Transformer and its applications

김주영

NAVER Connect: boostcamp AI Tech - 3기

Computer Vision Track

GitHub: nestiank

nestiank@naver.com

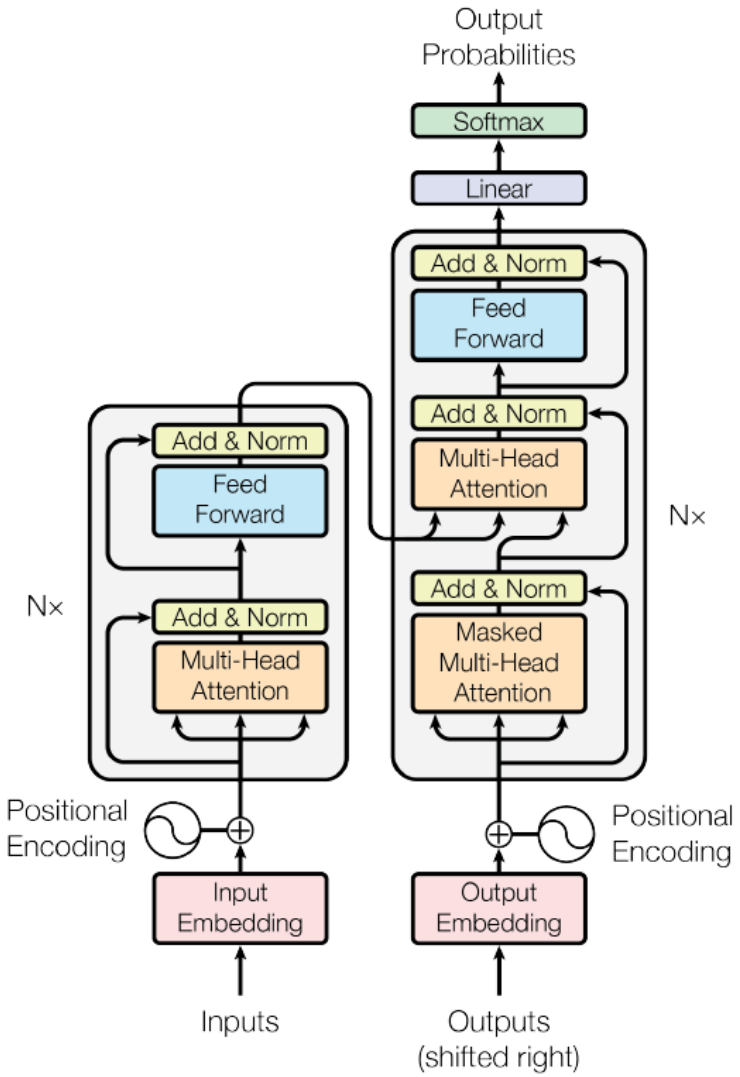
목차

- Transformer 이해
  - Attention is All You Need (https://arxiv.org/pdf/1706.03762)
  - Transformers in Vision: A Survey (https://arxiv.org/pdf/2101.0116)
- Transformer 활용
  - Non-local Neural Networks
  - BERT
  - ViT
  - UNITER
  - Unicoder
  - ViLBERT
  - Oscar
  - 12-in-1
  - VILLA
  - LXMERT
  - VirTex
  - DEIT
  - DETR
  - Deformable DETR
  - Generative Pretraining from Pixels

Transformer 이해

- Attention is All You Need
- Transformers in Vision: A Survey*
  * 여기서 다루지는 못하지만 읽어보면 많이 유용함

Positional Encoding

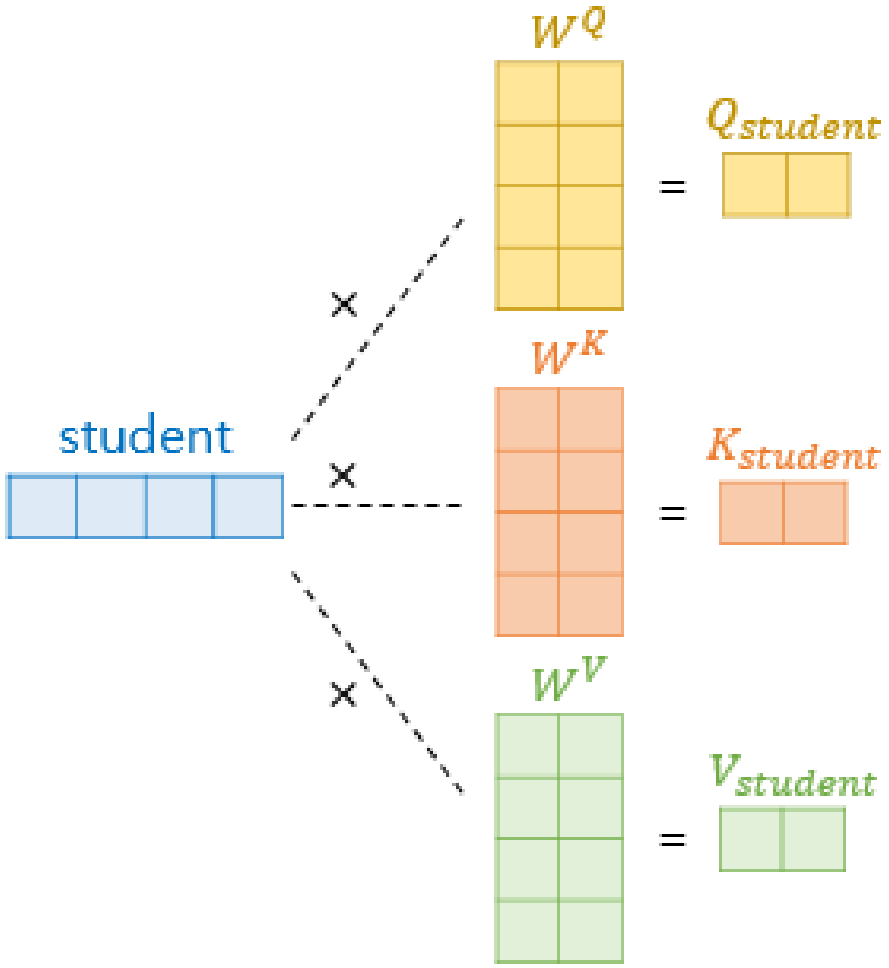


- 입력은 임베딩이 필요하다
- 임베딩에 Positional Encoding을 가한다

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

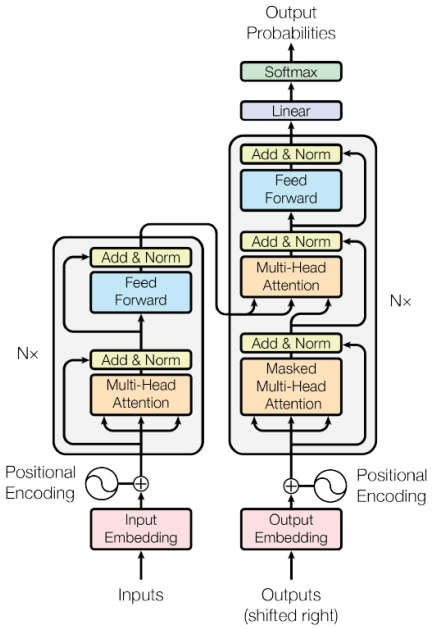
Query, Key, Value



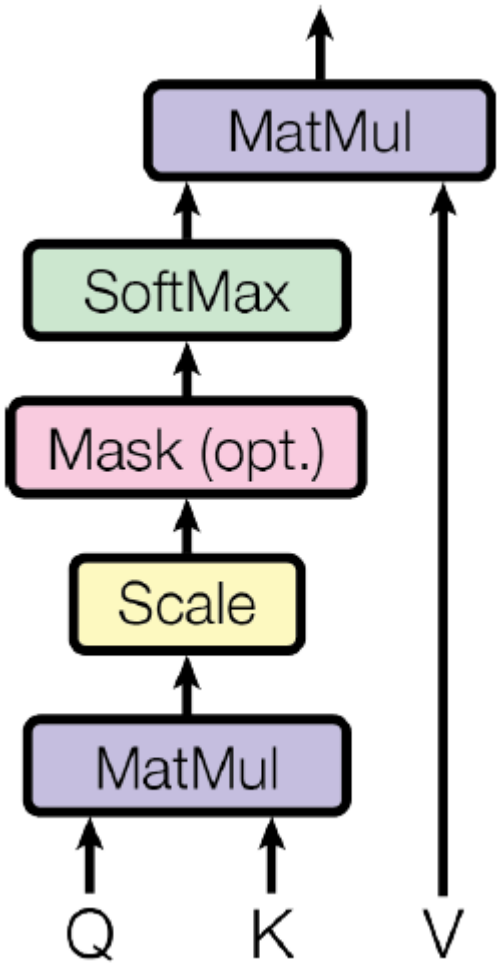
- 파란 벡터가 Positional Encoding을 마친 입력 토큰이라고 하자
  Single-Head Attention을 가정하자
- 주황색 레이어마다 3개의 W가 있음
- 이들을 곱해서 Q, K, V를 얻음

