



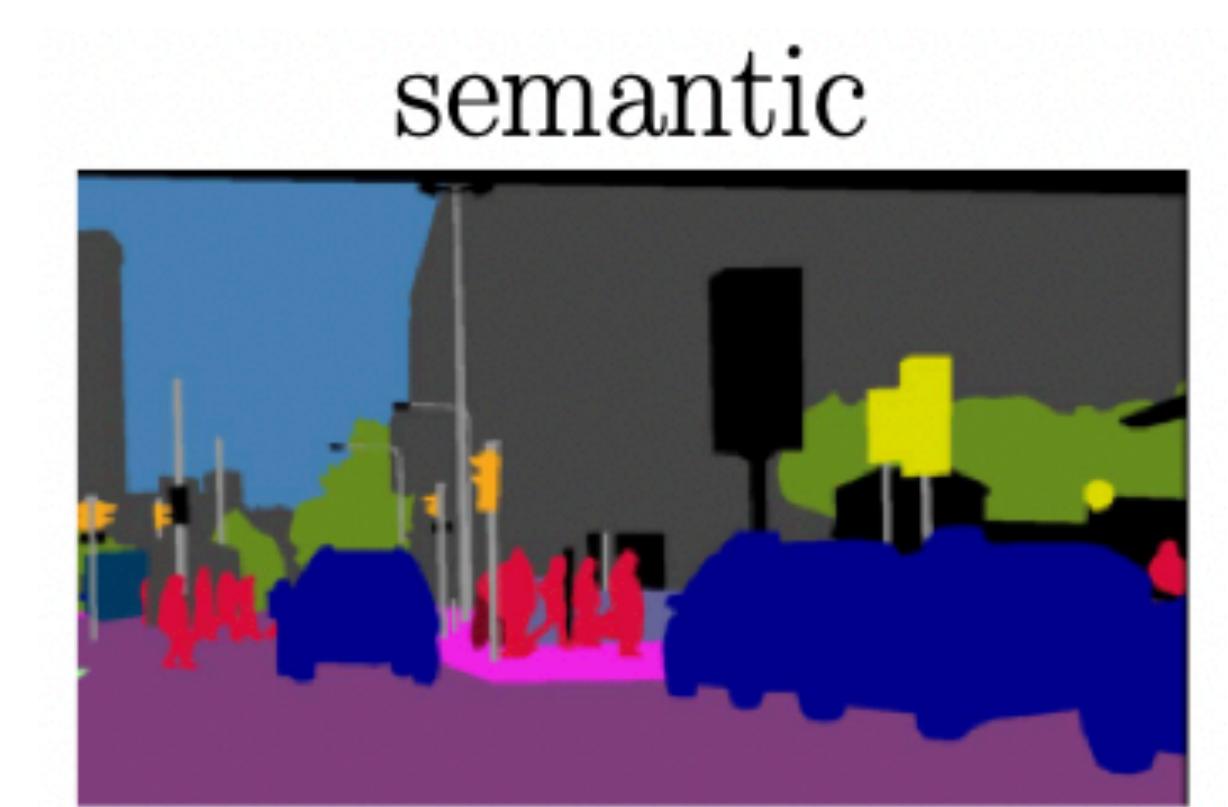
DEEP  
LEARNING  
INDABA



# Panoptic Segmentation with Transformers Tutorial

Mennatullah Siam

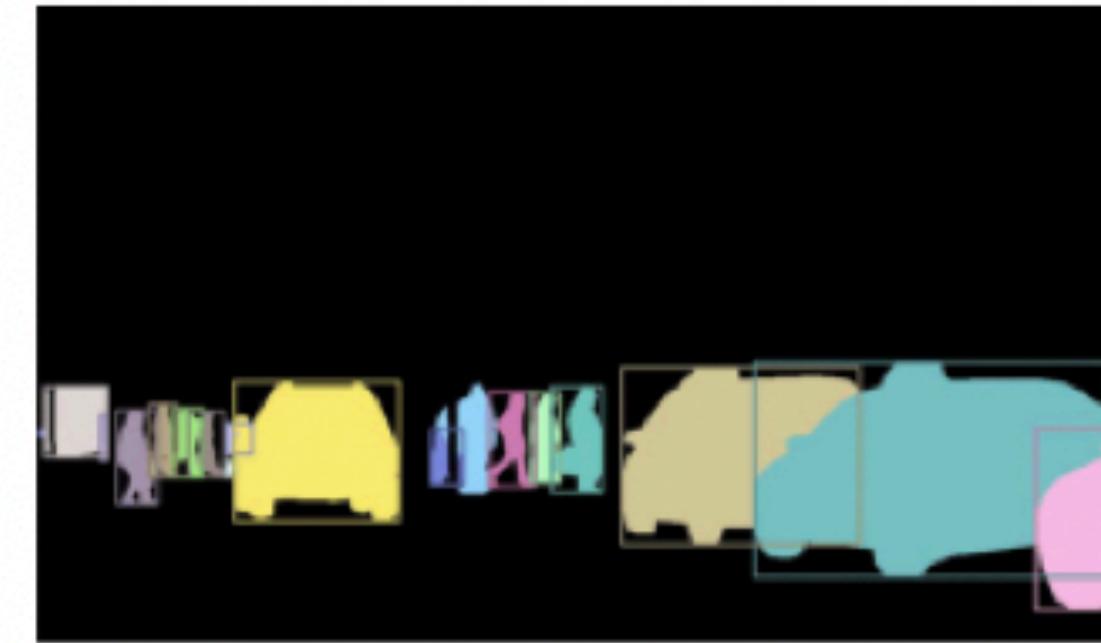
# Semantic/Instance/Panoptic Segmentation



