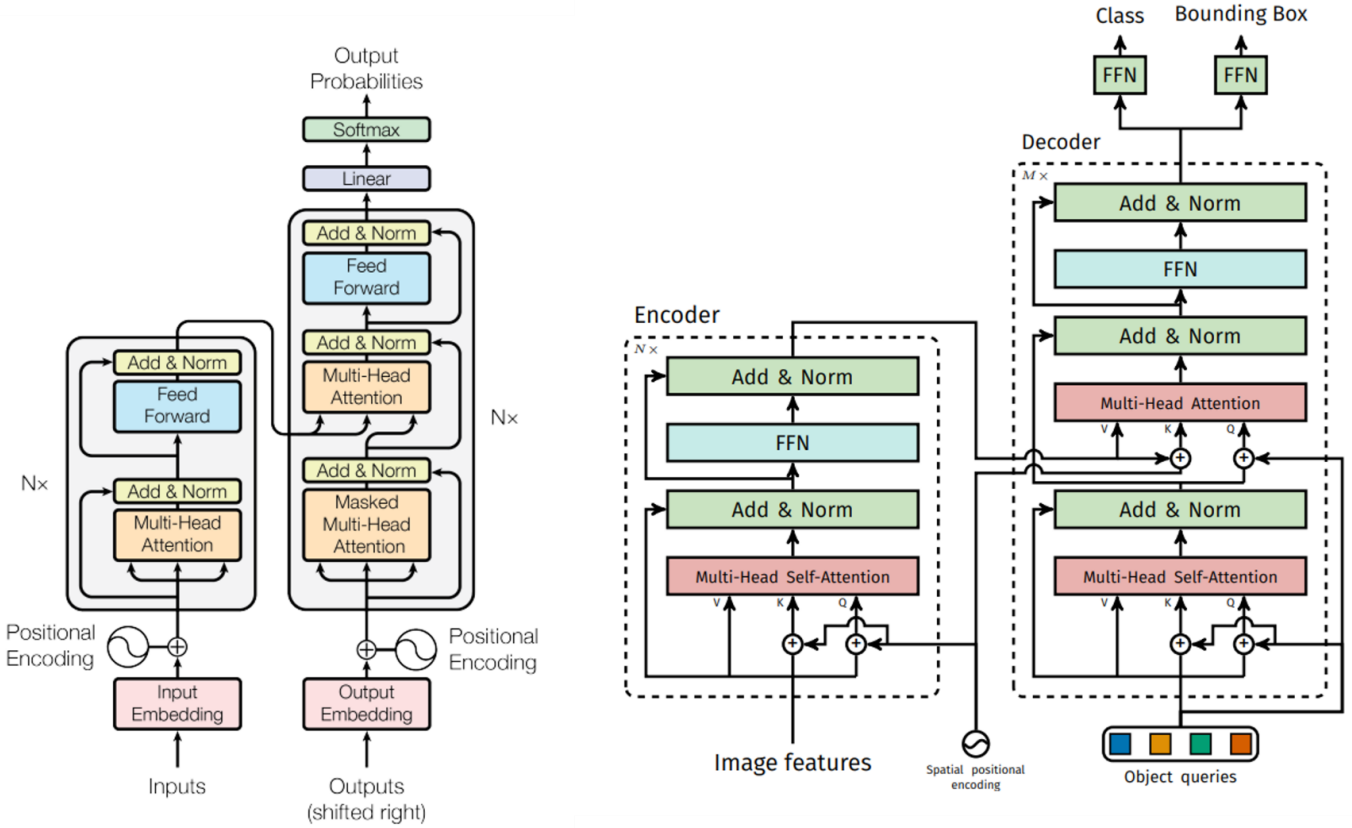


- 앞서 Hungarian algorithm 파트에서 살펴보았듯이 DETR은 반복되는 prediction을 제거하기 위해 object detection task를 prediction bounding box 집합과 ground truth box 집합을 matching하는 set prediction으로 정의함
- DETR은 효과적인 matching을 위해 encoder-decoder 구조의 Transformer를 사용함
- Transformer의 self-attention은 모든 입력 sequence의 token 사이의 상호작용(pairwise interaction)을 모델링하기 때문에 set prediction에 적합함



하지만 DETR에서 사용하는 Transformer와 NLP task에서 사용하는 Transformer는 입출력 측면에서 차이가 있음

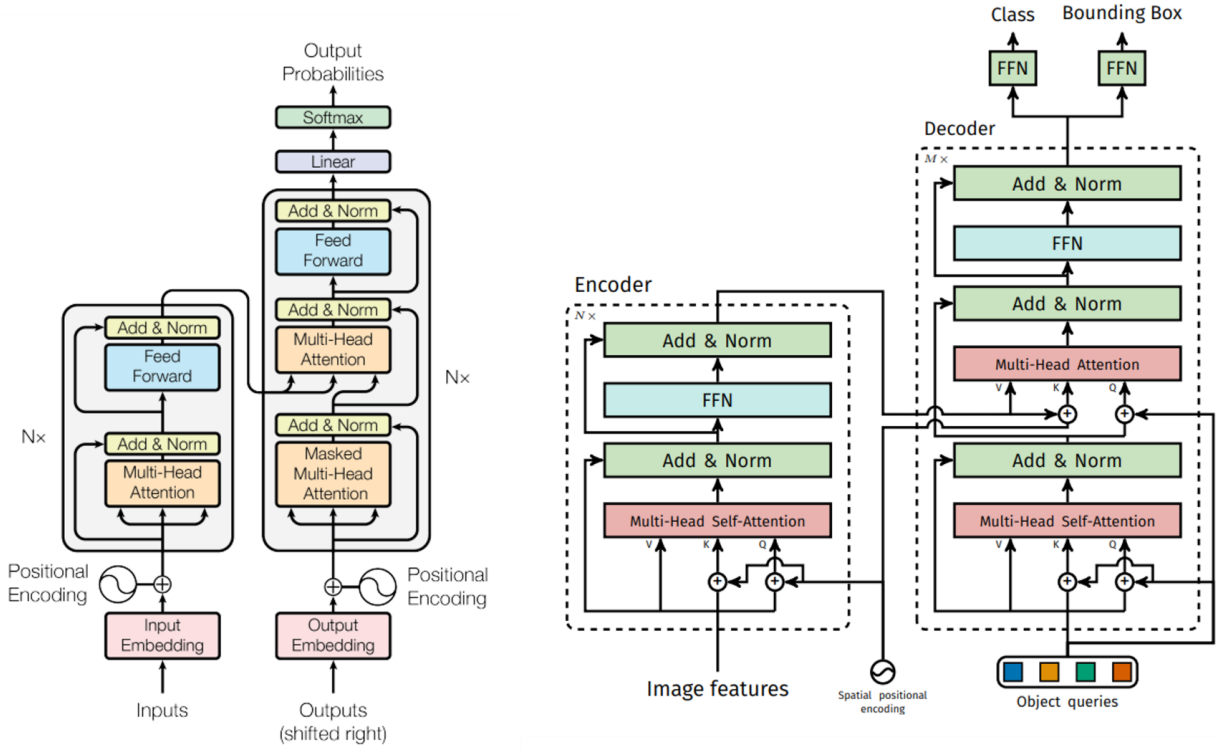
1. 첫 번째로 DETR는 encoder에서 이미지 feature map을 입력받는 반면, Transformer는 문장에 대한 embedding을 입력받음



