



End-to-End Object Detection with Transformers

Abstract

우리는 오브젝트 디텍션을 직접적인 예측 문제로 다루는 방식을 제시한다. 우리는 직접적으로 제작이 필요한 NMS나 anchor generation을 없애서 오브젝트 디텍션의 파이프라인을 간소화시킨다. 우리 모델의 핵심은 예측과 정답간 1대1 매칭과 트랜스포머의 인코더 디코더 구조이다. 학습 가능한 쿼리에 따라 물체와 전체 이미지 간의 관계를 찾고 각 예측에 대하여 병렬적으로 결과를 추출한다. 또한 개념적으로 간단하며 다른 디텍션 모델과 달리 특수 라이브러리를 필요로 하지 않는다. 그리고 DETR은 COCO object detection challenge에서 faster R-CNN과 견주는 정확도와 속도를 보입니다. 또한 panoptic segmentation이 가능하다.

- panoptic segmentation : semantic segmentation + instance segmentation으로 이미지 내 모든 픽셀에 대하여 클래스와 객체를 배정하는 segmentation

Introduction

오브젝트 디텍션의 목표는 바운딩 박스의 위치와 카테고리 라벨을 예측하는 것입니다. 많은 모델들은 proposal, anchors, window centers등을 사용하여 우회적인 방식으로 예측을 진행합니다. 게다가 이러한 방식은 겹치는 구역으로 인해 이를 처리하는(NMS와 같은) 후처리 방식이 필요합니다. 이를 간소화시키기 위해, 우리는 직접적인 방식을 사용합니다. 이 end-to-end 방식은 NLP와 같은 분야에서 많은 발전을 이뤘지만 아직 detection에서는 그렇지 않습니다. 있다고 하더라도, 성능면에서 좋지 않습니다. 그래서 우리는 이 격차를 해소하기 위해 논문을 작성합니다.

우리는 트랜스포머의 인코더 디코더 구조를 채택합니다. 트랜스포머는 문장 내 모든 원소가 상호작용하게 만듬으로써, 중복 예측을 제거하는 것과 같은 적절한 특정 제약을 줄 수 있습니다.

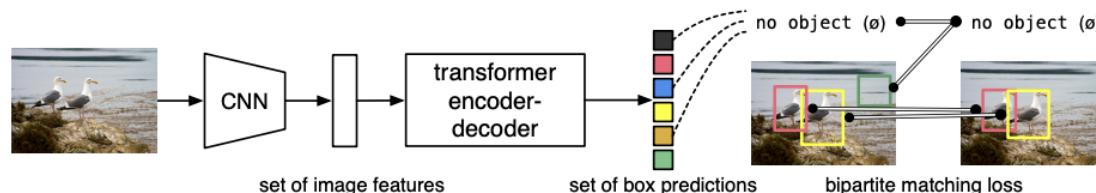


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a “no object” (\emptyset) class prediction.

우리의 모델 DEtection TRansformer(DETR)은 한번에 모든 물체에 대해 예측하며 end-to-end로 학습이 진행됩니다. DETR은 다양한 크기의 anchor box를 만들거나 NMS를 통하여 중복을 제거하는 등과 같은 작업을 없앰으로써 파이프라인을 간단하게 합니다. 그래서 기본적인 CNN과 Transformer만을 이용하여 제작가능합니다.

이전에 직접적인 예측 방식과 다른 점은 DETR은 1대1 매칭에 대한 로스와 병렬적인 디코딩 방식입니다. 이전의 작업은 RNN을 통해 decoding을 진행했습니다. 우리의 매칭 로스는 1대1 대응을 통하여 각 객체에 대하여 병렬적으로 수행이 가능합니다.

우리는 DETR을 COCO에서 평가했으며 이는 Faster R-CNN기반의 방식과 견줄 수 있는 성능을 보입니다. 게다가 DETR은 전체적인 정보를 파악하는 transformer구조로 인해 보다 큰 물체에 대해 파악을 잘합니다. 작은 물체에서는 성능이 떨어집니다.

DETR의 학습 방식은 다른 모델과는 다르게 긴 학습 구조를 가지고 있기 때문에 중간에 보조 분류기를 통해 이점을 얻습니다. 추가적으로 DETR은 segmentation head를 추가하여 panoptic segmentation으로도 사용이 가능합니다.