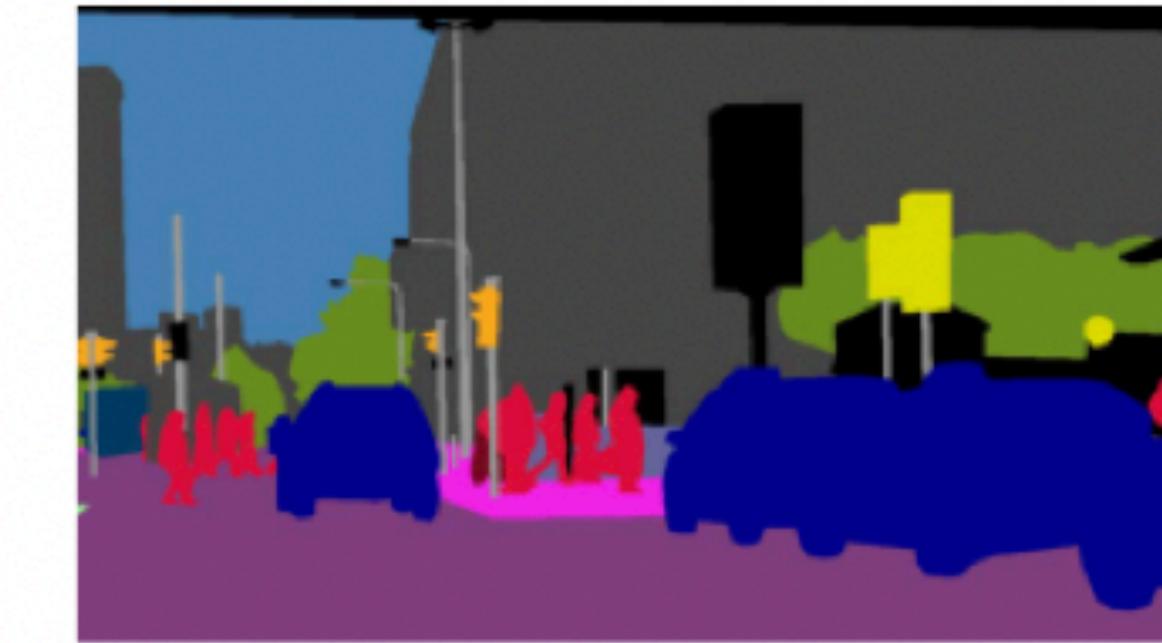
semantic

# Semantic/Instance/Panoptic Segmentation

instance



semantic



# Semantic/Instance/Panoptic Segmentation



**Holistic Scene Understanding**

# Transformers