- Transformer는 sequence 정보를 입력받는데 적합하기 때문에 DETR은 CNN backbone에서 feature map을 추출한 이후, 1x1 convolution layer를 거쳐 차원을 줄인 다음, spatial dimension을 flatten하여 encoder에 입력함

- h,w가 height, width이며, C가 channel 수, d가 C보다 작은 channel 수라고 할 때 $C \times h \times w$ 크기의 feature map을 $d \times hw$로 변환시킴

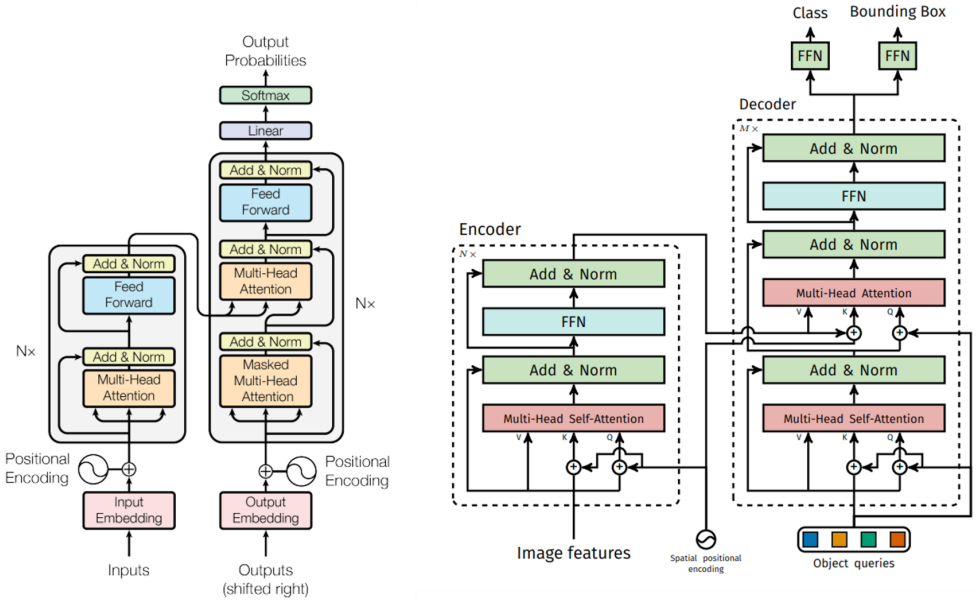
2. 두 번째로 positional encoding에서 차이



- Transformer는 입력 embedding의 순서와 상관 없이 동일한 값을 출력하는 순열 불변한 성질을 가졌기 때문에 positional encoding을 더해줌
- DETR은 x, y axis가 있는 2D 크기의 feature map을 입력받기 때문에 기존의 positional encoding을 2D 차원으로 일반화시켜 공간적인 positional encoding을 수행함
- 입력값의 차원이 d라고 할 때 x, y 차원에 대하여, row-wise, column wise로 $2d$ 크기로 sine, cosine 함수를 적용함
- 이후 channel-wise하게 concat하여 d channel의 공간적인 positional encoding을 얻은 후 입력값에 더해줌

Senseable AI Lab

3. Transformer는 디코더에 타겟 임베딩(target embedding)을 입력하는 반면, DETR은 객체 쿼리(object queries)를 입력함



- Transformer 모델에서는 번역 작업을 할 때, 디코더에 번역할 문장의 임베딩을 입력함. 이는 번역하고자 하는 문장을 이해하기 위한 정보라고 볼 수 있음
- 반면에, DETR 모델에서는 물체 감지 작업을 할 때, 디코더에 '객체 쿼리'라는 것을 입력함. 이 객체 쿼리는 모델이 이미지 속에서 특정한 물체를 찾도록 돕는 역할을 하는 정보임.
- 객체 쿼리는 길이가 정해져 있고, 학습 과정을 통해 모델이 스스로 학습할 수 있는 임베딩임. 쉽게 말해, 객체 쿼리는 " 어떤 종류의 물체를 찾아라 " 는 일종의 명령어 역할을 하는 것임.

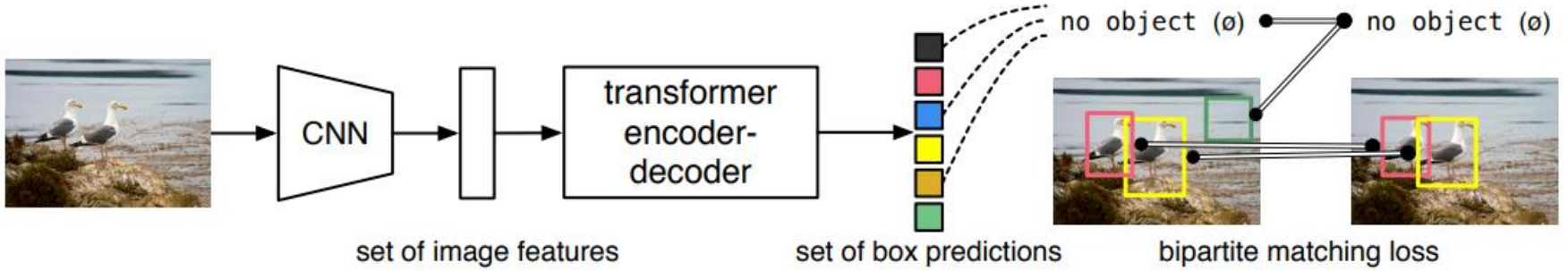
4. Transformer는 decoder에서 첫 번째 attention 연산 시 masked multi-head attention을 수행하는 반면, DETR은 multi-head self-attention을 수행함