Related work

우리의 작업은 몇가지 도메인에서 만들었습니다. set prediction, transformer, parallel decoding, object detection

Transformers and Parallel Decoding

트랜스포머는 attention 기반의 모델입니다. attention mechanism은 인풋의 전반적인 정보를 통합하는 방식입니다. 시퀀스의 각 요소를 확인하고 전체적인 시퀀스로 부터 나온 정보를 업데이트합니다. 가장 큰 장점은 전체 정보에 대한 계산과 긴 문장에 대해 적합하다는 것입니다. 트랜스포머는 현재 NLP, 음성 그리고 비전 분야에서 RNN을 대체하고 있습니다. 트랜스포머는 원래 하나씩 추론하는 방식이었으나 많은 계산 비용으로 인해 병렬적으로 변경됐습니다. 그리고 우리도 transformer와 병렬적인 디코딩 방식을 이용합니다.

Object detection

대부분의 발전된 Object detection model은 초기 모델들과 관련성이 있습니다. Two-stage의 경우 proposal에 대해 예측하며, one-stage의 경우 anchors나 그리드에 대해 예측합니다. 최근 연구에서 이러한 시스템의 최종 결과는 초기 설정에 많이 좌지우지 된다고 합니다. 그래서 우리는 수작업이 필요한 방식들을 제거하여 간단화합니다.

몇몇 모델은 양방향 매칭 로스를 사용합니다. 그러나 초기 모델에서는 다른 예측 사이의 관계는 convolution 혹은 fc layer로 구성되었고 NMS 방식이 성능을 올려줍니다.

학습가능한 NMS와 관계 네트워크는 명확히 어텐션과 함께 다른 예측 사이의 관계를 만듭니다. direct set losses를 사용하면, 어떠한 처리단계가 필요하지 않습니다. 그러나, 이러한 방식은 추가적인 방식이 필요합니다.

Recurrent detectors는 우리의 현재 방식과 유사하게 end-to-end set 예측을 합니다. CNN activations을 이용해 encoder-decoder 구조를 사용하며 바운딩 박스를 직접적으로 예측합니다. 그러나, 이는 작은 데이터셋에서만 평가 되었으며 RNN과 같은 autoregressive model에 기반합니다. 그리고 병렬 디코딩 같은 최근의 트랜스포머를 이용하지 않습니다.

The DETR model

직접적인 예측에서 두가지가 중요합니다.

1. 예측과 정답간의 1대1 대응에 대한 로스 산정
2. 1대1 대응을 예측하는 모델

Object detection set prediction loss

DETR은 N개의 예측을 하도록 고정되어 있습니다. N은 전형적인 이미지에 포함된 객체보다 훨씬 큰 숫자를 가집니다. 학습의 어려운점은 정답값에 대하여 class, 위치, 크기에 대한 평가를 산정해야하는 것입니다. 우리의 로스는 정답과 예측에 대하여 최적의 매칭을 만들고 위치 로스에 대해 최적화시킵니다. y 를 정답값의 집합, \hat{y} 를 예측값에 집합이라 할 때 각각 N개가 있습니다. N개는 이미지에 있는 객체보다 크기 때문에 정답 값 라벨은 \emptyset 으로 채웁니다. 그리고 우리는 양 쪽 집합에서 최적의 1대1 대응을 찾기 위해 아래와 같은 로스를 사용합니다.

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

$L(\text{match})$ 는 y 와 \hat{y} 간의 매칭 코스트를 이야기 합니다. 이러한 최적화 방식은 Hungarian algorithm을 사용합니다.

<https://gazelle-and-cs.tistory.com/29> (Hungarian algorithm 참조)

각각의 매칭되는 비용은 클래스 뿐만 아니라 박스에 대한 것도 고려합니다.

C_i 는 클래스를 고려하며 $B_i \in [0, 1]^4$ 는 박스의 가운데 값에 대한 위치와 크기를 고려합니다. 이에 따라 우리는 $L(\text{match})$ 를

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

로 표현이 가능합니다. 이러한 방식은 proposal을 찾는 휴리스틱한 방식과 같은 유사한 역할을 수행합니다. 가장 큰 차이는 우리는 정답 값에 대하여 한개의 예측 값만을 찾는다는 것입니다. 두 번째 단계는 Hungarian loss를 이전 단계에서 모든 쌍에 대하여 계산합니다. 클래스 예측을 위한 negative-log-likelihood의 선형 결합을 로스와 박스 로스를 사용합니다.