Multi-Head Attention

Masked Multi-Head Attention



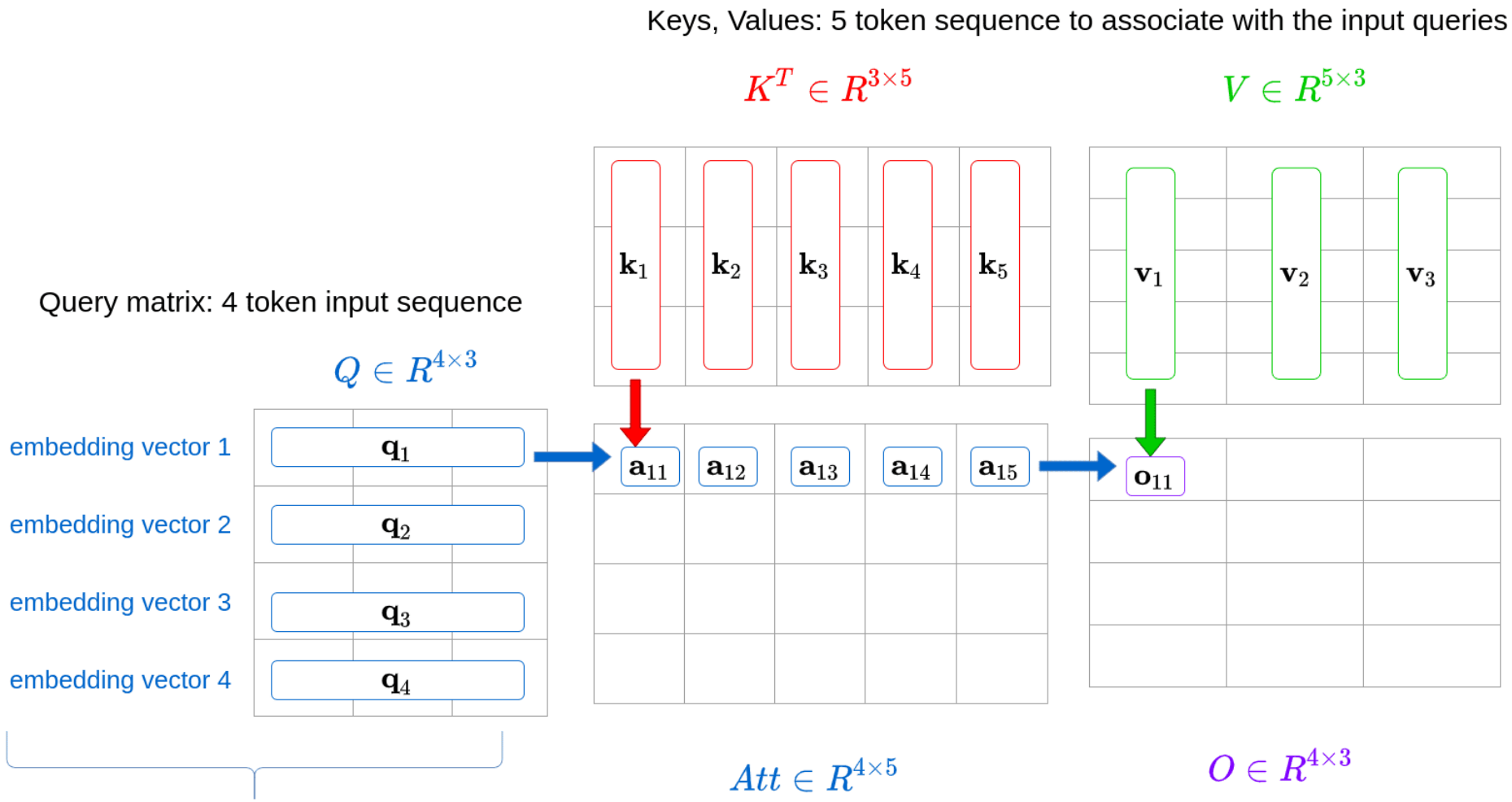
Attention



- 구한 Q, K, V로 Attention을 구함

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Attention

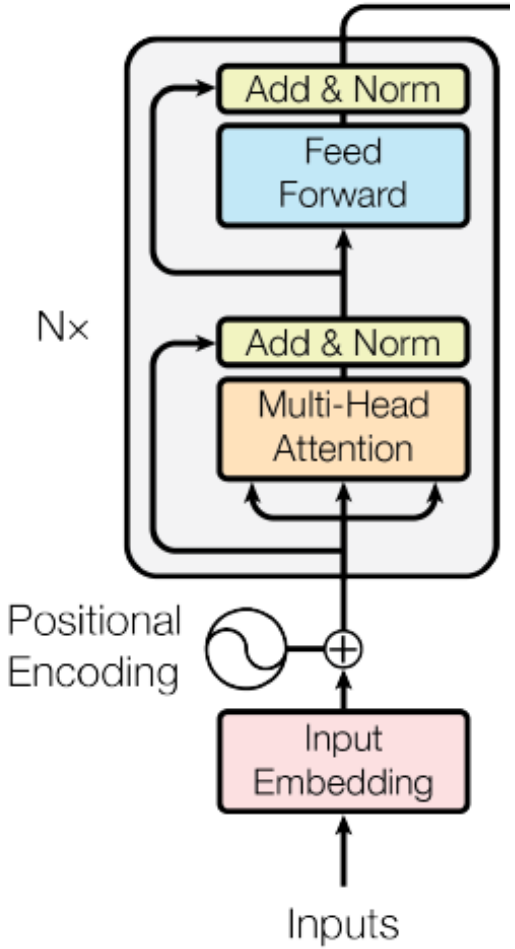


Embeddings are processed in parallel with a simple matrix multiplication

Attention matrix
each dot product show the similarity between a query and key vector
(softmax is applied row-wise)

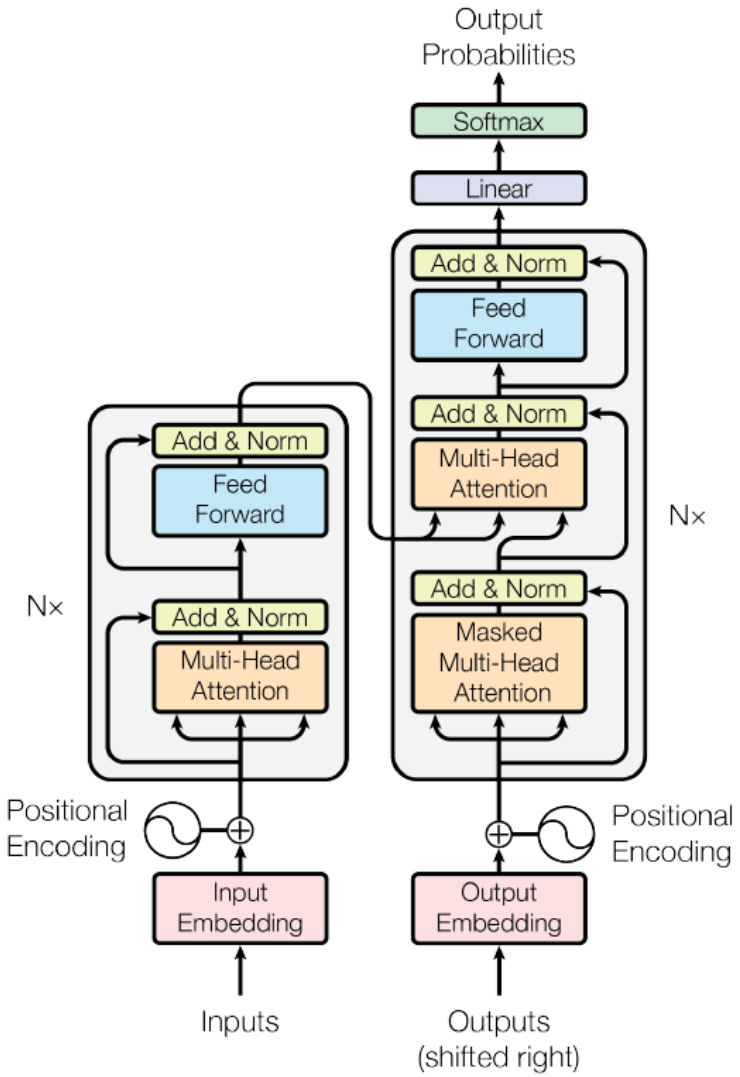
The output weighted values:
information is aggregated/routed

Transformer: Encoder



- Attention의 차원은 입력 토큰의 차원과 같음
  (그렇게 되도록 W^V의 차원을 고정함)
- 이들을 더하고 Normalization
  (또는 각각 Normalization 후 더함)
- 이후 Residual FC 레이어를 거침

