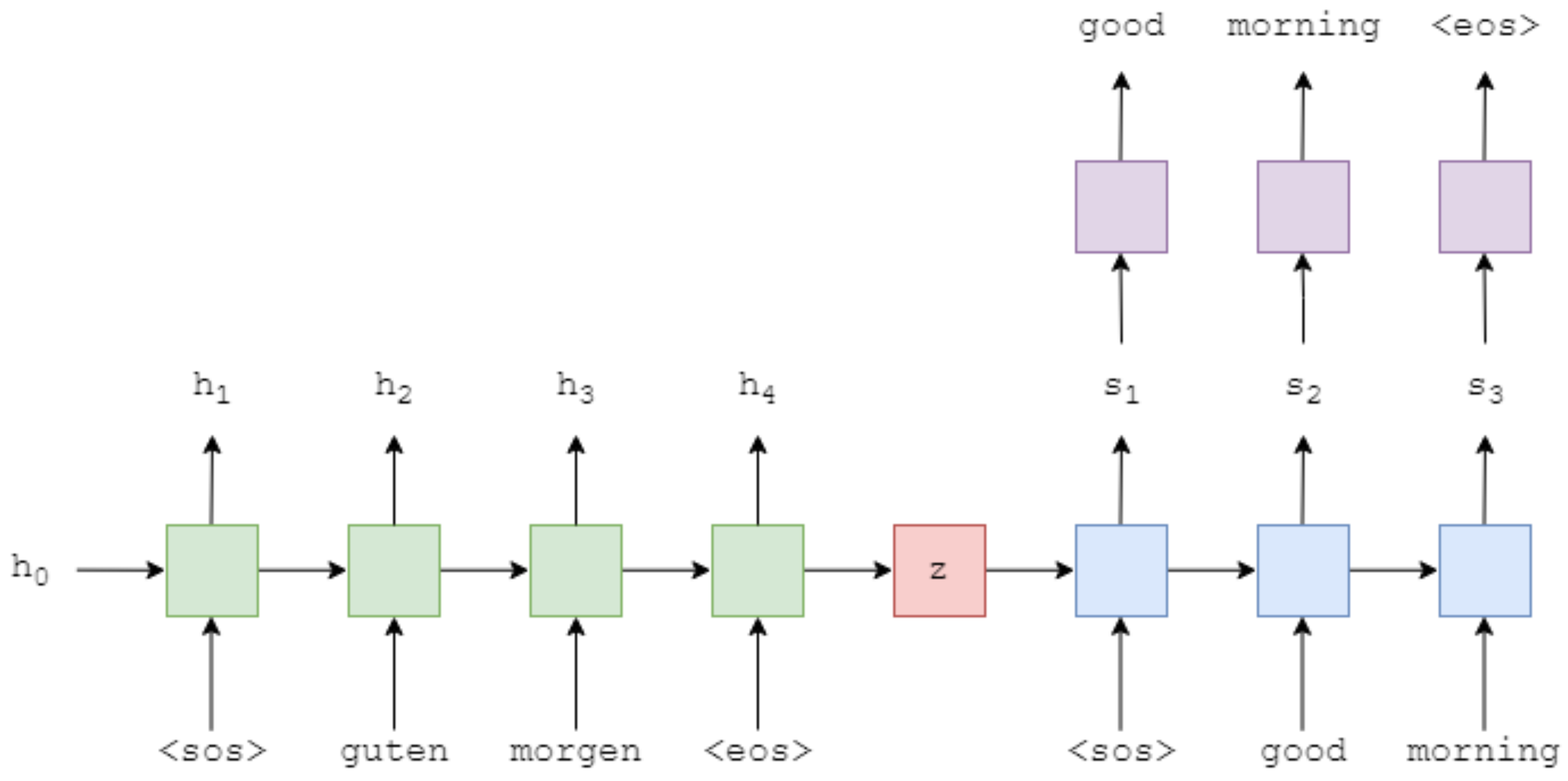
Special tokens

Базовые токены
EOS, BOS, UNK

- BOS/SOS - токен начала последовательности
- EOS - токен окончания последовательности
- UNK - токен для слов/подслов, которых нет в токенизаторе