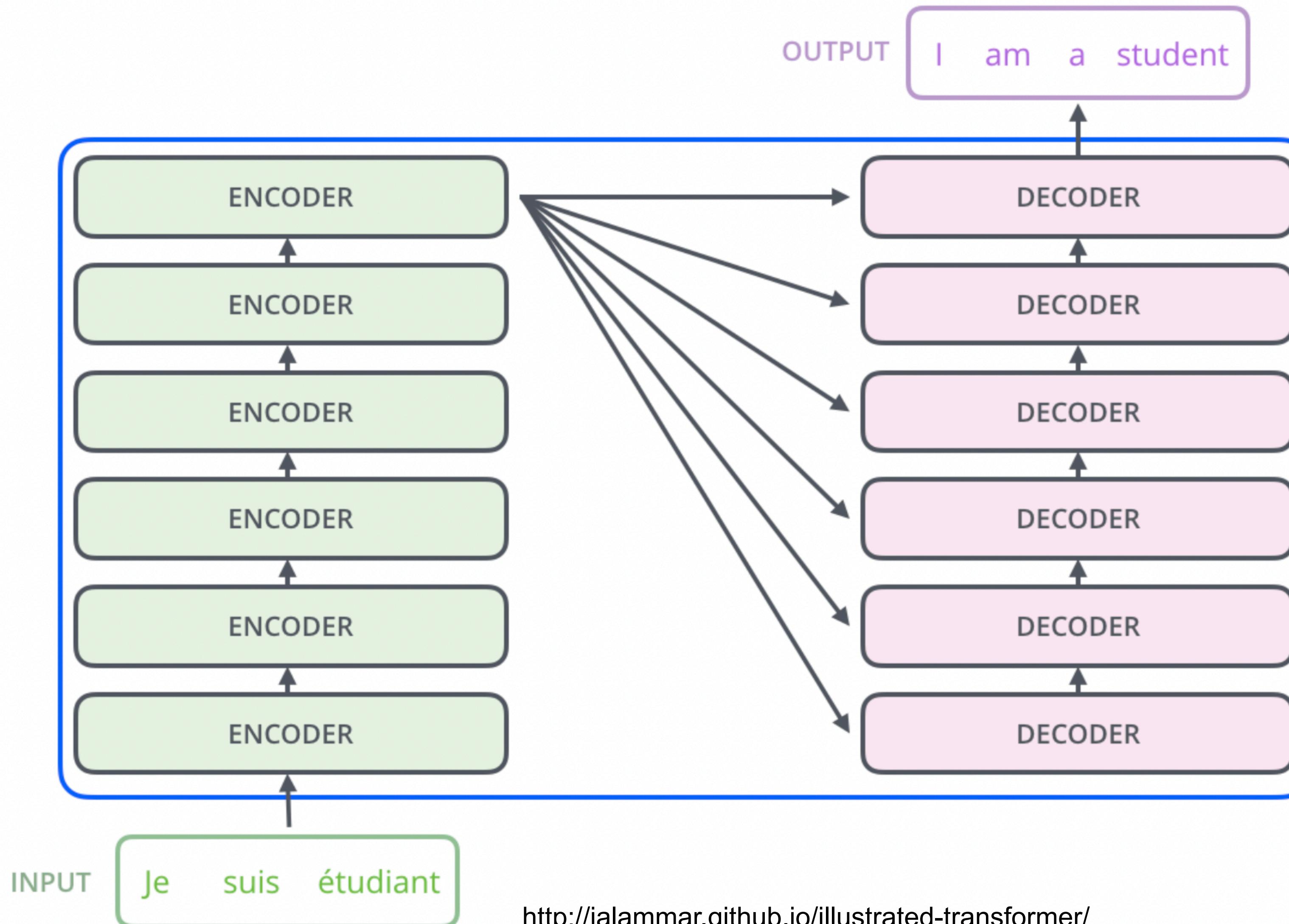
Example



<http://jalammar.github.io/illustrated-transformer/>

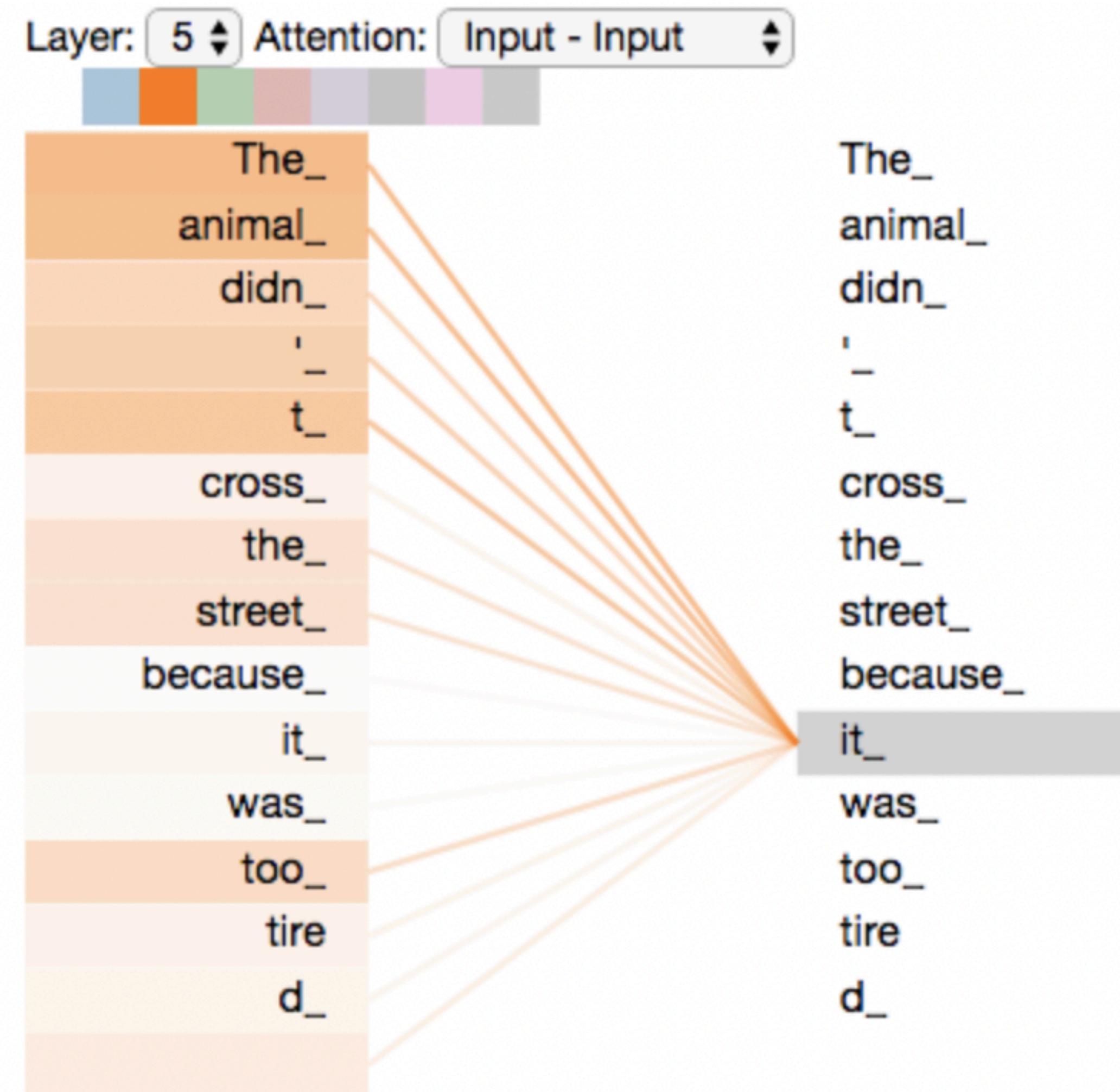
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