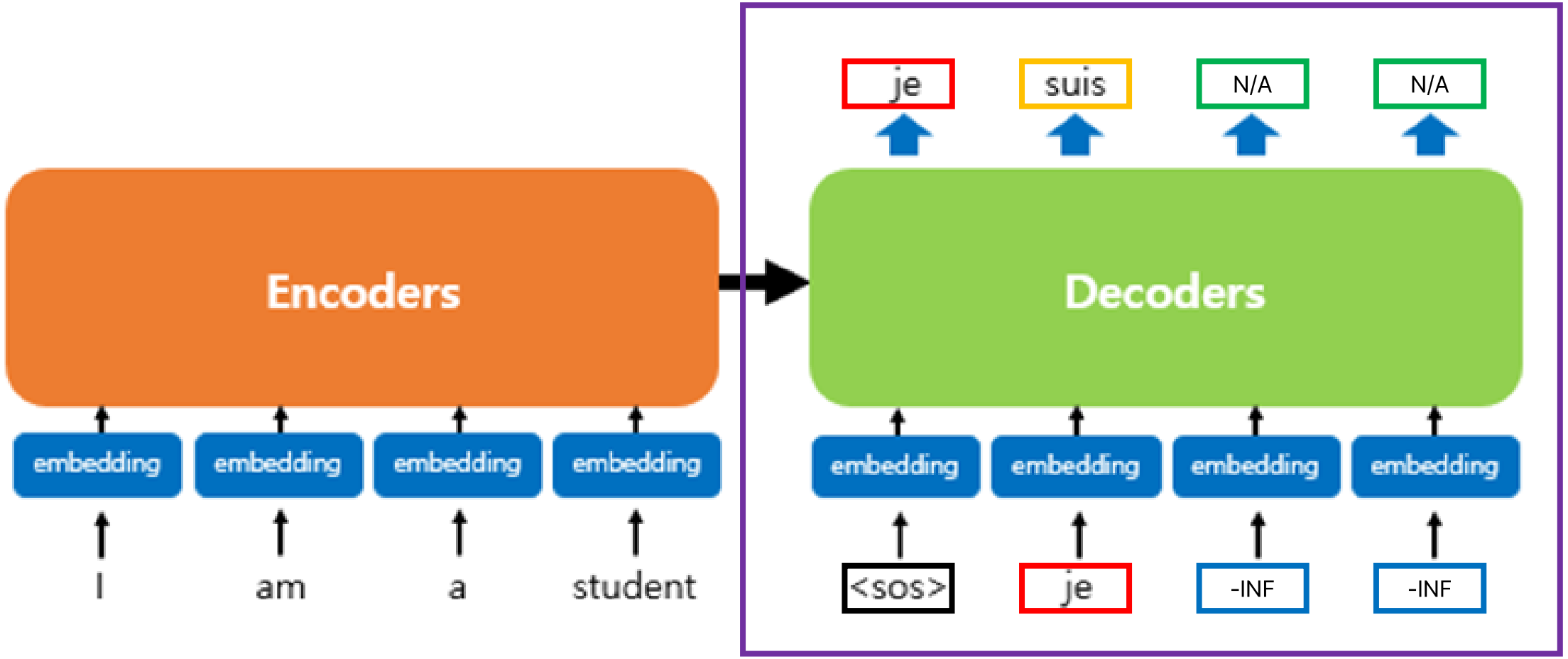
Transformer: Decoder



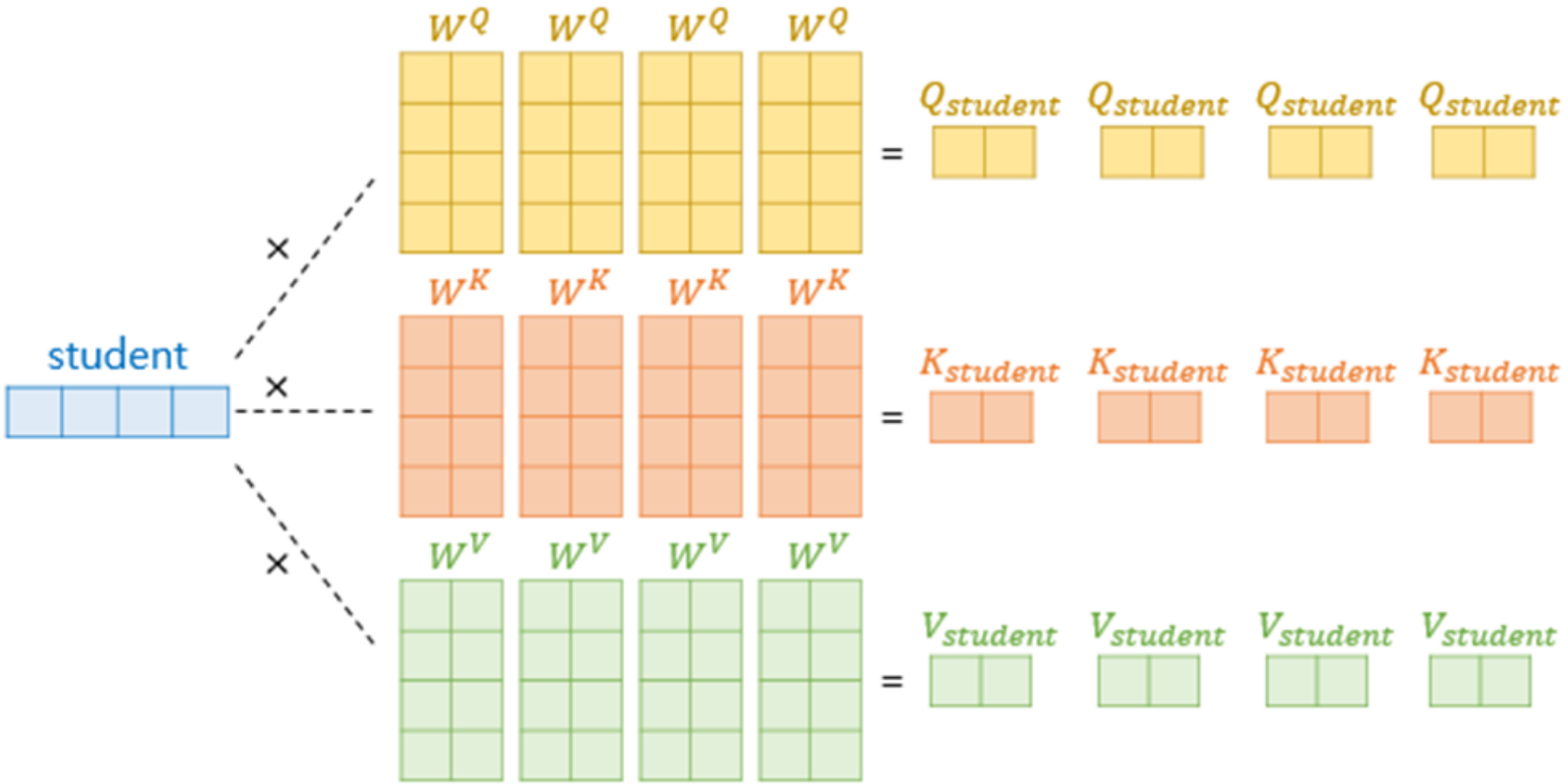
- 먼저 Masked Attention을 구함(이후 설명)
- 이것을 Q로 사용
- 마지막 Encoder의 출력을 K, V로 사용
  (같은 차례의 Encoder의 출력이 아님)
- 이후 Encoder와 같음
- 마지막 Decoder 뒤에는 FC 레이어를 통과

Masked Attention

- 이전 출력 토큰만을 가지고 다음 출력 토큰을 예측



Multi-Head Attention



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Multi-Head Attention

1) This is our input sentence*

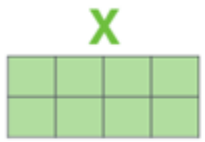
2) We embed each word*

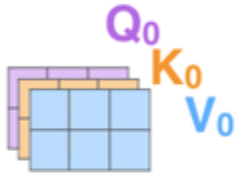
3) Split into 8 heads. We multiply X or R with weight matrices

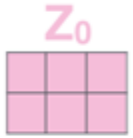
4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

Thinking Machines

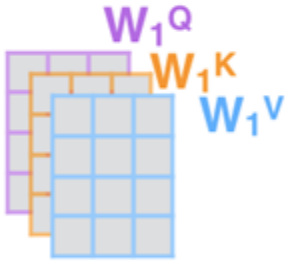


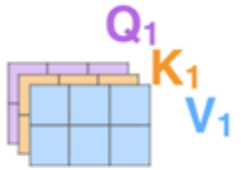


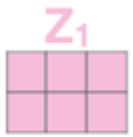


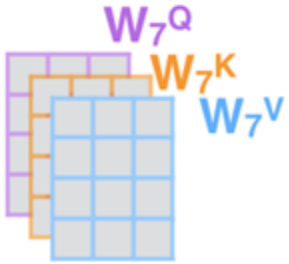
추가적인 W^O의 도입

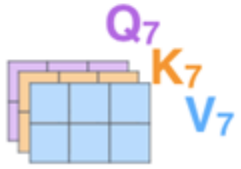
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