- Transformer는 디코더에서 첫 번째 어텐션(attention) 연산 시 masked multi-head attention을 수행하는 반면, DETR은 multi-head self-attention을 수행함
- 이 차이는 Transformer가 자동 회귀적(auto-regressive) 방식으로 다음 토큰(token)을 예측하기 때문임. 자동 회귀적 방식은 다음 토큰을 예측할 때 현재까지의 정보만을 사용하여 다음 단어를 생성을 의미함
- 따라서, 어텐션 연산 시 다음 토큰에 대한 정보가 미리 반영되는 것을 방지하기 위해 후속 토큰 위치를 마스킹(masking)함. 이 마스킹은 어텐션 연산에서 사용하는 소프트맥스(softmax) 함수의 입력에 후속 토큰 위치에 -무한대(-inf) 값을 넣는 방식으로 이루어짐
- 반면, DETR은 입력된 이미지 내에서 모든 객체의 위치를 동시에 예측해야 하기 때문에 별도의 마스킹 과정이 필요 없음
- 이는 DETR이 이미지 전체를 한 번에 처리하여 객체의 위치를 파악하기 때문임
- 따라서 모든 위치 정보를 한꺼번에 고려할 수 있으므로, Transformer에서 사용하는 같은 마스킹 방법이 필요하지 않음

5. Transformer는 Decoder 이후 하나의 head를 가지는 반면, DETR는 두 개의 head를 가짐

- Transformer는 다음 token에 대한 class probability를 예측하기 때문에 하나의 linear layer를 가짐
- 반면, DETR은 이미지 내 객체의 bounding box와 class probability를 예측하기 때문에 각각을 예측하는 두 개의 linear layer를 가짐

3. Method



1.1 Find optimal matching

- 기존의 방법에서는 이미지에서 객체를 찾기 위해 수많은 앵커 박스를 생성하여 각 앵커 박스가 객체일 가능성을 평가함
- 이는 매우 많은 계산을 필요로 하며, 여러 앵커 박스 중 어떤 것이 실제 객체를 나타내는지 결정하는 것이 복잡할 수 있음
- 반면에, DETR은 고정된 수의 예측만을 수행함
- 예를 들어, N을 100으로 설정하면, DETR은 이미지 내에서 최대 100개의 객체를 예측함
- 이 수는 실제로 이미지에 존재하는 객체 수보다 많기 때문에, 모든 실제 객체를 커버할 수 있음
- 이렇게 하면 예측된 결과와 실제 객체를 매칭하는 과정이 더 간단해지고, 예측의 정확성도 높아질 수 있음
- y는 객체에 대한 ground truth set이며 $\hat{y}$는 N개의 prediction임
- 객체의 수를 제외한 나머지는 ⌀ (no object)로 채워짐
- 즉 이미지 내 객체의 수가 3개이면, y에서 97개는 ⌀ 로 채워짐. 이는 불필요한 예측을 줄이고, 이미지에서 실제로 중요한 객체만을 강조하기 위한 방법으로 사용됨
- 이 때 두 개의 set에 대하여 이분 매칭(bipartite matching)을 수행하기 위해, 아래의 cost를 minimiz할 수 있는 N의 순열을 탐색함

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{\text{match}}\left(y_i, \hat{y}_{\sigma(i)}\right)$$

Senseable AI Lab

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

σ : Ground truth의 object set의 순열(permutation)
$\hat{\sigma}$: L_{match}를 최소로 하는 예측 bounding box set의 순열
y: Ground truth의 object set & $\hat{y}$: 예측한 N개의 object set
c: class label
$p_{(c)}$: 해당 class에 속할 확률
b: bounding box의 위치와 크기 (x, y, w, h)

- $L_{match}(y_i, \hat{y}_{\sigma(i)})$는 ground truth인 y_i와 index가 $\hat{y}_{\sigma(i)}$인 예측값 사이의 pair-wise matching cost임(이 값이 낮을 수록 두개가 잘 매칭이 되었다는 의미)
-
- 이 matching cost는 class prediction과 predicted bounding box와 ground truth box 사이의 similarity(유사도)를 모두 고려함
- 기존 연구에서는 predicted objects와 ground truth objects 간의 거리(distance)를 계산하는 방법은 bounding box regression loss를 사용하였음
- 하지만 DETR의 loss는 **predicted bounding box와 ground truth bounding box 간의 거리를 계산하여, 이 거리가 작을수록 더 좋은 예측**이라고 판단함
- 이렇게 계산된 bipartite matching loss는 **predicted objects와 ground truth objects 간의 일대일 대응을 보장**하며, predicted objects의 순서에 불변함

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

여기서 **순서에 불변한다**는 의미는, **predicted objects를 출력할 때 그들의 순서가 중요하지 않다는 것**을 의미함
예를 들어, 이미지 내에 세 개의 객체가 있다고 가정해 보자. DETR은 이 세 개의 객체를 각각 object query로부터 예측함
이때, DETR은 predicted objects를 출력할 때, **predicted objects를 어떤 순서로 출력하더라도, 이들이 ground truth objects와 일대일 대응되는 한 bipartite matching loss를 최소화**할 수 있음
이러한 특징은 **predicted objects를 병렬로 출력**할 수 있도록 하며, 이는 **DETR의 빠른 속도와 높은 정확도를 보장하는 중요한 요소 중 하나**라고 볼 수 있음