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

그리고 우리는 클래스가 백그라운드일 경우 업데이트 가중치를 10분의 1로 감소시킵니다. 이는 클래스 불균형을 해결하기 위해서입니다. 매칭 코스트에 관해서는 우리는 클래스에 대하여 로그를 사용하지 않았는데 이는 실험적으로 더 좋은 결과를 보였습니다.

Bounding box loss 두 번째 매칭 비용인 박스 로스의 경우 다른 디텍터의 박스 예측이 초기 값을 가지는 것에 비해 우리는 직접적인 예측을 합니다. 간소해진 우리의 방식에 따라 객체에 크기에 따른 상대적인 로스 문제를 겪습니다. 흔히 사용하는 L1 loss의 경우 객체에 사이즈에 따라 상대적인 로스 부과 문제를 겪습니다. 이를 해결하기 위해 우리는 L1 loss와 GIOU loss를 결합하여 사용합니다. GIOU loss는 scale에 영향을 받지 않습니다. $L(\text{BOX})$ 로스는 아래와 같이 정의됩니다.

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

DETR architecture

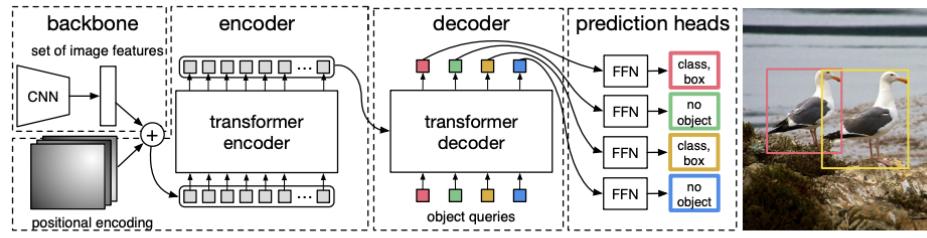


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a “no object” class.

우리의 모델은 놀랍게도 간단합니다. 총 3가지의 메인 요소를 가지고 있습니다. 피쳐 추출을 위한 CNN backbone, encoder-decoder transformer, 그리고 간단한 최종 예측을 하는 FFN입니다. 다른 디텍션 모델과 다르게 DETR은 CNN과 transformer의 구조만 있으면 사용이 가능합니다. 인퍼런스 코드도 50줄만 있으면 됩니다. 이러한 간단함은 디텍션 연구에 큰 영향을 가져다 줄 것입니다.

BACKBONE 초기 이미지 $3 \times H \times W$ 에서 시작해서 CNN BACKBONE은 피쳐맵 $C=2048$, $H1 = H/32$, $W1 = W/32$ 로 나옵니다.

TRANSFORMER ENCODER 먼저, 1×1 conv는 채널의 차원을 감소시킵니다. 그리고 ENCODER는 인풋으로 시퀀스를 요구하기 때문에 우리는 피쳐맵을 감소된 C , $H1 \times W1$ 로 변경시킵니다. 각각의 인코더 레이어는 멀티 헤드 셀프 어텐션과 FFN으로 구성되어있습니다. 따라서 트랜스포머는 위치 정보를 신경쓰지 않으므로 positional encoding을 추가합니다.

TRANSFORMER DECODER 디코더는 트랜스포머의 기본 구조로 d size의 N 개의 임베딩을 진행합니다. 원본 트랜스포머와 다른 점은 우리는 디코더 레이어에 N 개의 오브젝트를 병렬적으로 주입합니다. 디코더 또한 위치에 따른 정보가 반영되지 않기 때문에 각각의 N 개의 임베딩은 다른 결과를 도출합니다. 이 임베딩이 우리가 언급한 *object queries*입니다. 그리고 인코더와 유사하게 이 임베딩을 각각의 어텐션 레이어에 주입합니다. N 개의 쿼리는 디코더에 의해 아웃풋 임베딩을 생산합니다. 그리고 독립적으로 FFN에 의해 class label과 박스 정보를 도출하고 이는 총 N 개의 결과를 내보냅니다. 셀프 어텐션 인코더와 디코더를 통해 모델은 이미지 전체의 모든 객체를 한번에 찾게 됩니다.