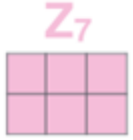


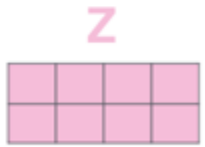












Transformer 활용

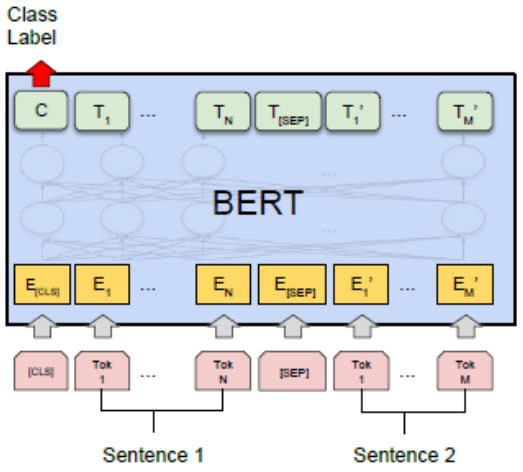
- Non-local Neural Networks
- BERT
- ViT
- UNITER
- Unicoder
- ViLBERT
- Oscar
- 12-in-1
- VILLA
- LXMERT
- VirTex
- DEIT
- DETR
- Deformable DETR
- Generative Pretraining from Pixels

Non-local Neural Networks(2017/11) CVPR 2018

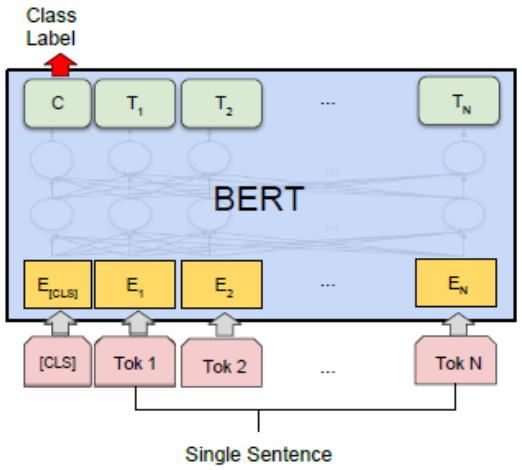
Figure 3. Examples of the behavior of a non-local block in res_3 computed by a 5-block non-local model trained on Kinetics. These examples are from held-out validation videos. The starting point of arrows represents one $\mathbf{x}_i$, and the ending points represent $\mathbf{x}_j$. The 20 highest weighted arrows for each $\mathbf{x}_i$ are visualized. The 4 frames are from a 32-frame input, shown with a stride of 8 frames. These visualizations show how the model finds related clues to support its prediction.

영상 처리를 위해
모든 프레임에 대해
Attention 계산

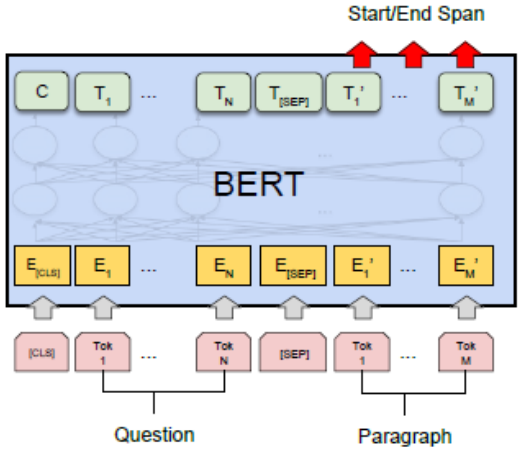
BERT(2018/10)



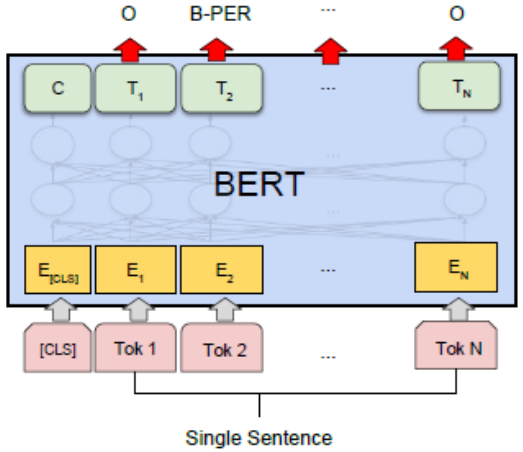
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



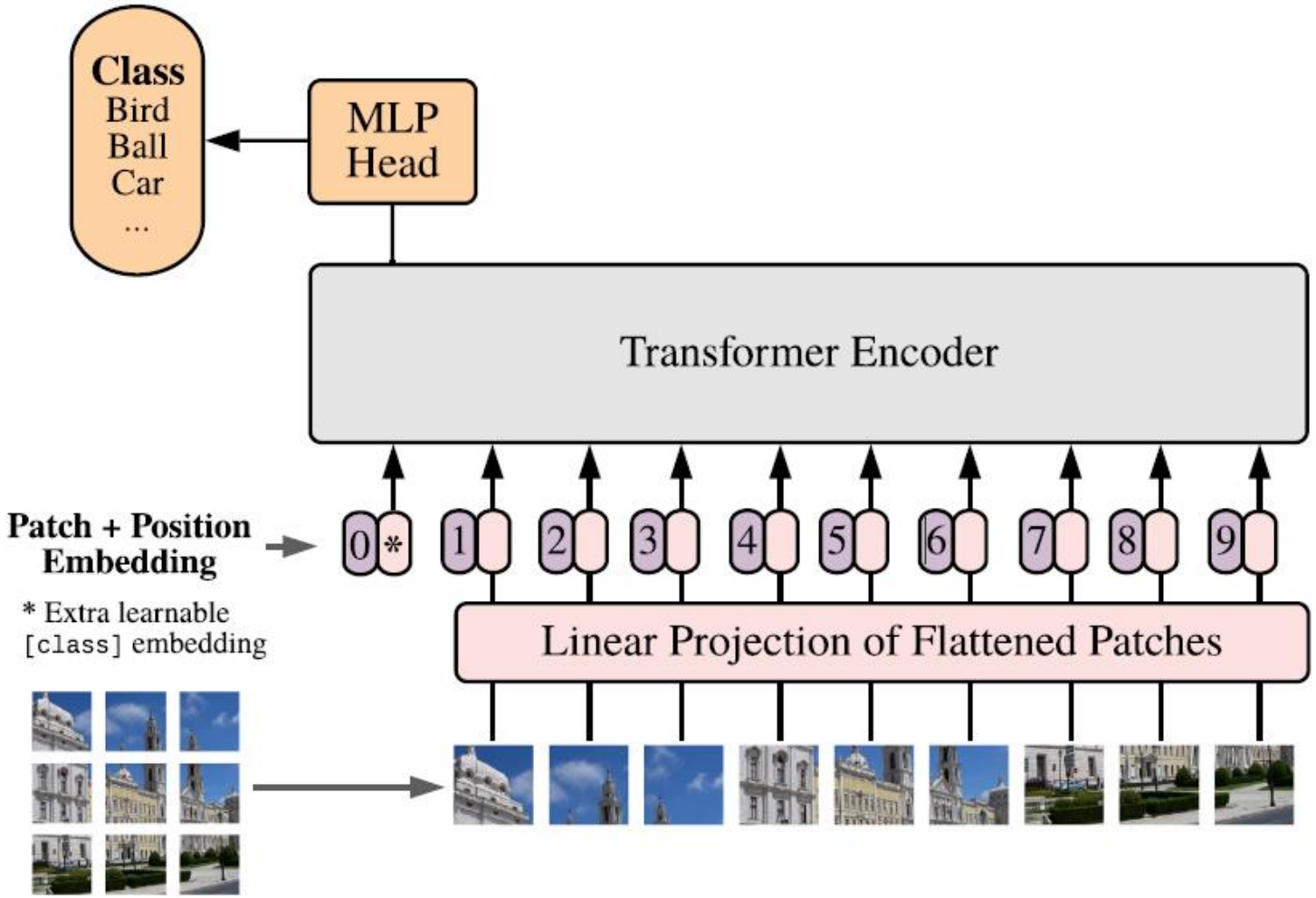
(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