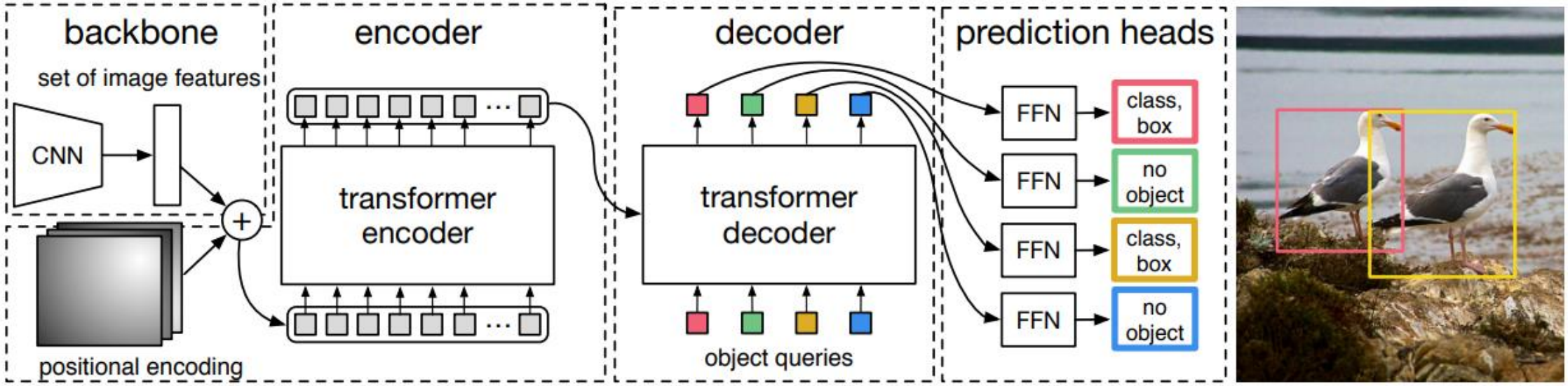
Senseable AI Lab

1.2 Compute Hungarian loss

- loss는 class loss와 box loss로 구성됨. Class loss는 prediction에 대한 negative log-likelihood를 구함
- Box loss는 Preliminaries에서 언급했듯이 l1 loss와 generalized IoU loss를 결합하여 사용함

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

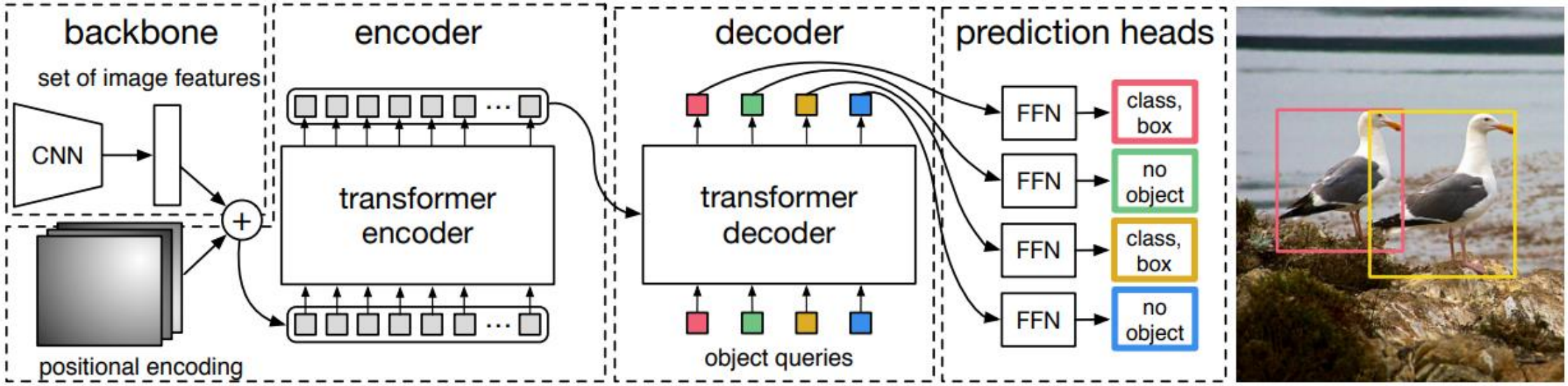
- $\hat{\sigma}(i)$는 이전 단계에서 구한 optimal assignment임
- 실제 학습 시 예측하는 객체가 없는 경우인 $c_i = \varnothing$에 대하여 log-probability를 1/10로 down-weigh한다고 함
- 이는 실제로 객체를 예측하지 않는 negative sample의 수가 매우 많아 class imbalance를 위해 해당 sample에 대한 영향을 감소시키기 위함임



DETR은 1) CNN backbone 2) Transformer encoder와 decoder 3) FFN(Feed Forward Network)로 구성되어 있음

1. Backbone

- 입력이미지 $x_{img} \in \mathbb{R}^{3\times H\times W}$를 CNN backbone network에 입력하여, feature map $f \in \mathbb{R}^{C\times h\times w}$를 생성함
- C = 2048, h, w = $\frac{H}{32}, \frac{W}{32}$



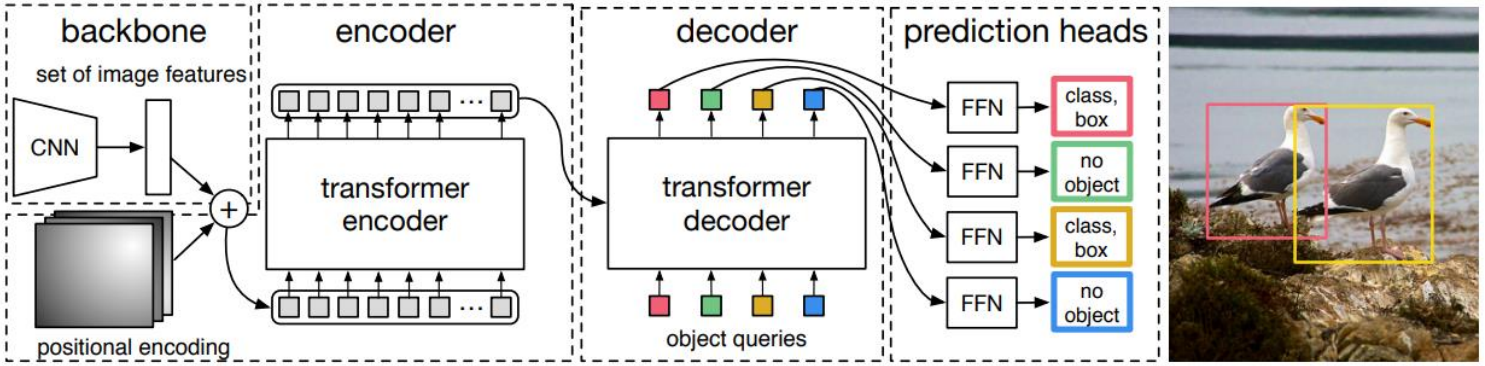
2. Transformer encoder

- 1x1 convolution 연산을 적용하여, C차원의 **feature map을 d 차원으로 감소**시켜 새로운 feature map $z_0 \in \mathbb{R}^{d \times h \times w}$을 생성함
- Transformer encoder는 sequence를 입력으로 받기 때문에 z_0의 공간 차원을 축소 시켜 크기를 $d \times hw$ 형태로 나타냄
- **각 encoder layer는 multi-head self-attention module과 feed forward network(FFN)으로 구성**되어 있음
- Transformer 구조는 입력 embedding의 순서와 상관없이 같은 출력값을 생성하는 permutation-invariant 속성이 있기 때문에 encoder layer 입력 전에 **입력 embedding에 positional encoding을 더해**줌