# Transformers



<http://jalammar.github.io/illustrated-transformer/>

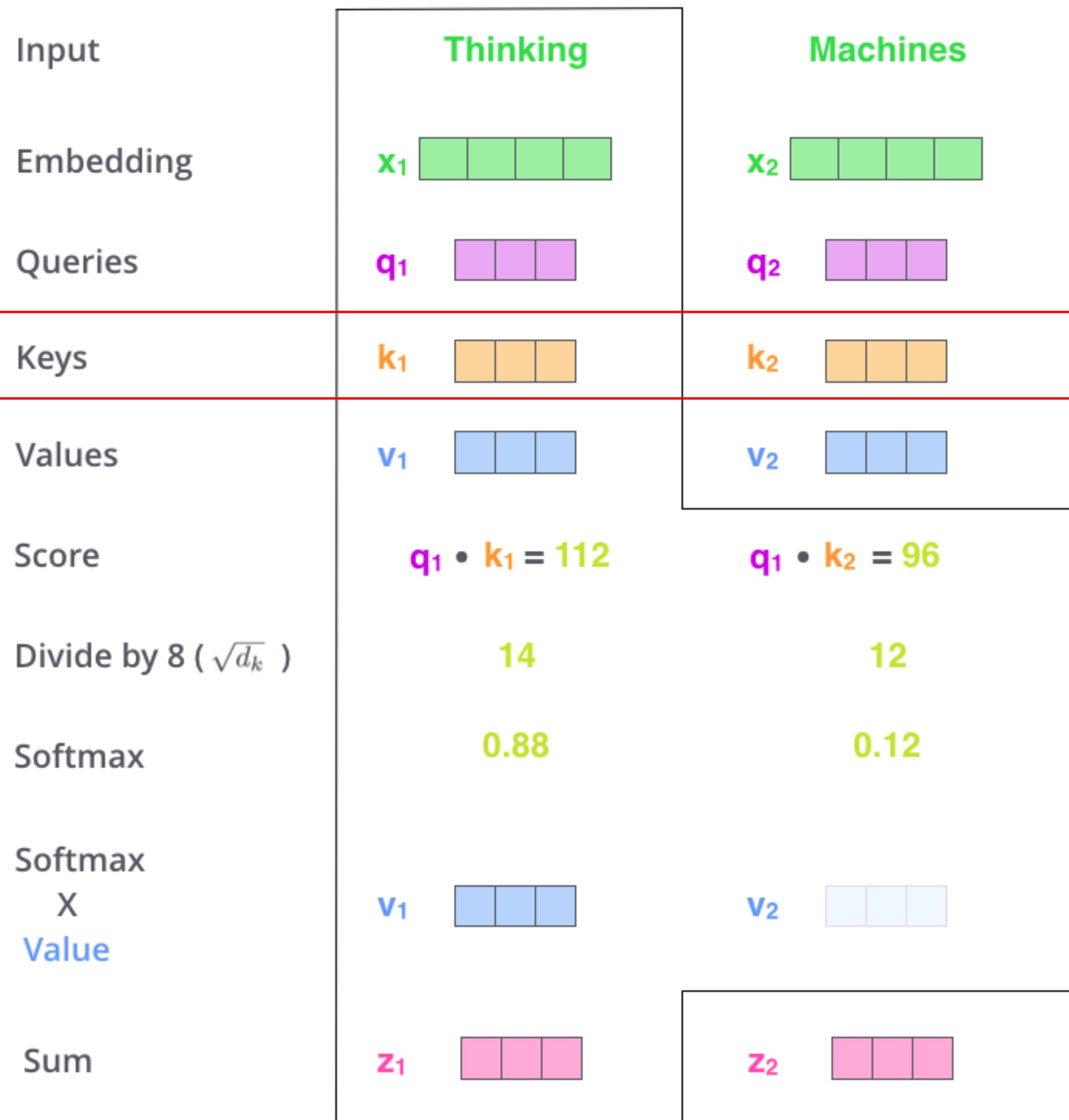