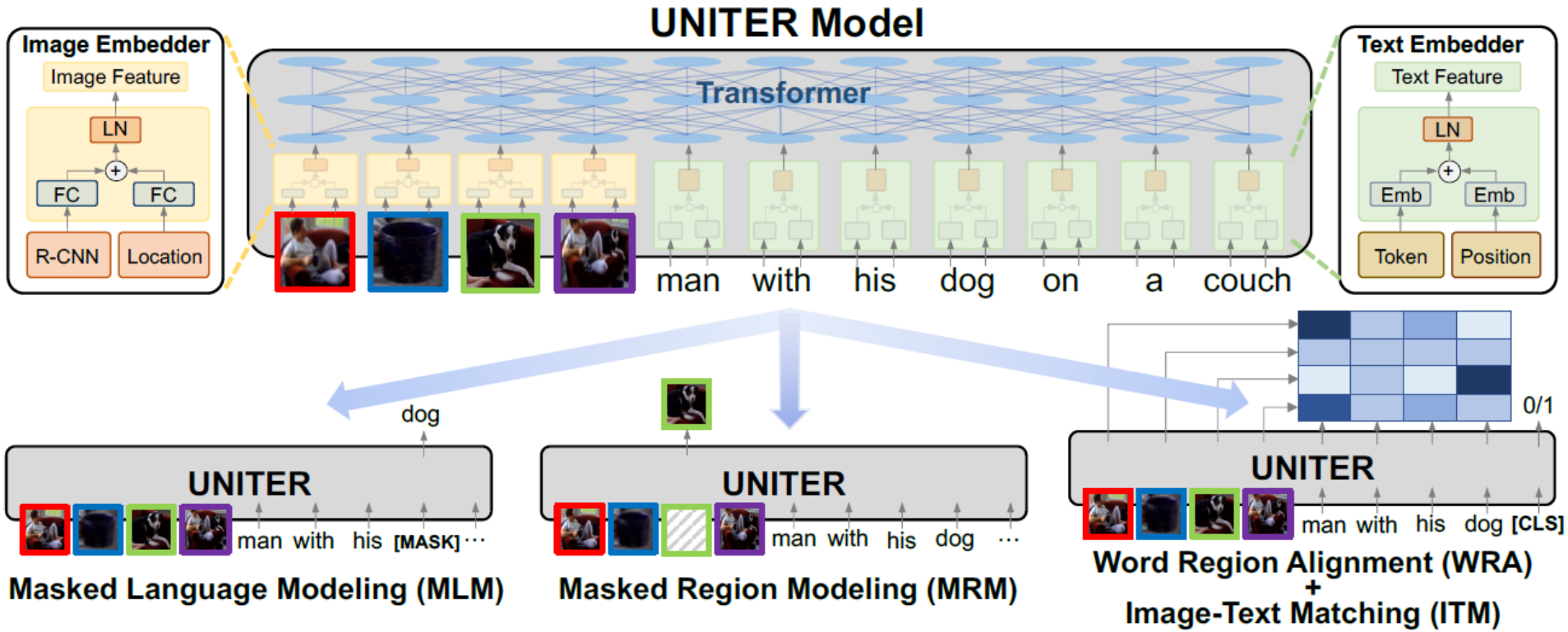
Encoder만 사용해서 단일 모델로 여러 가지 Task 처리

ViT(2020/10) ICLR 2021



Encoder만 사용해서
이미지를 조각내서 분류

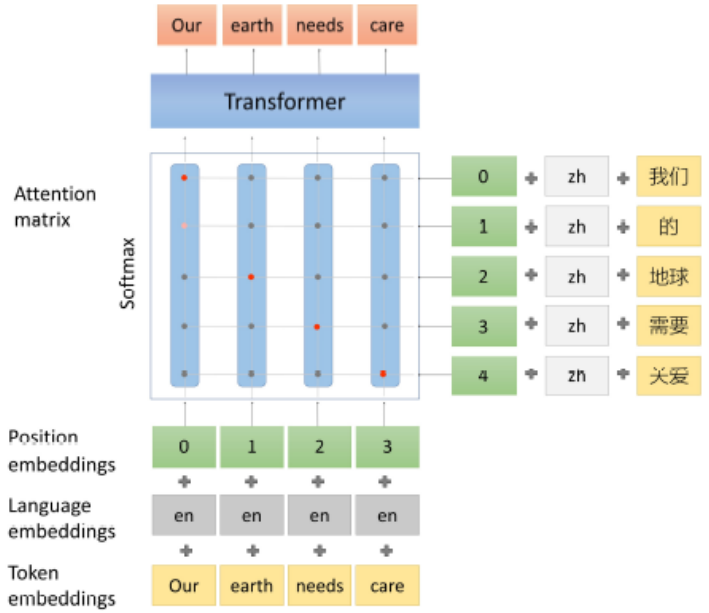
UNITER(2019/09) ECCV 2020



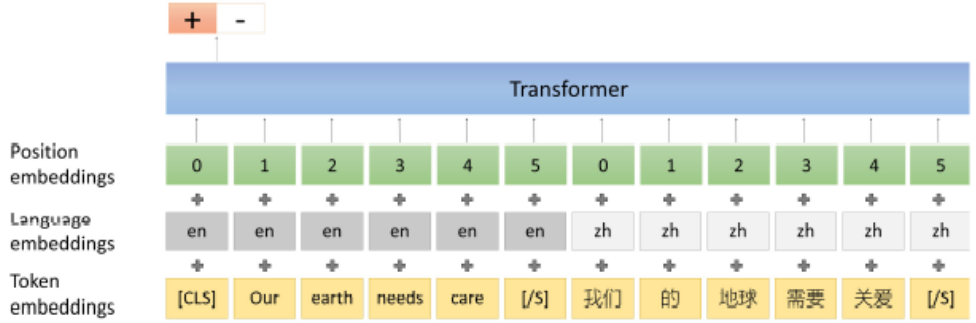
Encoder만 사용해서
여러 ViL* Task 처리

* Vision-and-Language

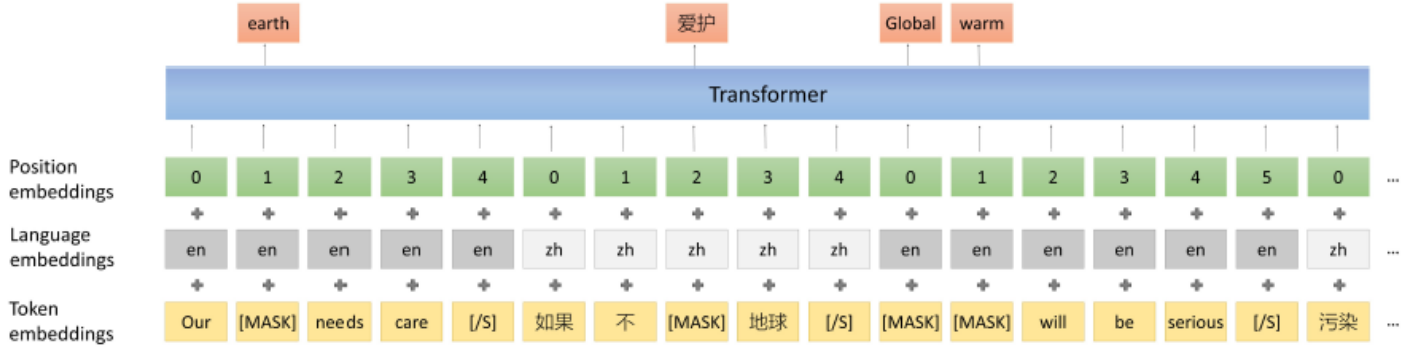
Unicoder(2019/09) EMNLP2019



(a) Cross-lingual word recovery



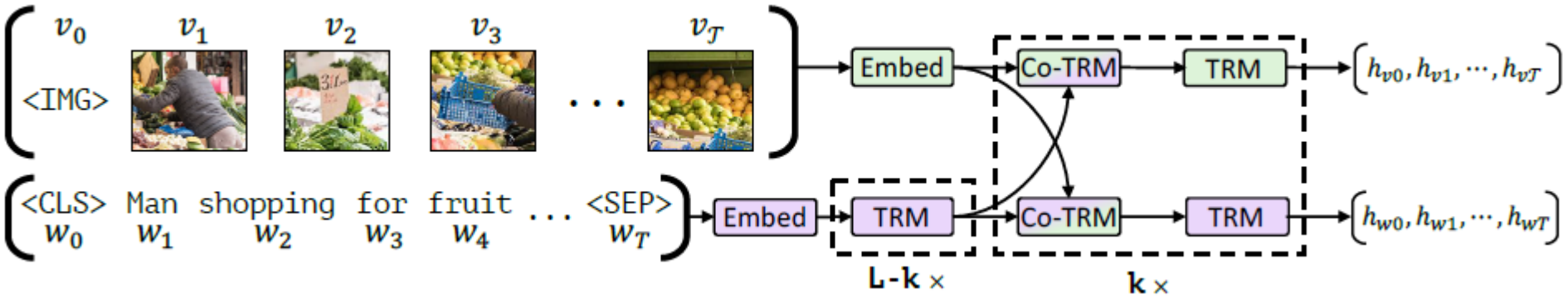
(b) Cross-lingual paraphrase classification