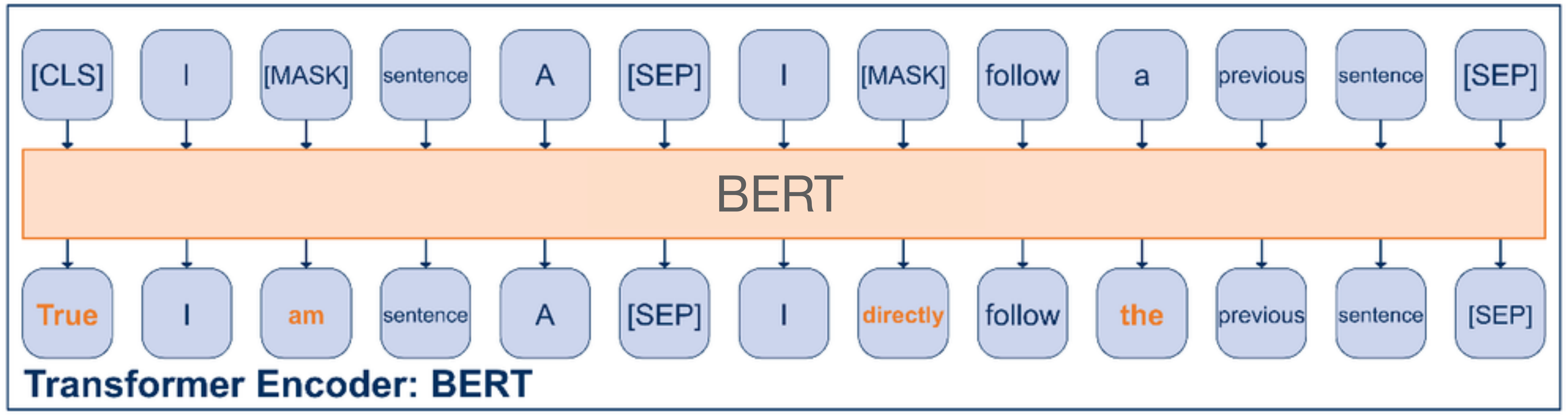
Базовые токены

BERT: SEP, CLS, MASK

- CLS - токен классификации, в выходе энкодера на его месте лежат эмбединги для классификации

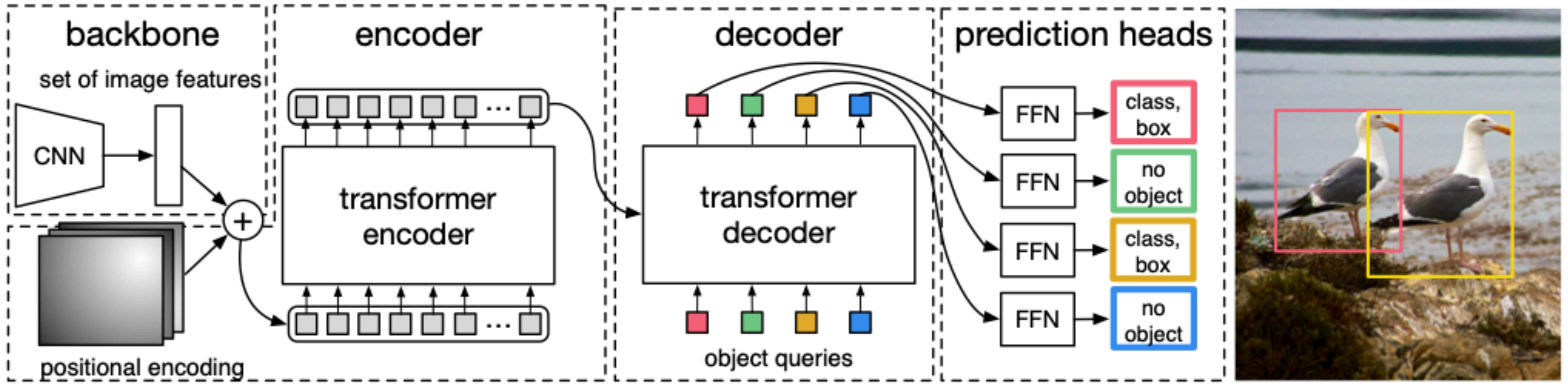
- SEP - токен для разделения двух предложений