3. Transformer decoder

- Transformer decoder는 masking을 통해 다음 token을 예측하는 autoregressive 방법을 사용하는 반면, DETR decoder는 N개의 object에 대한 정보를 한번에 출력함

- Decoder 역시 permutation-invariant하기 때문에 입력으로 받는 embedding으로 object queries라고 불리는 learnt positional encoding을 사용함



- **object query는 object query feature과 object query positional embedding으로 구성**되어 있음

- object query feature는 decoder에 initial input으로 사용되어, decoder layer를 거치면서 학습됨
- query positional embedding은 decoder layer에서 attention 연산 시 모든 query feature에 더해짐
- Query feature는 학습 시작 시 0으로 초기화(zero-initialized)되며, query positional embedding은 학습 가능(learnable)함

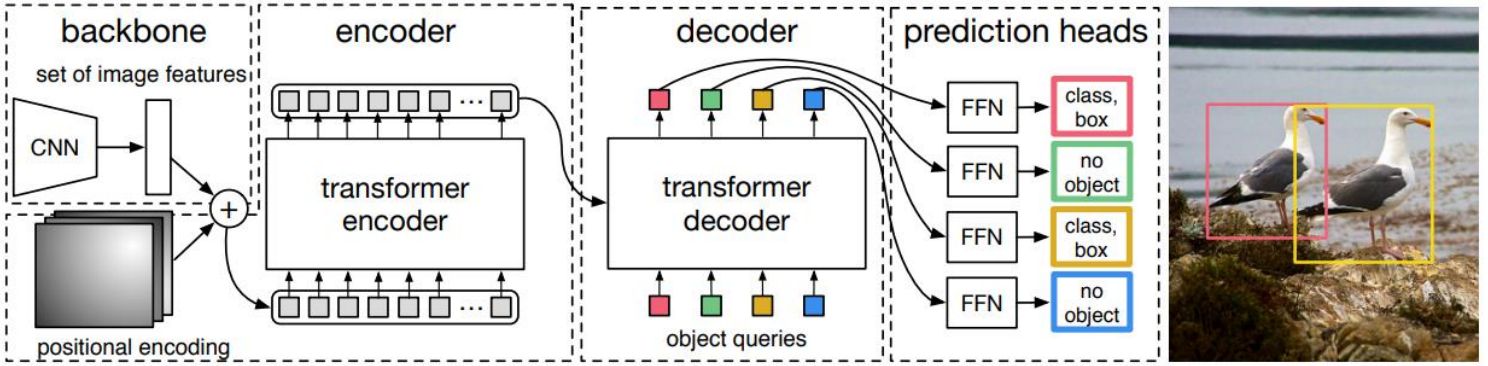
- 이러한 object queries는 길이가 N으로, decoder에 의해 output embedding으로 변환(transform)되며
- 이후 FFN을 통해 각각 독립적으로 box coordinate(좌표)와 class label로 decode됨

- 이는 **각각의 object query는 하나의 객체를 예측하는 region proposal에 대응**된다고 볼 수 있음
- 즉, object queries는 N개의 객체를 예측하기 위한 일종의 사전 지식으로도 볼 수 있음

- encoder와 유사하게 object query를 각 attention layer의 입력에 더해줌
- 이 때 embedding은 self-attention과 encoder-decoder attention을 통해 이미지 내 전체 context에 대한 정보를 사용함
- 이를 통해 **객체 사이의 pair-wise relation을 포착하여 객체간의 전역적(global)인 정보를 모델링**하는 것이 가능해짐

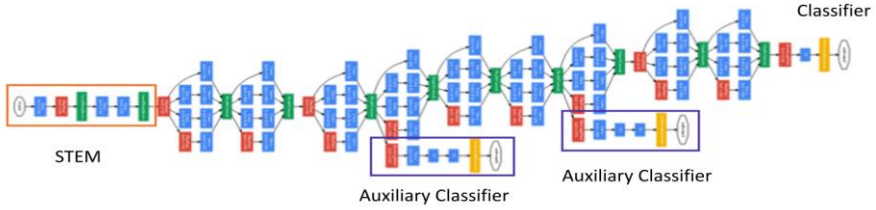
4 Prediction feed-forward networks(FFNs)

- Decoder에서 출력한 output embedding을 3개의 linear layer와 ReLU activation function으로 구성된 FFN에 입력하여 최종 예측을 수행함
- **FFN은 이미지에 대한 class label과 bounding box에 좌표**(normalized center coordinate, width, height)를 예측함
- 이 때 예측하는 class label 중 ∅은 객체가 포착되지 않은 경우로, "background" class를 의미함



5 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 준다고 함
- 추가한 FFN은 서로 파라미터를 공유하며, FFN 입력 전에 사용하는 layer normalization layer도 공유함




```
total_loss = real_loss + 0.3 * aux_loss_1 + 0.3
* aux_loss_2
```