PREDICTION FEED-FORWARD NETWORKS(FFNs) 최종 예측은 3개의 퍼셉트론 레이어와 ReLU 활성화 함수로 진행됩니다. FFN은 박스의 가운데 좌표와 높이 너비를 예측하며 클래스를 예측하게 됩니다. 그리고 \emptyset 는 객체가 없음을 의미합니다.

AUXILIARY DECODING LOSSES 우리는 디코더에서 보조 분류기가 유용한 것을 발견했습니다. 특히 이미지 내 객체의 갯수를 판단하는데 도움이 되었습니다. 우리는 보조 분류기를 각각의 디코더 레이어에 추가합니다. 그리고 모든 FFN은 동일한 가중치를 공유합니다. 레이어 노멀라이즈 값도 공유합니다.

Experiments

우리는 DETR이 COCO에서 Faster R-CNN과 대등함을 보였습니다. 그리고 새로운 구조 및 로스의 제시를 합니다.

AdamW, LR 0.0001, Xaview init, DropOUT 0.1 등을 사용했으며 Backbone으로 ResNet50, 101을 사용했습니다. 또한, backbone의 마지막 레이어에서 피쳐의 해상도를 증가시키는 실험도 진행했습니다. 이는 해상도를 두배로 증가시켰고 전체적으로 2배의 컴퓨터 자원을 소비했습니다. 어그먼테이션으로 랜덤 크롭, 리사이즈를 진행했습니다. 랜덤 크롭의 경우 1AP를 증가시켰습니다.

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 $9\times$ training schedule. DETR models achieve comparable results to heavily tuned Faster R-CNN baselines, having lower APs 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	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

우리는 컨피던스 스코어를 통한 결과 필터링을 통해서 2AP 정도의 상승을 얻었습니다.

우리는 16개의 GPU를 통해 3일간 300Epochs를 학습시킵니다. Faster R-CNN과 비교하기 위해서는 500epoch을 학습시켰으며 이는 1.5AP 상승을 가져다 줍니다.

우리는 인코더의 중요성을 파악하기 위해 인코더 없이 실험을 해보았을 때 3.9AP 감소를 얻었습니다.

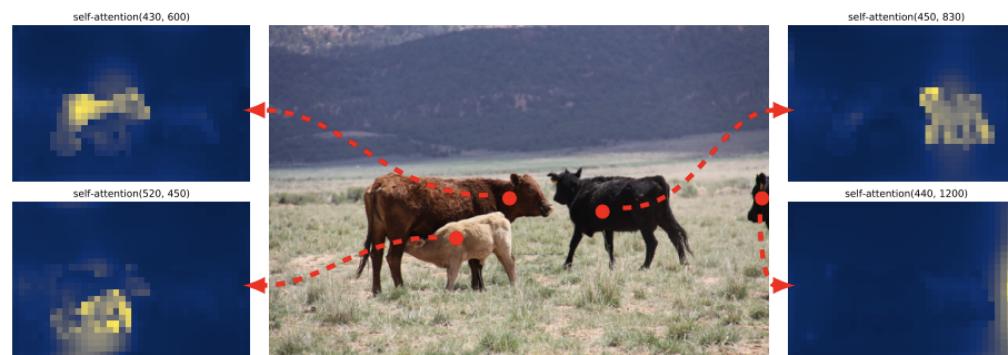


Fig. 3: Encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image.

또한 인코더 마지막 레이어를 시각화 해보았을 때 위와 같은 결과를 얻었고 인코더를 통해 이미지의 적절한 정보를 디코더에 가져다 줄 수 있습니다.

디코더 레이어의 중요성은 보조 분류기에서 알 수 있습니다. 첫 번째 레이어의 보조 분류기보다 마지막 레이어의 결과값은 8~9정도의 AP 차이가 납니다.

또한, DETR은 NMS가 필요 없다는 것을 증명하기 위해 NMS를 각 레이어에 넣어 실험해보았습니다. 첫번째 레이어의 경우 성능이 증가했으나 마지막 레이어에서는 성능이 감소했습니다. 첫 번째 레이어의 경우에는 한 객체에 대해 여러 예측을 하는 경향이 있기 때문입니다.