- MASK - токен для маски во время обучения



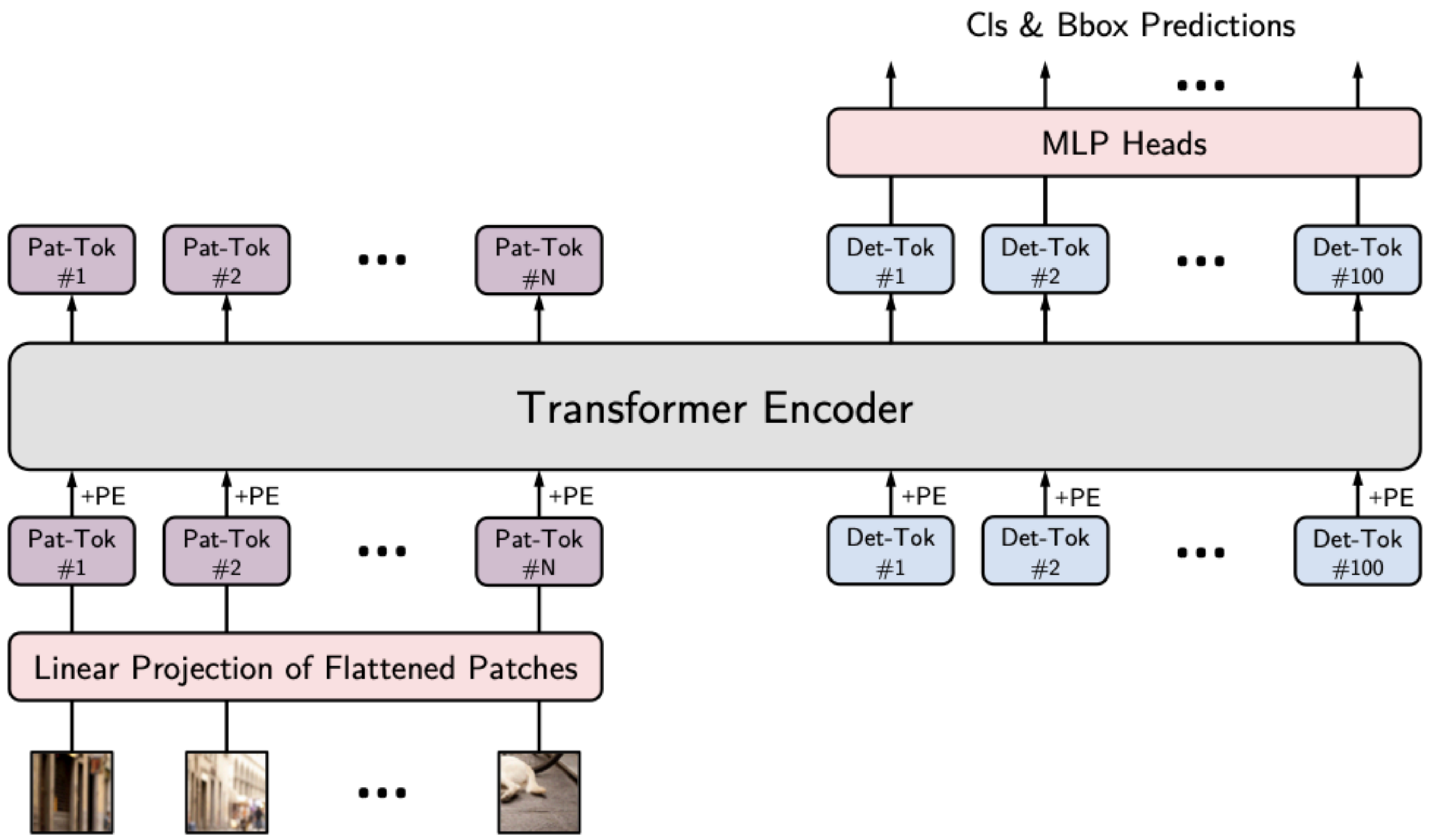
Токены детекции
DETR

- CNN backbone - learn 2D Representation of image: C x H x W
- Flatten + Linear + PE: d x HW
- Decoder input: Learned positional embeddings (object queries)



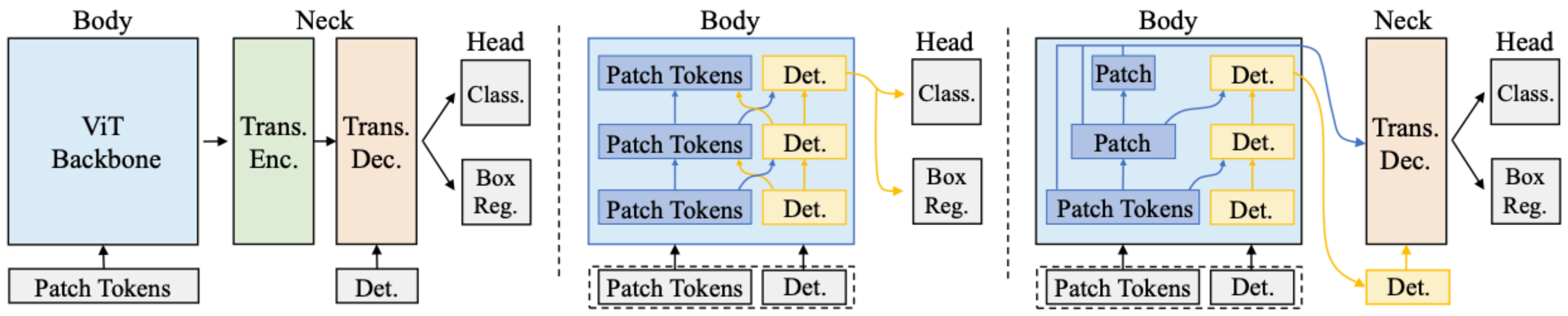
Токены детекции
YOLOS

- Based on ViT
- Encoder-only
- Also learnable DET tokens



Токены детекции

ViDT



(a) DETR (ViT). (b) YOLOS. **(c) ViDT (Ours)**.