# Self Attention



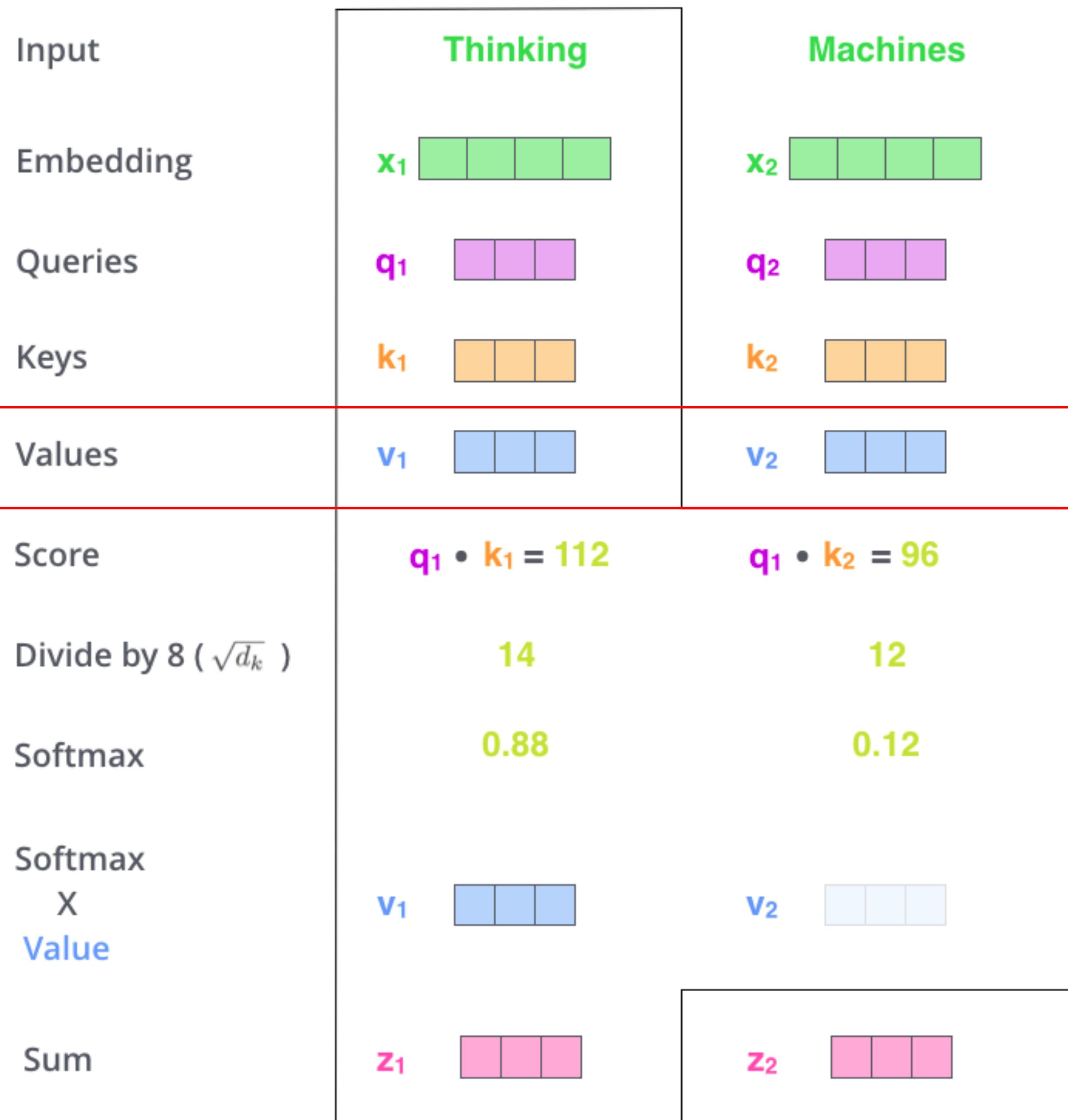
<http://jalammar.github.io/illustrated-transformer/>