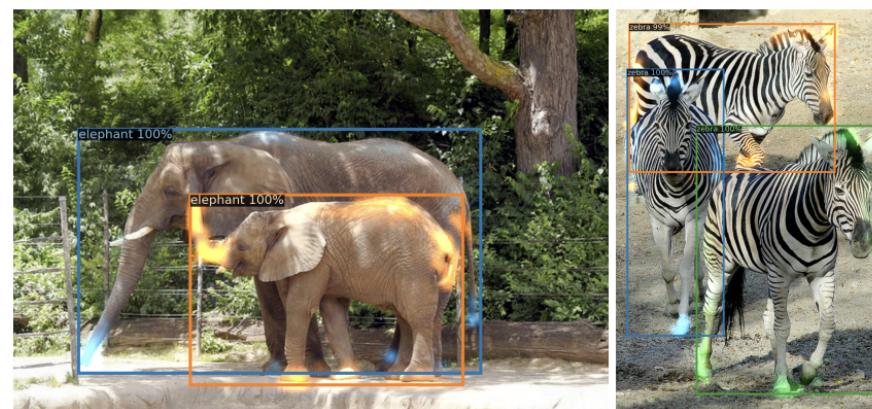


Fig. 6: Visualizing decoder attention for every predicted object (images from COCO val set). Predictions are made with DETR-DC5 model. Attention scores are coded with different colors for different objects. Decoder typically attends to object extremities, such as legs and heads. Best viewed in color.

위는 디코더 레이어를 시각화한 결과입니다. 이를 봤을 때 디코더 레이어는 박스의 위치와 클래스에 집중하는 것을 볼 수 있습니다.

FFN도 없애보았으나, 이는 파라미터가 10M이상 감소했으나 2.3AP가 감소하여 FFN이 중요함을 확인합니다.

Positional Encoding또한 없애서 진행했는데 7.8AP가 감소함을 확인했습니다.

Table 4: Effect of loss components on AP. We train two models turning off ℓ_1 loss, and GIoU loss, and observe that ℓ_1 gives poor results on its own, but when combined with GIoU improves AP_M and AP_L. Our baseline (last row) combines both losses.

class	ℓ_1	GIoU	AP	Δ	AP ₅₀	Δ	AP _S	AP _M	AP _L
✓	✓		35.8	-4.8	57.3	-4.4	13.7	39.8	57.9
✓		✓	39.9	-0.7	61.6	0	19.9	43.2	57.9
✓	✓	✓	40.6	-	61.6	-	19.9	44.3	60.2

로스의 영향력도 살펴 보았는데 다 사용한 것이 확실한 성능 보장을 알 수 있습니다.

Detr은 N개의 query를 통해 진행하다 보니, 많은 객체에 대해서 못잡을 수도 있다고 생각할 수 있습니다. 또한, COCO에서 조차 이미지 내 같은 객체가 가장 많이 있는 경우를 예를 들어 13개의 기린입니다. 따라서 우리는 객체 수에 대한 강건함을 증명하기 위해 이미지 합성을 통해 24개의 기린 이미지를 만들어 냈고 이에 대해 잘 수행했습니다.

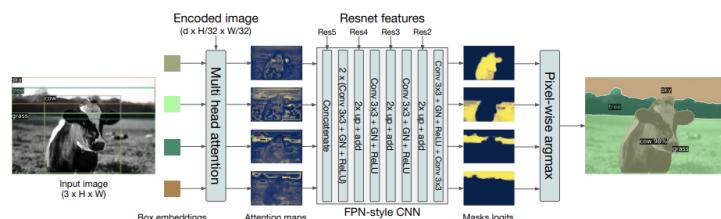


Fig. 8: Illustration of the panoptic head. A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax.



Fig. 9: Qualitative results for panoptic segmentation generated by DETR-R101. DETR produces aligned mask predictions in a unified manner for things and stuff.

Detr은 panoptic segmentation에서도 사용 가능하며 학습 방식은 FPN의 헤드와 로스만을 바꾸어 추가적인 학습을 통해 진행됩니다.

Conclusion

우리는 오브젝트 디텍션의 새로운 형식인 DETR을 제시합니다. DETR은 COCO에서 Faster R-CNN과 대등한 성능을 나타냈습니다. DETR은 실행하기에 간단하며 panoptic segmentation에서도 유용합니다. 게다가 Faster R-CNN보다 큰 물체에 대한 탐지에 강력합니다.