Figure 2. Pipelines of fully transformer-based object detectors. DETR (ViT) means Detection Transformer that uses ViT as its body. The proposed ViDT synergizes DETR (ViT) and YOLOS and achieves best AP and latency trade-off among fully transformer-based object detectors.

Использование служебных токенов

- Object binding - место для эмбедингов выхода
- Punctuation - сслужебная информация о начале/конце последовательности/предложения

Memory token

Мотивация

- Combination of local and global information in the same vector results in blurring global features and make in harder to acces them
- Poor scaling of attention span

Memory token

Memory Transfomer

1. **Self-attention**. Calculate normalized sum of input X with multi-head attention $MH(Q, K, V)$ between all elements of the sequence:

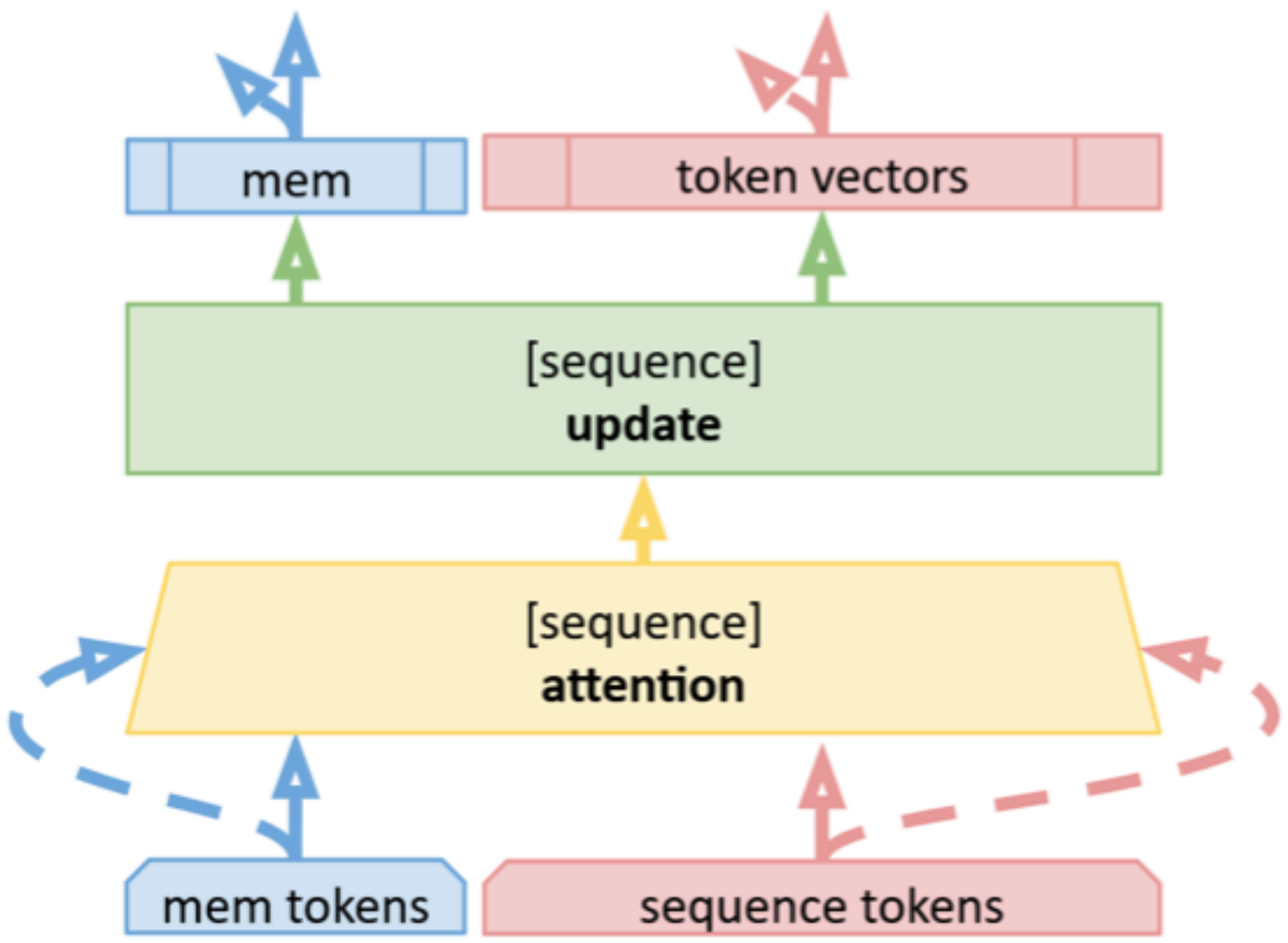
$$A = LN(X + MH(X, X, X)). \tag{1}$$

2. **Update**. For every element of the sequence update aggregated representation A with FF feed-forward sub-layer then add skip connection and normalize:

$$H = LN(A + FF(A)). \tag{2}$$

$$X^{mem+seq} = [X^{mem}; X^{seq}] \in \mathbb{R}^{(n+m)\times d}, X^{mem} \in \mathbb{R}^{m\times d}, X^{seq} \in \mathbb{R}^{n\times d}.$$

b. Mem Transformer layer



Memory token

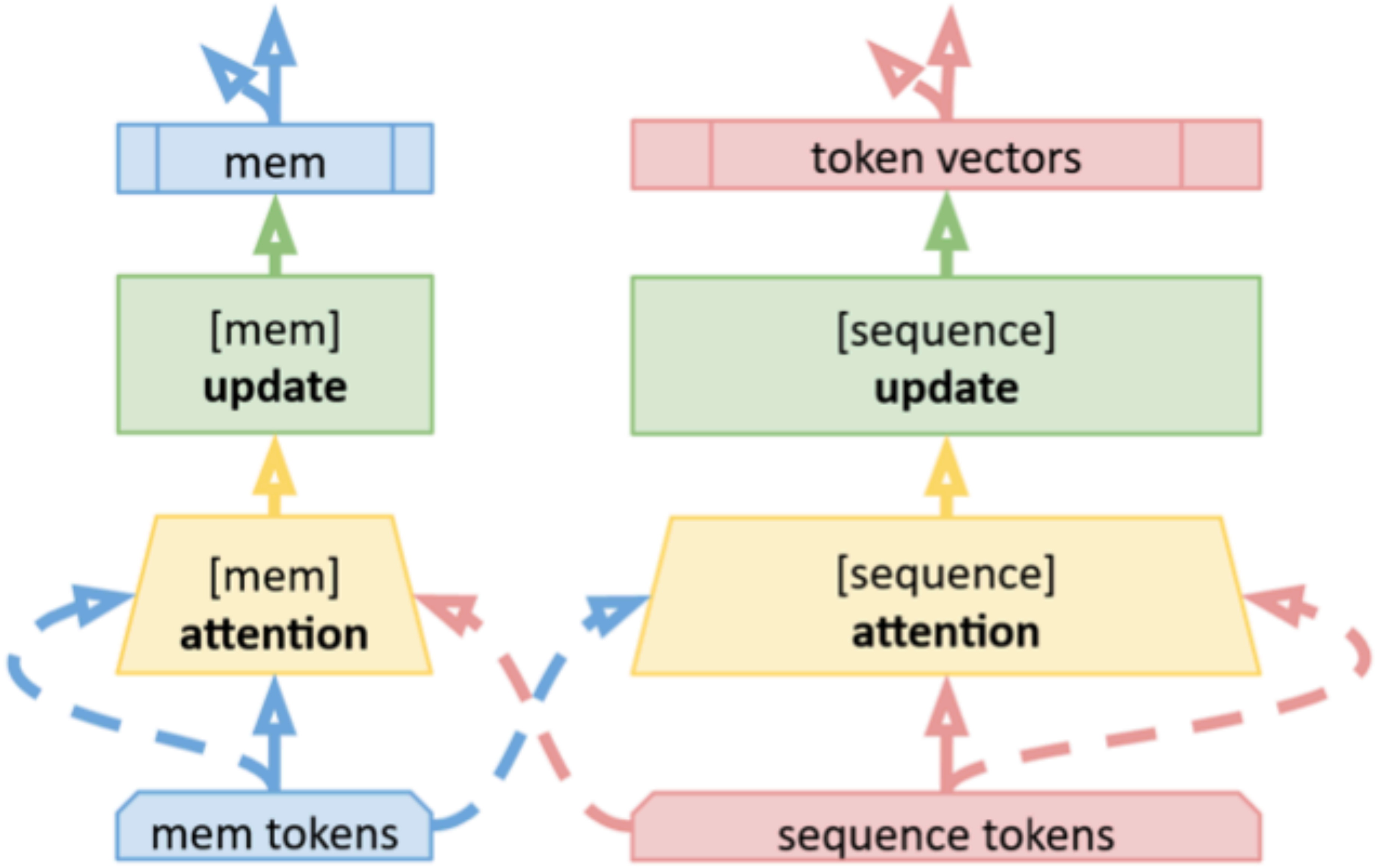
Memory Transformer: MemCTRL

$$A^{mem} = LN(X^{mem} + MH^{mem}(X^{mem}, X^{mem+seq}, X^{mem+seq})),$$
$$H^{mem} = LN(A^{mem} + FF^{mem}(A^{mem})).$$