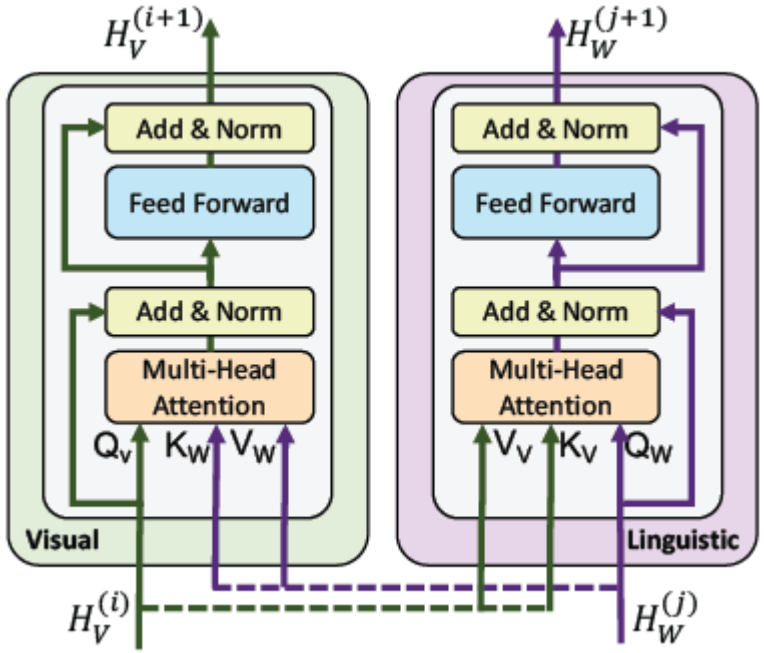


(c) Cross-lingual masked language model

Encoder만 사용해서
다중 언어 Task 처리

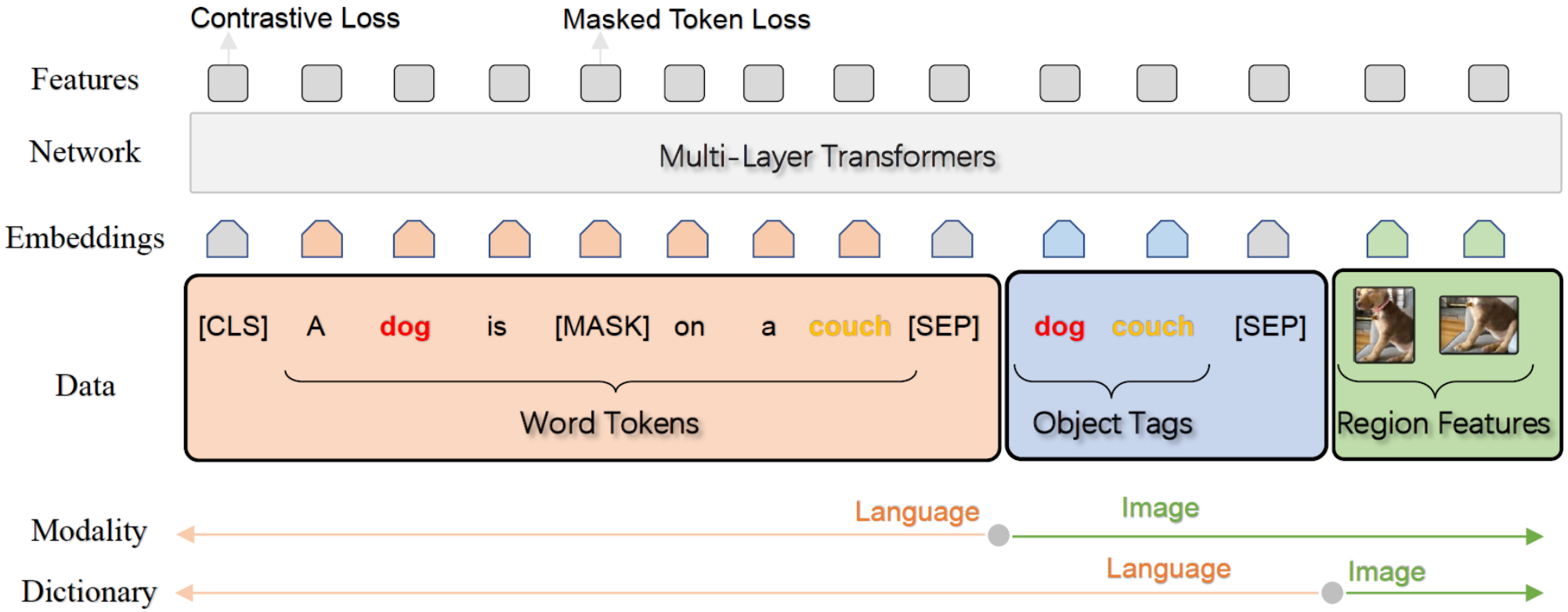
ViLBERT(2019/08)





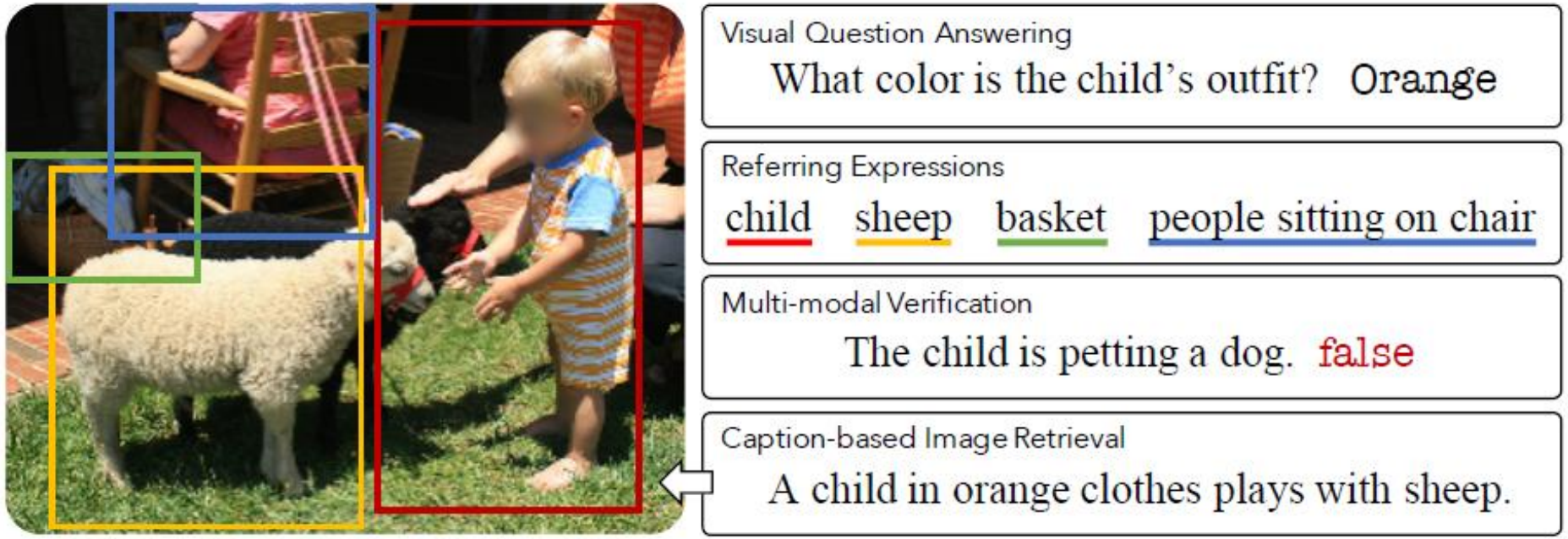
Co-Attention Transformer 레이어를 도입해 ViL Task 처리

Oscar(2020/04) ECCV 2020



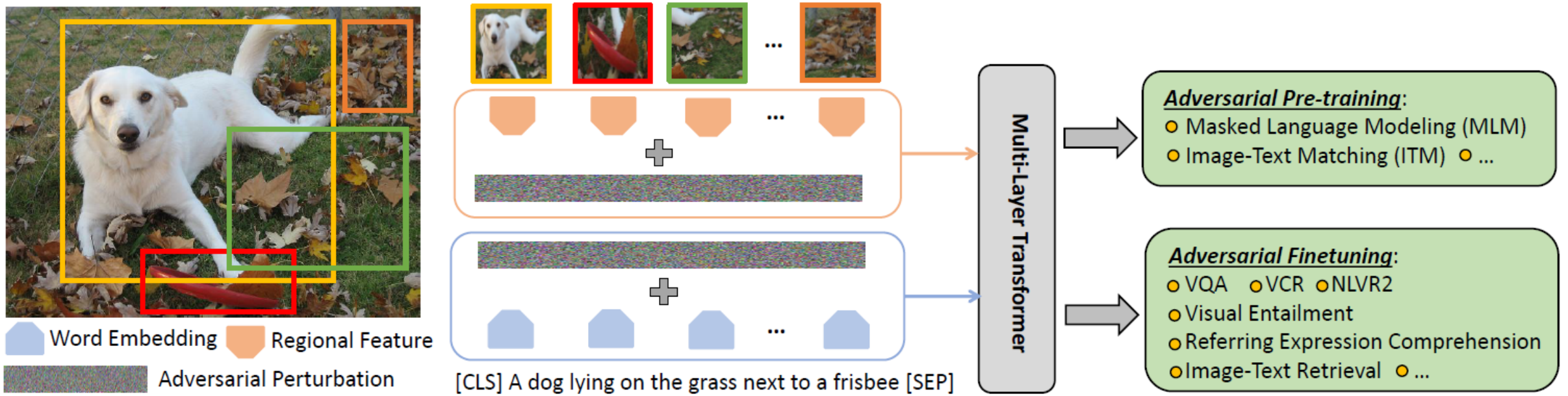
Object 태그와 Region 패치를 넣어서 ViL Task 처리

12-in-1(2019/12)