Think of this as your  
Dictionary keys used for  
addressing

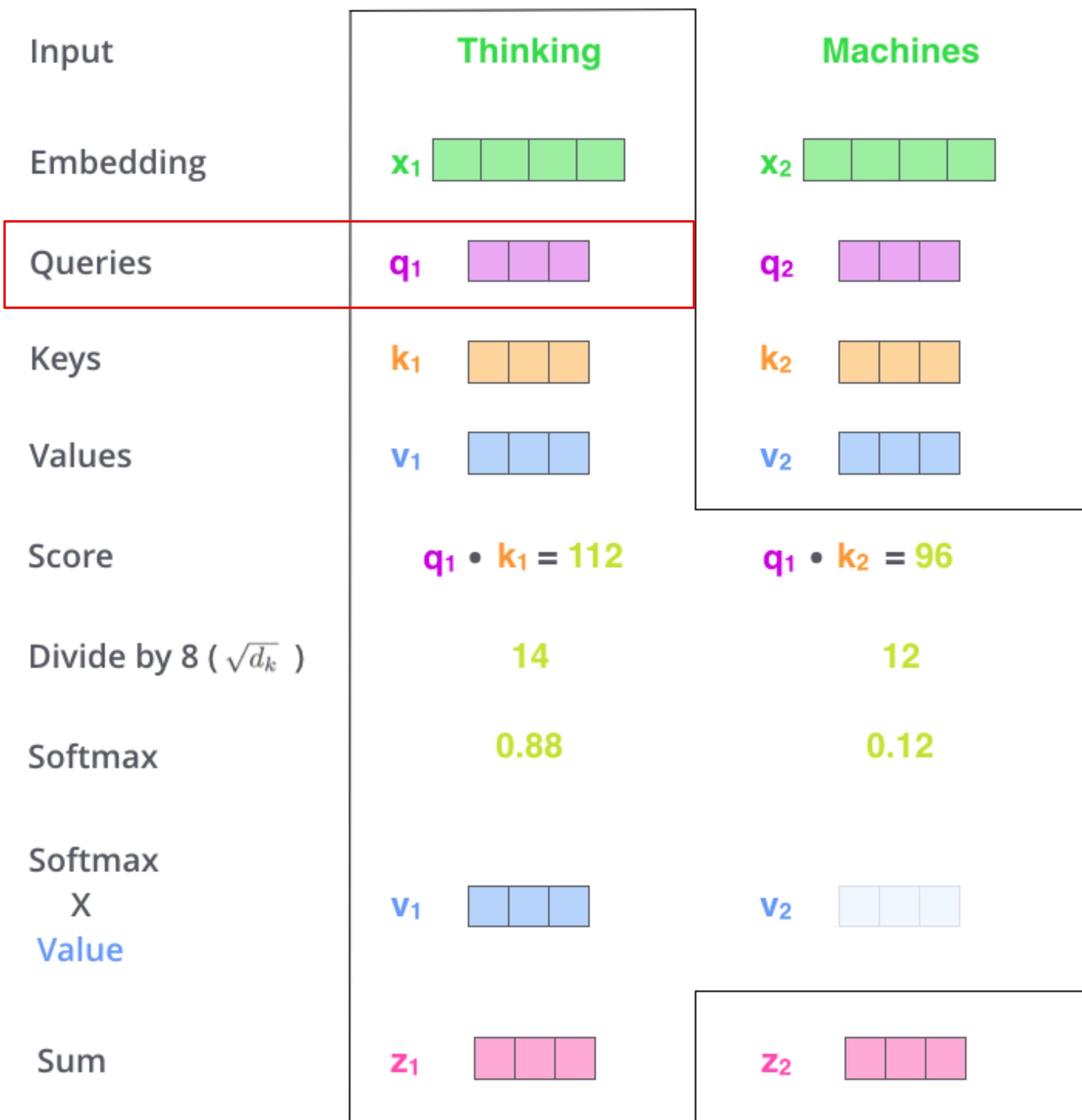
## Single Head Attention



Detailed information - what I want to read out from memory



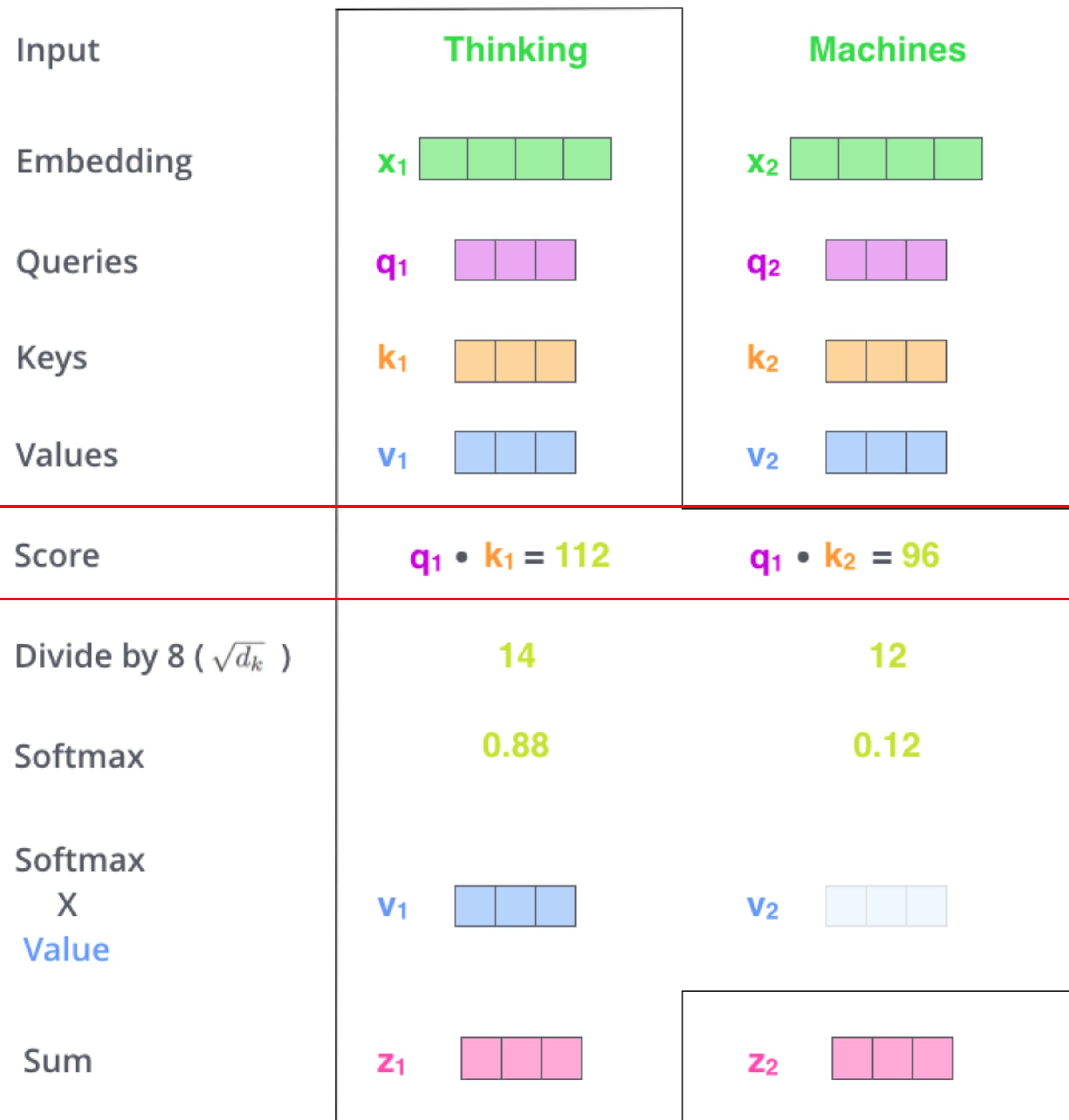
My current word embedding

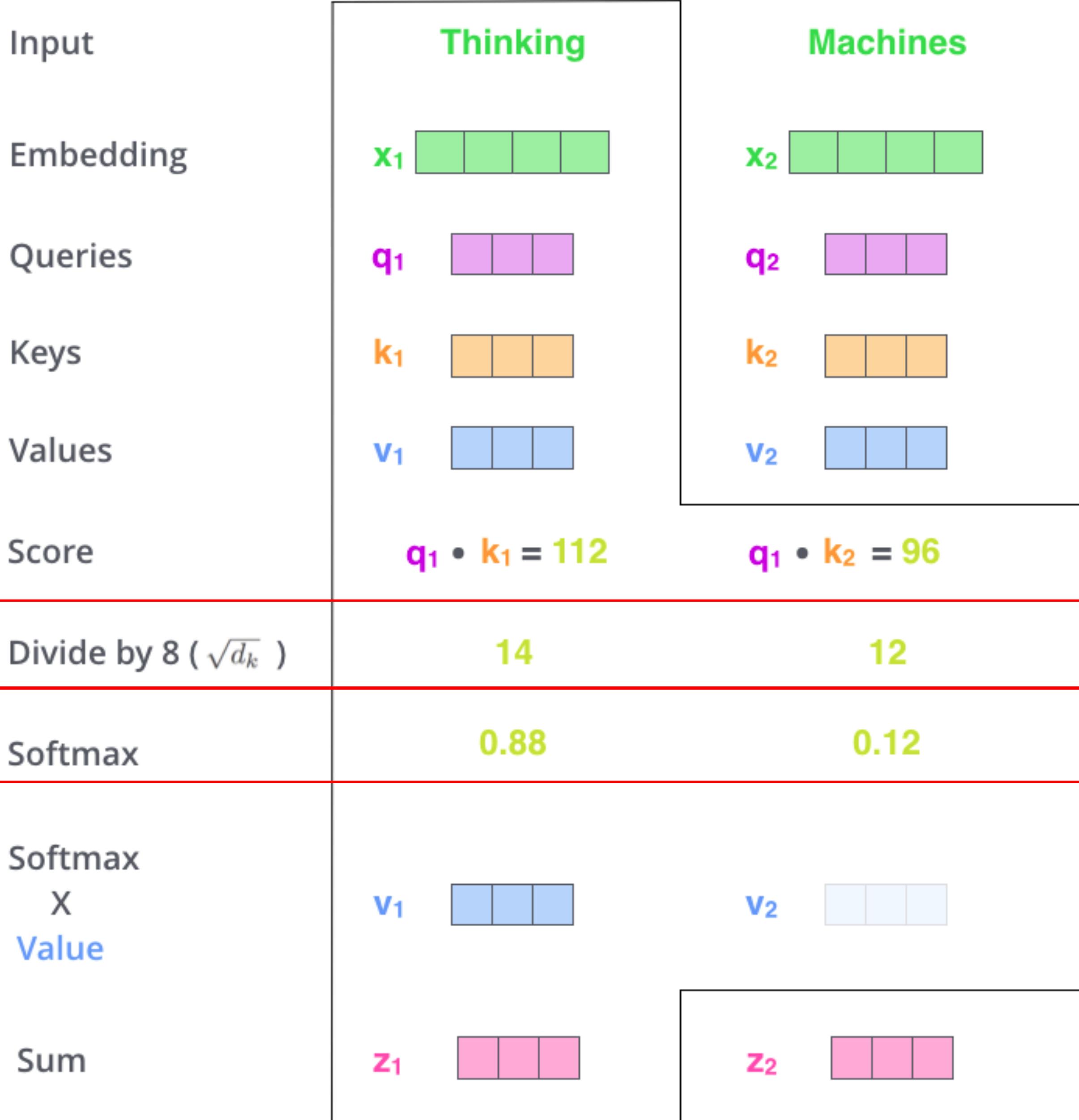


## Scalar, Pairwise

1] Relate