Sequence representation is updated as:

$$A^{seq} = LN(X^{seq} + MH^{seq}(X^{seq}, X^{mem+seq}, X^{mem+seq})),$$
$$H^{seq} = LN(A^{seq} + FF^{seq}(A^{seq})).$$

c. MemCtrl Transformer layer



Memory token

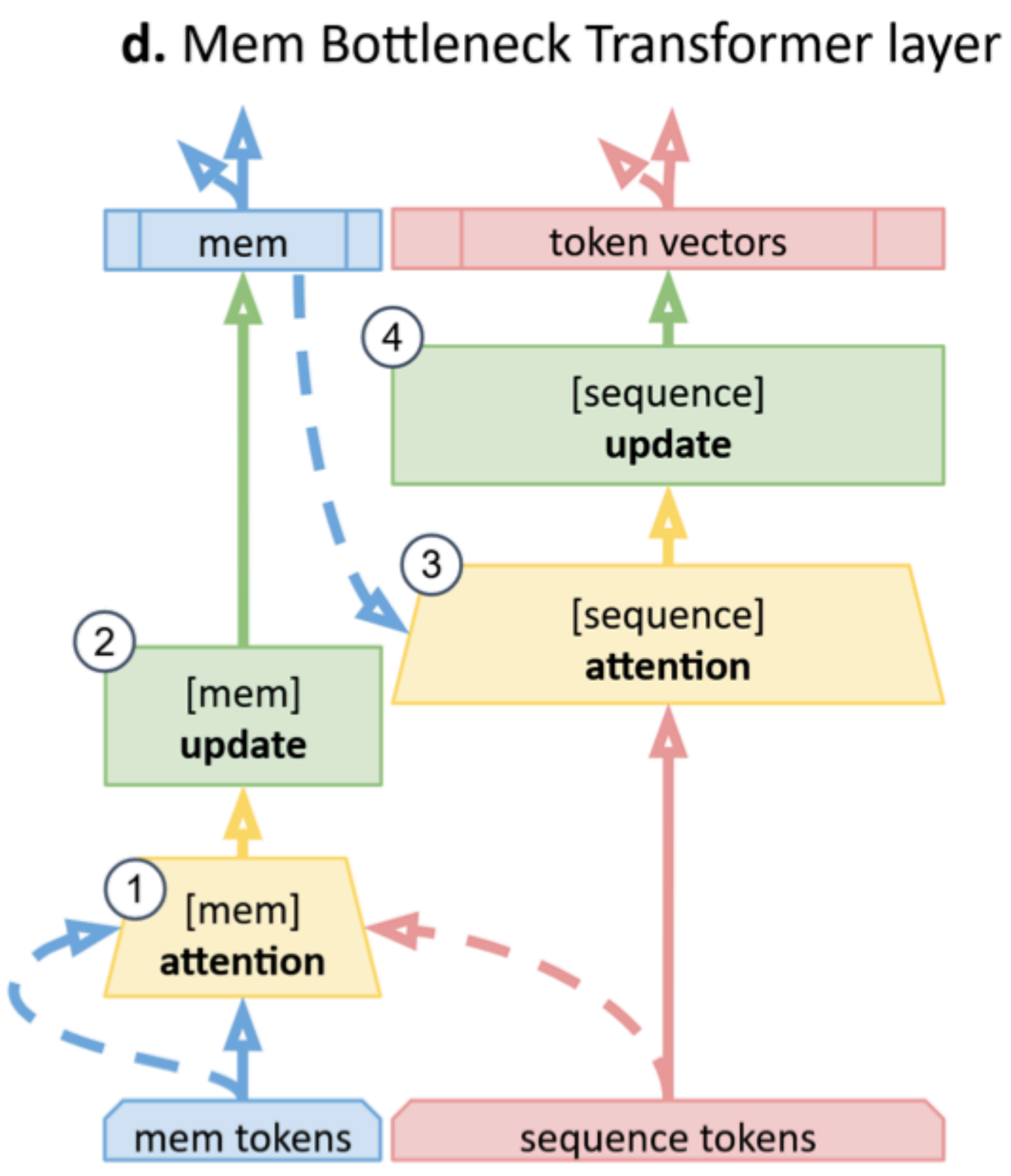
Memory Transformer:

1. Memory update. First, calculate attention between every memory token and full sequence of memory X^{mem} and input X^{seq} (see Step 1 on the fig. 1d), and update memory token representations (see Step 2 on the fig. 1d):

$$A^{mem} = LN(X^{mem} + MH^{mem}(X^{mem}, X^{mem+seq}, X^{mem+seq})),$$
$$H^{mem} = LN(A^{mem} + FF^{mem}(A^{mem})).$$

2. Sequence update. Calculate attention between sequence and memory (Step 3 on the fig. 1d), and update sequence token representations (Step 4 on the fig. 1d):