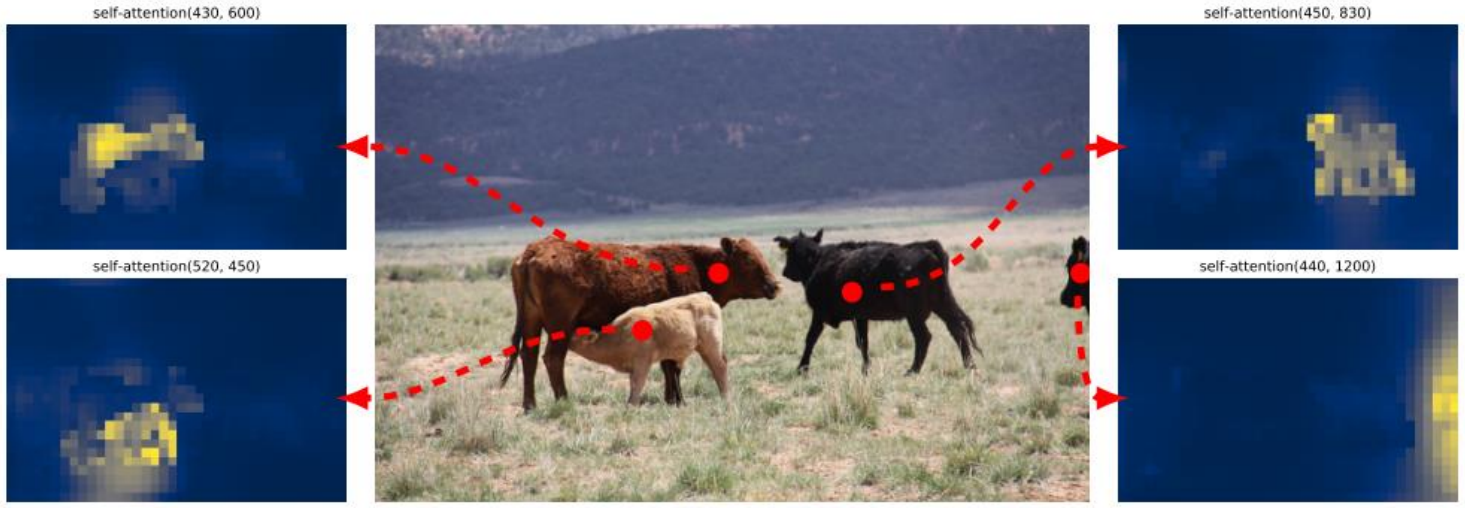
GoogLeNet
- Auxiliary Loss는 깊어진 layer에서 발생하는 vanishing gradient 문제를 개선하기 위하여 적용된 트릭임
- 중간 중간에 Loss를 계산할 수 있는 layer를 추가적으로 만들어서 최종 loss에 반영하게 됨
- 즉 label이 있으면 최종 output에서 loss를 계산할 때 사용하고 중간 중간에도 있는 Auxiliary loss에서도 값을 계산해서 반영함

4. Experiments

Comparison with Faster R-CNN

- COCO dataset에 대한 실험 결과, DETR은 같은 수의 파라미터로 Faster R-CNN과 비슷한 수준의 성능을 보였음

Ablations

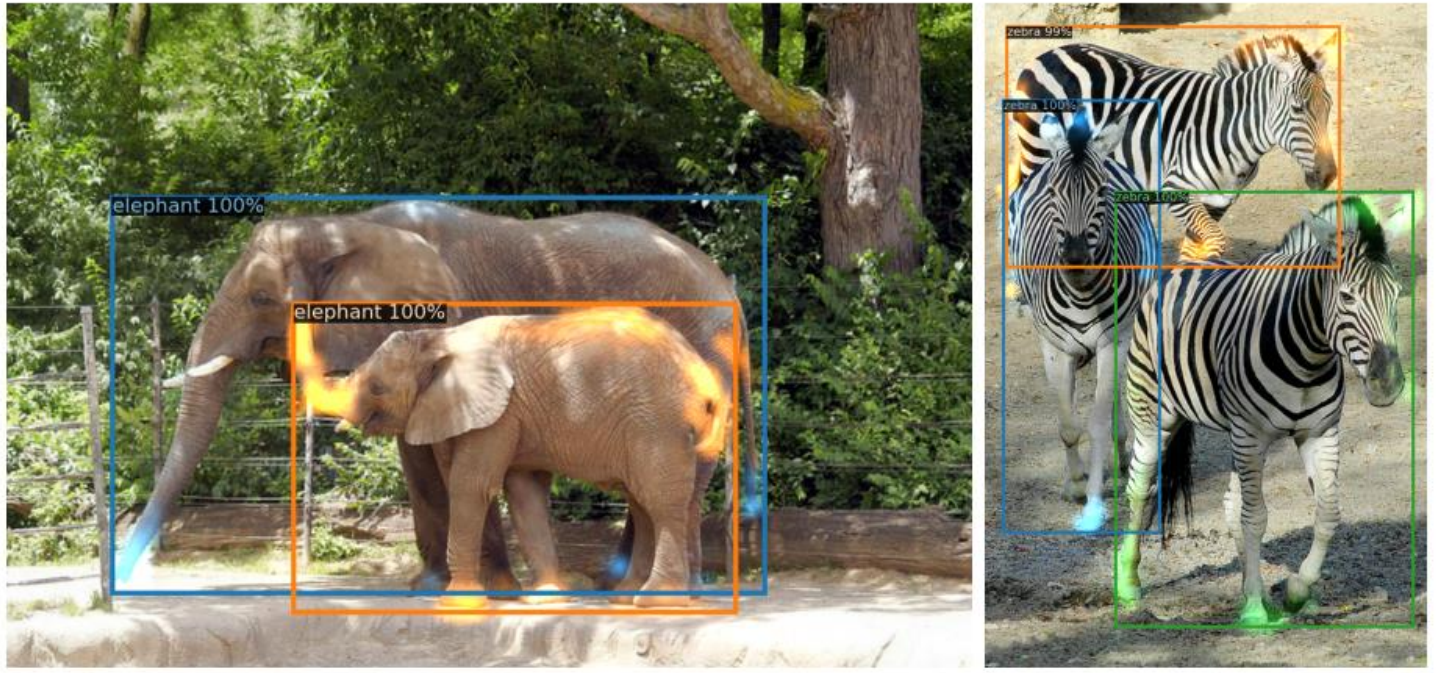


Encoder의 attention map

Number of encoder layers

- Encoder 수로 ablation study를 진행한 결과 encoder를 전혀 사용하지 않으면 성능이 3.9%p 감소했음

- 저자들은 encoder가 global scene reasoning을 통해 객체를 분리하는 작업을 수행하기 때문이라고 가정함
- 이는 encoder layer의 attention map을 시각화한 그림을 통해 알수 있음