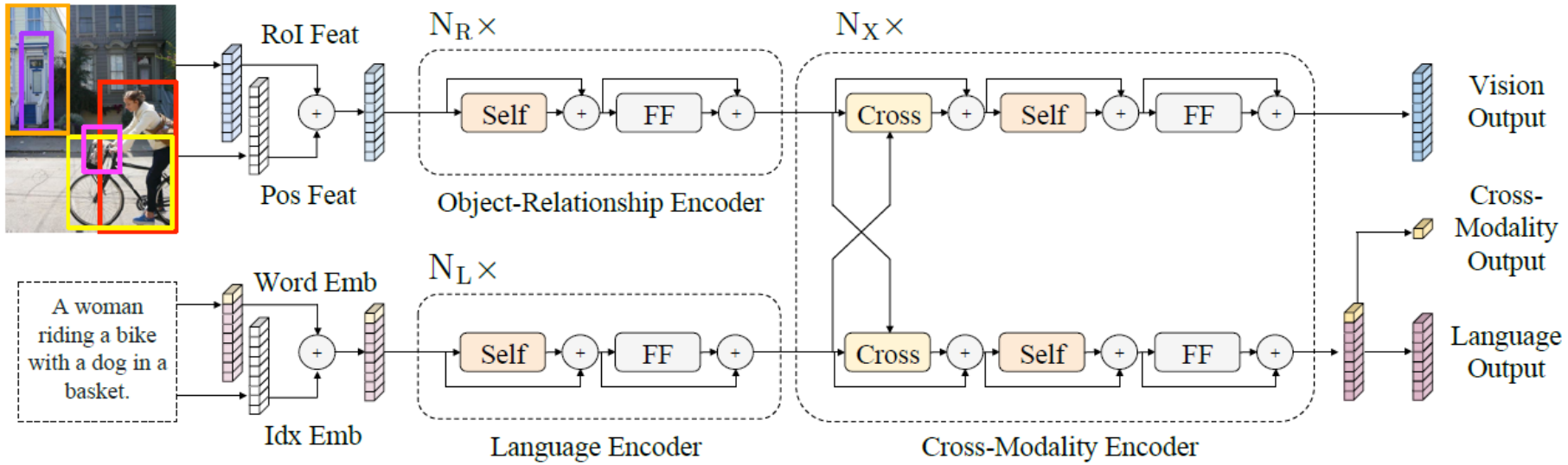
ViLBERT의 학습 규칙을 살짝 수정해서 12가지 ViL Task를 단일 모델로 처리