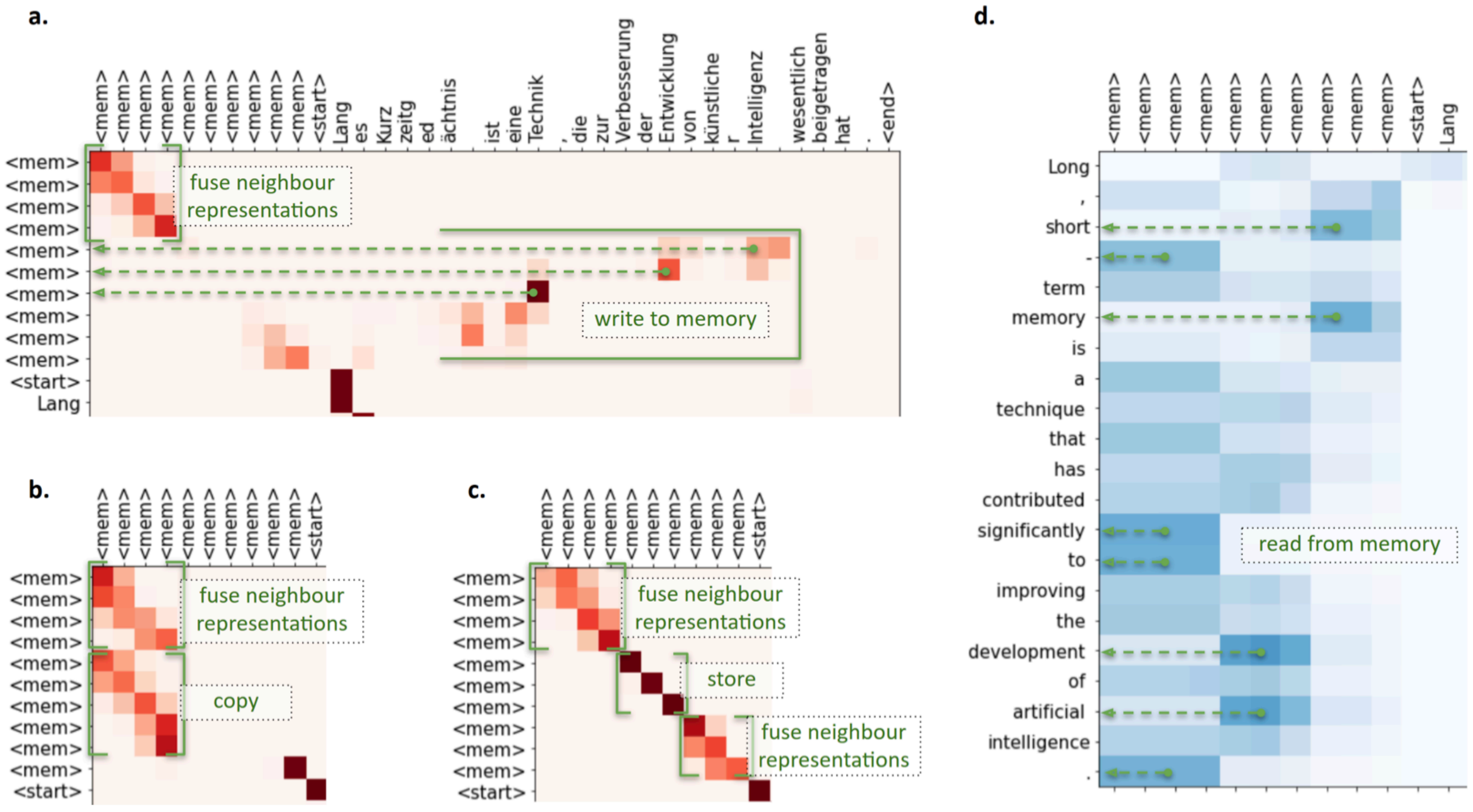
$$A^{seq} = LN(X^{seq} + MH^{seq}(X^{seq}, H^{mem}, H^{mem})),$$
$$H^{seq} = LN(A^{seq} + FF^{seq}(A^{seq})).$$



Memory token

Memory Transformer: Results

	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE
BERT-base	**62.9**	**92.7**	90.2/85.8	86.0/85.8	86.6/89.8	83.0/**83.5**	90.5	65.0
5mem	61.3	92.4	90.4/86.4	86.0/85.8	86.8/90.1	82.7/83.3	90.7	**68.0**
5mem+pool	62.1	92.3	89.4/84.8	85.8/85.6	86.9/**90.2**	**83.3**/83.3	**90.8**	60.2
10mem	60.6	92.5	**91.3/87.6**	86.6/86.4	86.4/89.8	82.8/83.3	90.5	66.8
10mem+pool	62.6	92.6	90.2/86.0	**86.7/86.5**	**87.1/90.2**	83.1/83.0	90.7	61.2
20mem	60.9	92.4	91.2/87.5	86.4/86.2	86.8/90.1	82.8/83.1	90.7	65.3