예측한 객체에 대한 Decoder attention 시각화

Number of decoder layers

- decoder를 추가할수록 성능이 향상되었으며, 첫 번째와 마지막 decoder 사이에는 +8.2/9.5 AP 차이가 있었음
- 하나의 decoder만을 사용한 경우 output에 대한 cross-correlation을 연산하지 못하기 때문에 duplicate prediction이 발생하였음
- Decoder의 attention map을 시각화한 결과, 그림과 같이 다소 local하게 객체를 포착하고 있음
- 저자들은 encoder가 global attention을 통해 객체를 배경으로부터 분리했기 때문에 decode는 객체의 경계 부분만 포착하면 될 것이라고 가정함

Senseable AI Lab

Importance of FFN

- Transformer에서 FFN을 제거하고 attention layer만 사용한 경우, AP 값이 2.3%p 감소했음

spatial pos. enc. encoder	spatial pos. enc. decoder	output pos. enc. decoder	AP	Δ	AP_{50}	Δ
none	none	learned at input	32.8	-7.8	55.2	-6.5
sine at input	sine at input	learned at input	39.2	-1.4	60.0	-1.6
learned at attn.	learned at attn.	learned at attn.	39.6	-1.0	60.7	-0.9
none	sine at attn.	learned at attn.	39.3	-1.3	60.3	-1.4
sine at attn.	sine at attn.	learned at attn.	**40.6**	-	**61.6**	-

positional encoding 방법에 따른 성능

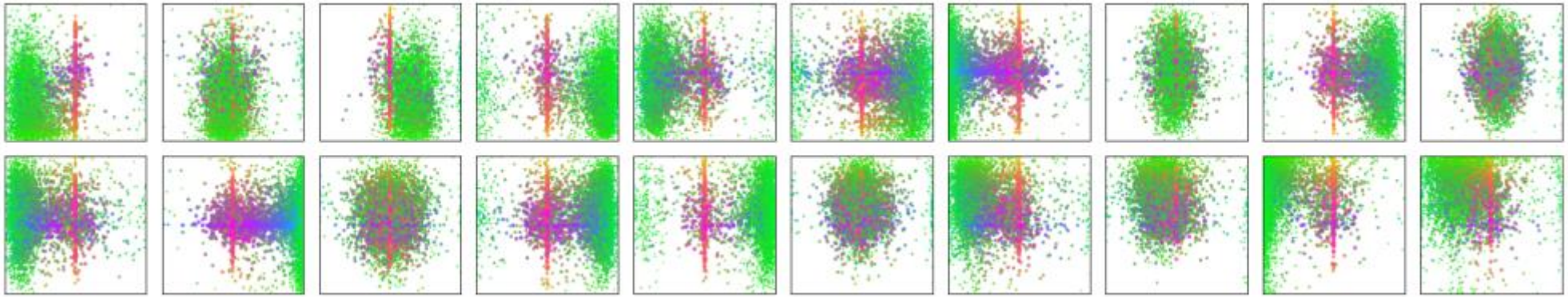
Importance of positional encoding

- spatial positional encoding을 사용하지 않고 decoder에 object queries만 사용한 결과 베이스라인에 비해 AP 값이 7.8%p 감소했음
- 또한 decoder에 단 한 번 object query를 입력한 결과 모든 attention layer마다 object query를 더해주었을 때에 비해 AP 값이 1.4%p 감소했음

Loss ablations

- L1 loss와 GIoU loss에 대한 ablation 실험을 수행한 결과, GIoU loss만 사용한 경우 성능이 0.8%p 정도만 감소했으나, L1 loss만 사용한 경우 성능이 5%p 가까이 감소하였음
- 이를 통해 GIoU loss가 사실상 bounding box loss에 대한 기여도가 매우 높음을 알 수 있음