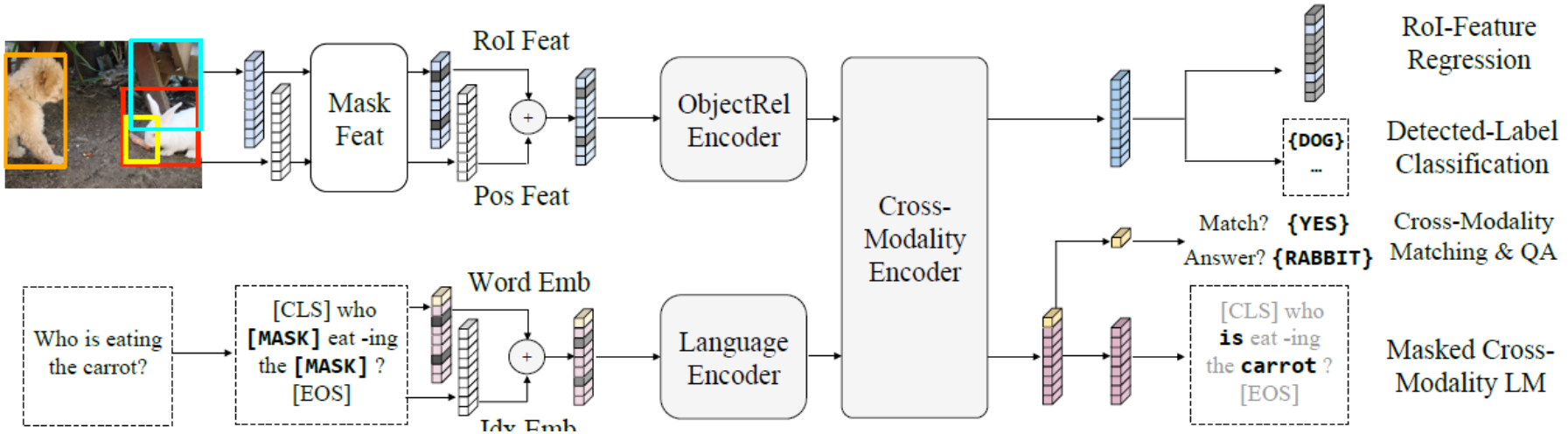
VILLA(2020/06) NeurIPS 2020



GAN처럼 Adversarial 학습을 통해 ViL Task 처리

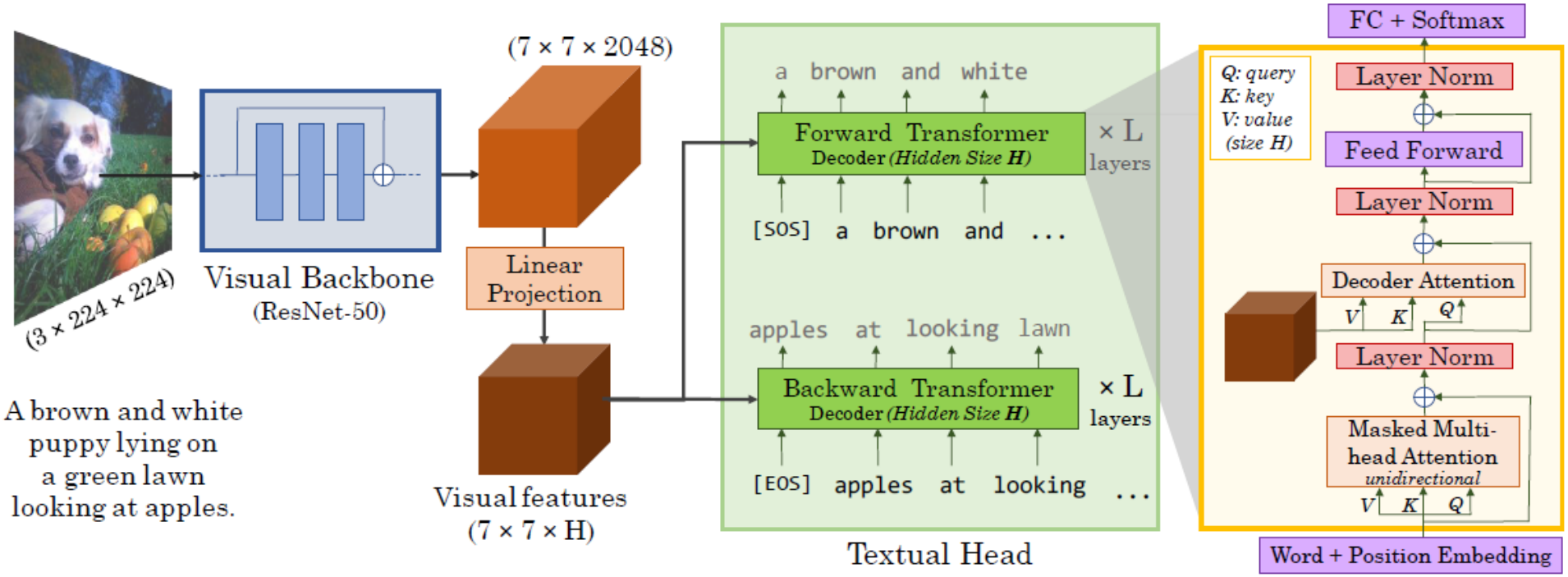
LXMERT(2019/08) EMNLP 2019





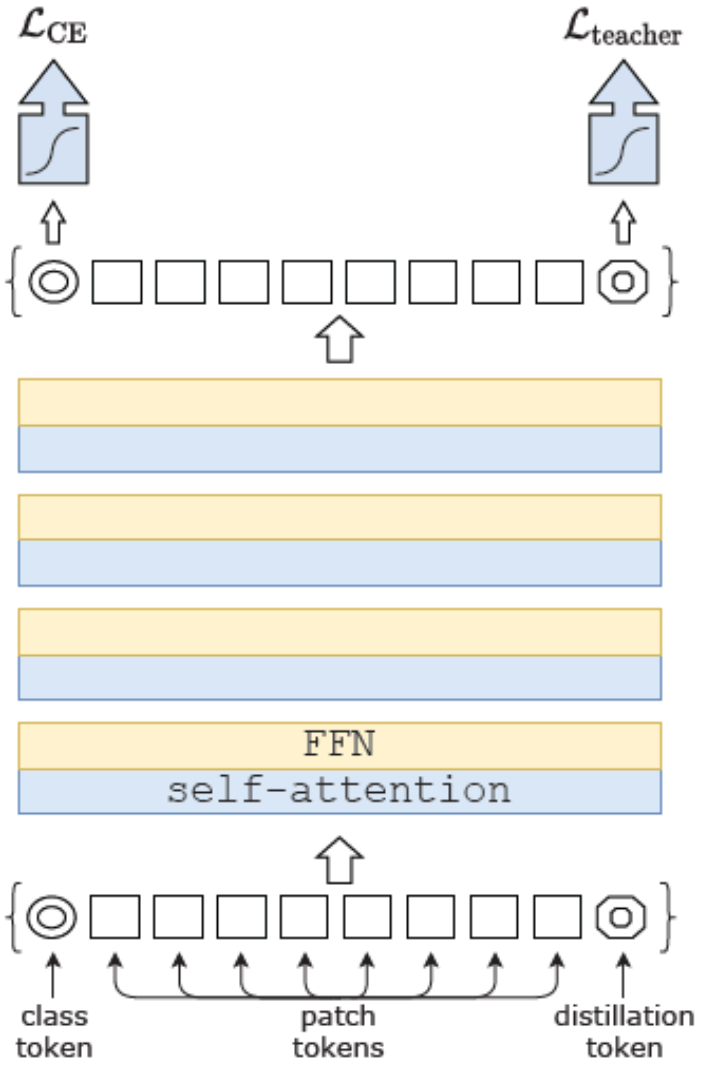
ViLBERT를 응용한 모델

VirTex(2020/06) CVPR 2021



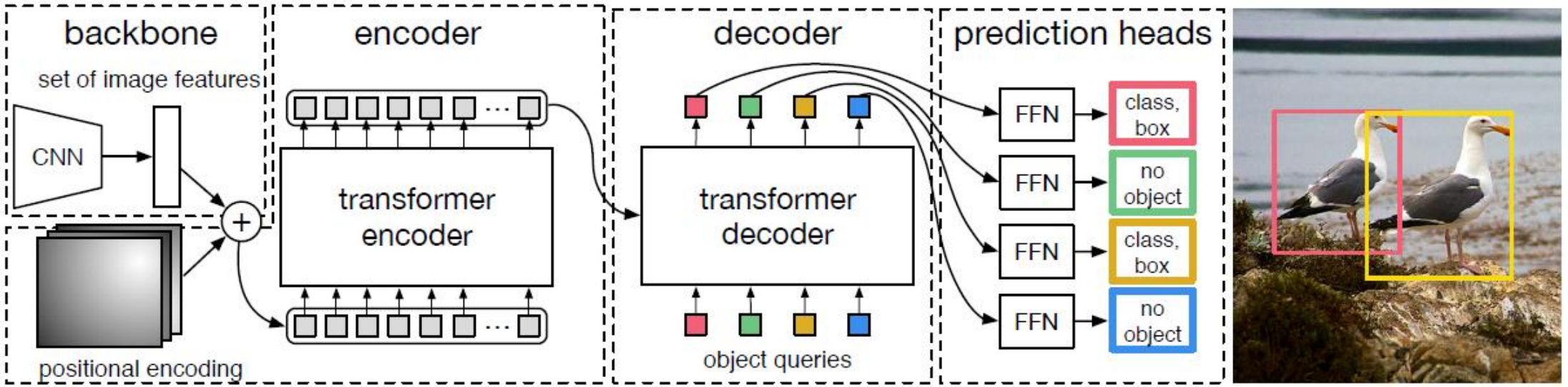
Text를 써서 Visual Backbone을 학습시켜 Image Task 처리

DEIT(2020/12)



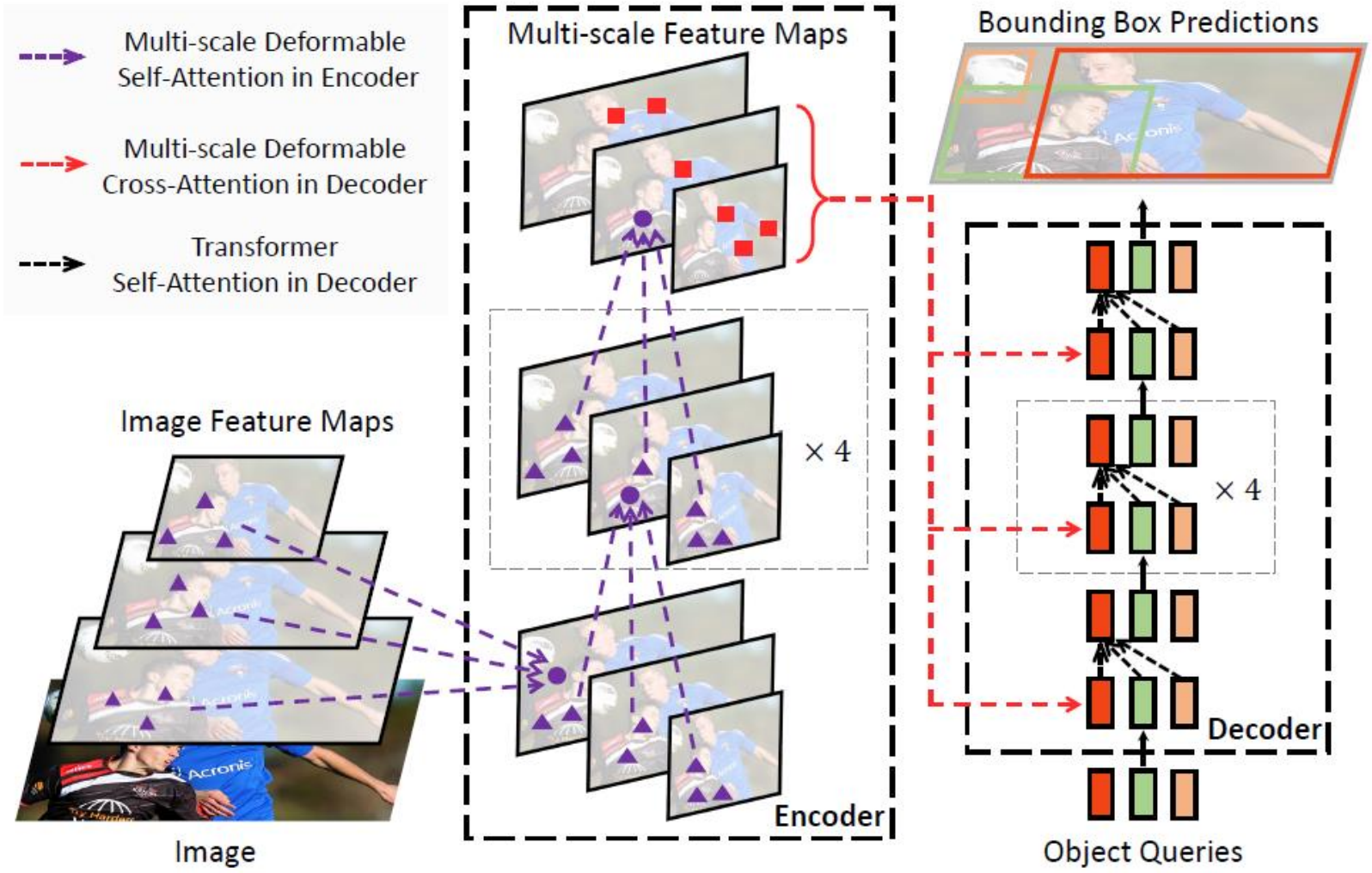
Pretrained 모델을 이용해서 적은 새로운 학습 데이터로 학습을 진행

DETR(2020/05)



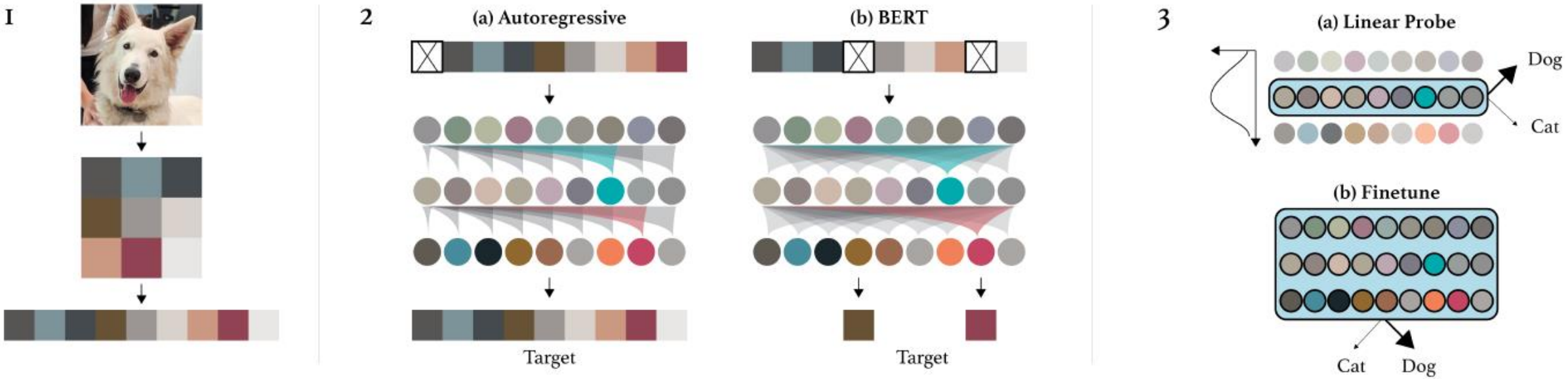
Transformer만으로
Detection Task 처리

Deformable DETR(2020/10) ICLR 2021



DETR을 개선하여 점차 이미지의 특정 부분에 집중해 나감

Generative Pretraining from Pixels(PMLR 2020)



Generative 학습이 Unsupervised Image Representation 학습을 위해 가장 나은 방법임을 재확인

Thank you!

Q & A