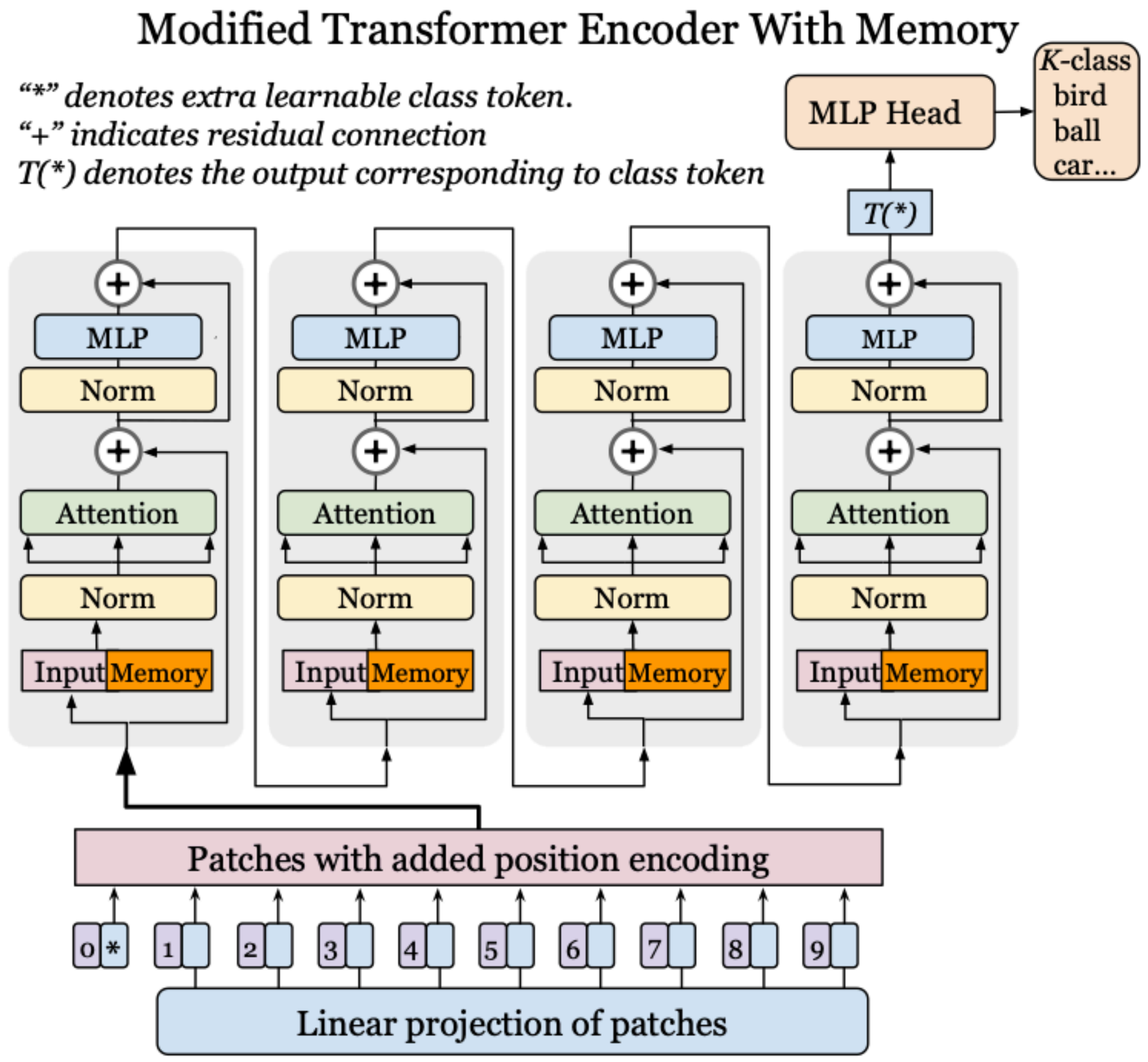


Memory token

Fine-tuning Image Transformers using Learnable Memory

- Memory token provide contextual information useful for specific datasets

.



Memory token

Attention masking for computation reuse

Q \ K	INP	CLS	C_1	C_2		C_k	M_1	M_2		M_k
INP	✓	✓	-	-		-	-	-		-
CLS	✓	✓	-	-	…	-	-	-	…	-
C_1	✓	✓	✓	-		-	✓	-		-
C_2	✓	✓	★	✓		-	★	✓		-
			…				…			
C_k	✓	✓	★	★		✓	★	★		✓

Table 1. Attention mask for model extension and concatenation. Here ✓ indicates that corresponding token type Q (query) attends to token type K (key), - that it does not attend, and ★ indicates that attention is used for model extension, but not for concatenation. For brevity, we denoted CLS-K as C_k and MEM-K as M_k and omitted memory rows since they do not attend to other tokens. See Sec. 2.2 for details.