Analysis



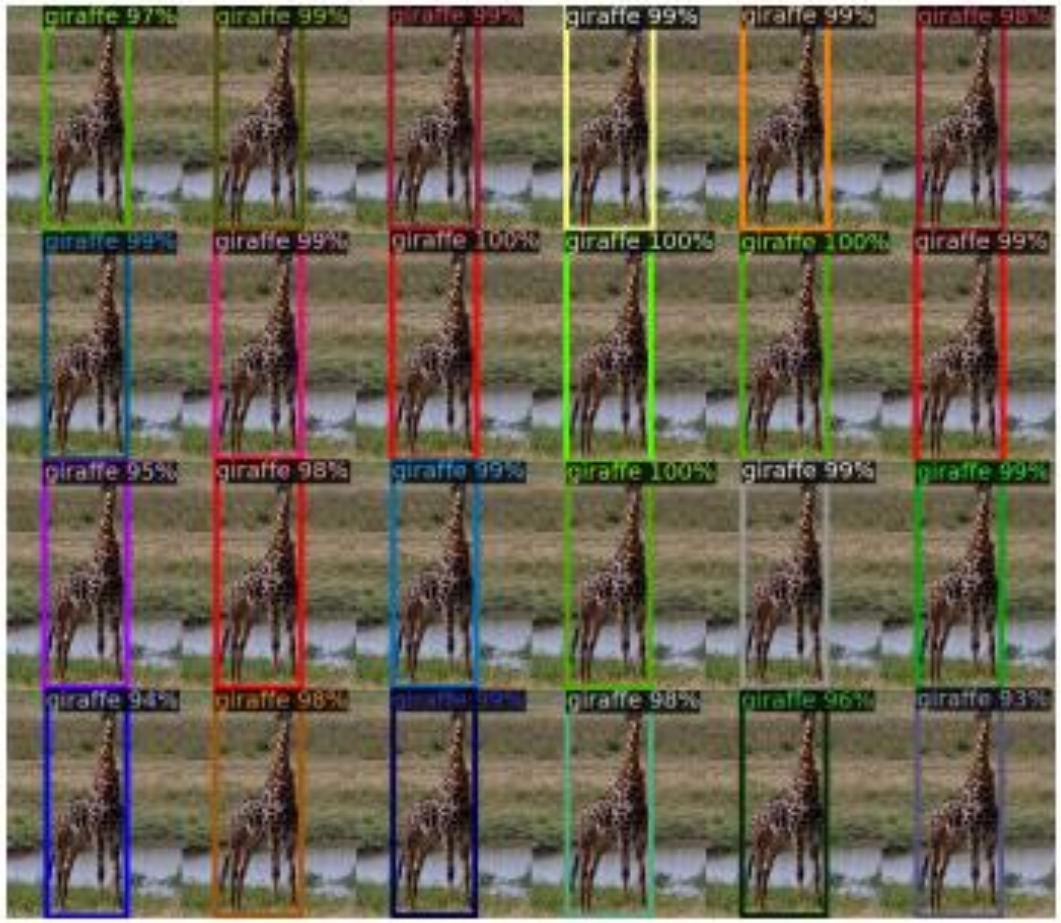
box prediction에 대한 시각화

Decoder output slot analysis

- 각 object query의 slot에 대한 분석을 진행한 결과 각 slot은 여러 모드로 특정 영역과 box 크기의 객체를 학습하는데 특화된 모습을 보였음
- Fig 16 은 20개의 slot에 대한 시각화 결과로, 각 점은 예측한 bounding box의 center coordinat을 의미함
- 초록색은 작은 box, 빨간색은 큰 수평 box, 파란색은 큰 수직 box에 해당함
- 또한 거의 모든 slot이 COCO dataset에서 자주 등장하는 입력 이미지와 비슷한 크기의 box를 예측하는 모드를 가지고 있음

Senseable AI Lab

Analysis



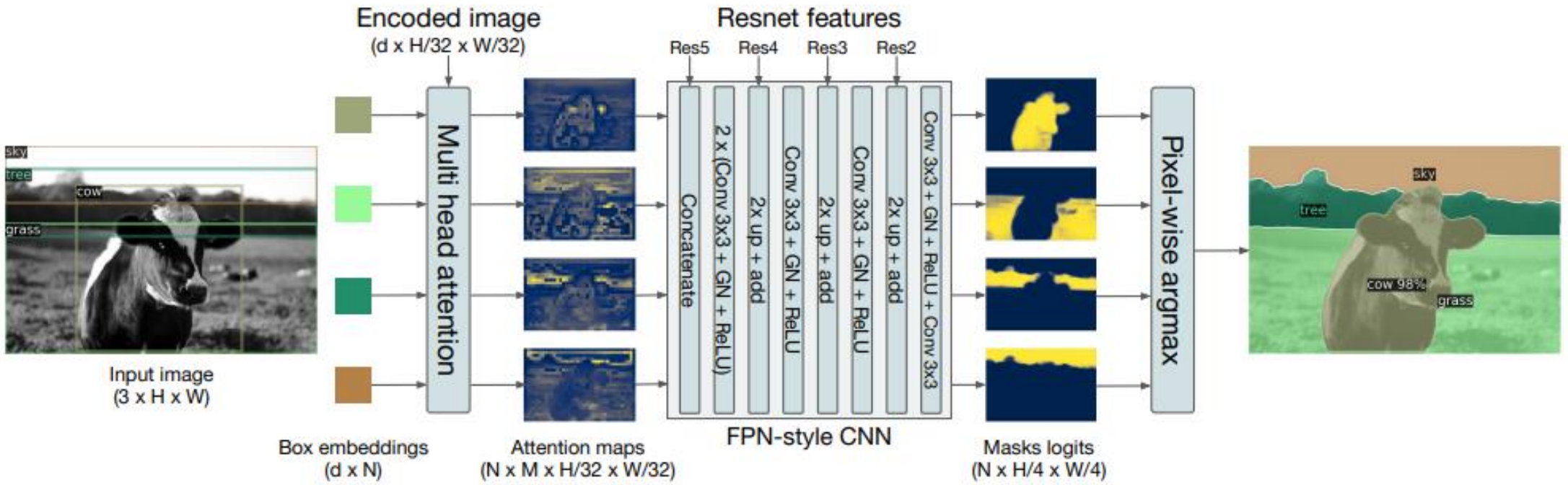
Synthetic data 예시 및 예측 결과

Generalization to unseen numbers of instances

- 몇몇 class는 이미지 내 instance가 충분히 등장하지 않는 경우가 있음
- COCO dataset에서 기린(giraffe)이 13마리 이상 등장하는 이미지가 없음
- 학습 데이터셋에 없는 수의 instance에 대한 실험을 진행하기 위해 그림과 같이 synthetic image(가상 이미지)를 생성하여 실험을 진행
- 실험한 결과 이미지 내 등장하는 24마리의 기린(out-of-distribution)을 모두 잘 포착하였음
- 이를 통해 각 object query에는 강력한 class-specialization이 없음을 확인하였음

4. Experiments

4.2 DETR for panoptic segmentation



- DETR은 panoptic segmentation task에도 확장 가능함을 보였음
- Panoptic segmentation은 semantic- instance segmentation을 결합한 task임
- 해당 task에서는 셀 수 있는 객체(차, 사람 등)을 Thing class, 셀 수 없는 객체(하늘, 길 등)을 Stuff class라고 합니다. Thing class에 대해서는 instance segmentation을, Stuff class에 대해서는 semantic segmentation을 수행함

- Decoder output에 mask를 예측하는 head를 추가함으로써 segmentation을 수행할 수 있음
- 실험 결과 기존의 연구와 비슷한 수준의 성능을 보였음
- 특히 DETR은 Stuff class에 대하여 좋은 결과를 보였음. 저자들인 encoder attention에서 학습한 global reasoning 덕분이었다고 가정함

5. Conclusion

5. Conclusion

- DETR은 object detection을 set prediction task로 정의하여 prediction과 ground truth 사이의 일대일 matching을 수행하였음
- 이를 통해 near duplicate prediction을 효과적으로 감소시켜 post-processing 과정이 없는 end-to-end 프레임워크를 제안하였음
- 이 과정에서 encoder-decoder 구조의 Transformer를 활용함으로써 입력 token 사이의 pair-wise interaction과 global reasoning을 통해 준수한 성능을 보였음

- 하지만 본 논문에서 제안한 방법도 단점이 있다고 할 수 있음
- 여러 크기의 anchor를 사용하지 않기 때문에 다양한 크기, 형태의 객체를 포착하지 못함
- 또한 하나의 예측 bounding box를 ground truth에 matching하기 때문에 수렴하는데 훨씬 긴 학습시간을 필요로 함
- 그럼에도 불구하고, 본 논문에서 제안한 DETR은 Faster R-CNN과 비슷한 성능을 보이면서도 추가적인 추론 과정이나 후처리을 필요하지 않다는 점에서 큰 의의를 가진다고 할 수 있음

Q & A

Thank You