Compatibility bet. each embedding in the dictionary to myself



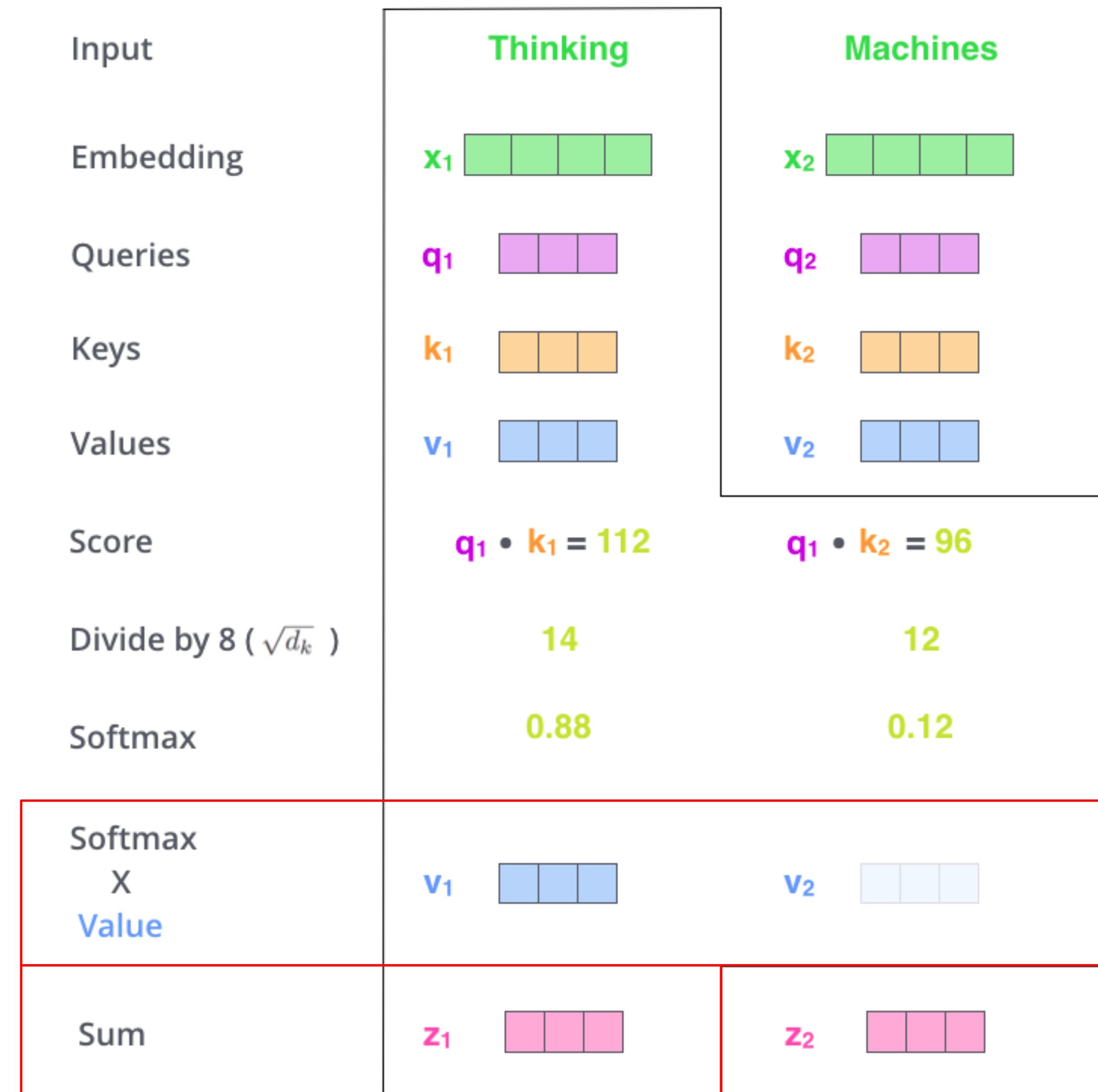


## 1] Relate

**Scaled:** because for large values of  $d_k$   
 → large values of dot product  
 → pushes the softmax to have small gradients.

## 2] Aggregate

Aggregate  
information from all  
tokens



# Single-Headed Attention

$$X \times W^Q = Q$$

The diagram illustrates the computation of the Query matrix  $Q$ . It shows the input matrix  $X$  (green) being multiplied by the weight matrix  $W^Q$  (magenta) to produce the output matrix  $Q$  (magenta). The input matrix  $X$  is a 2x4 matrix, and the weight matrix  $W^Q$  is a 4x4 matrix. The resulting matrix  $Q$  is also a 2x4 matrix.

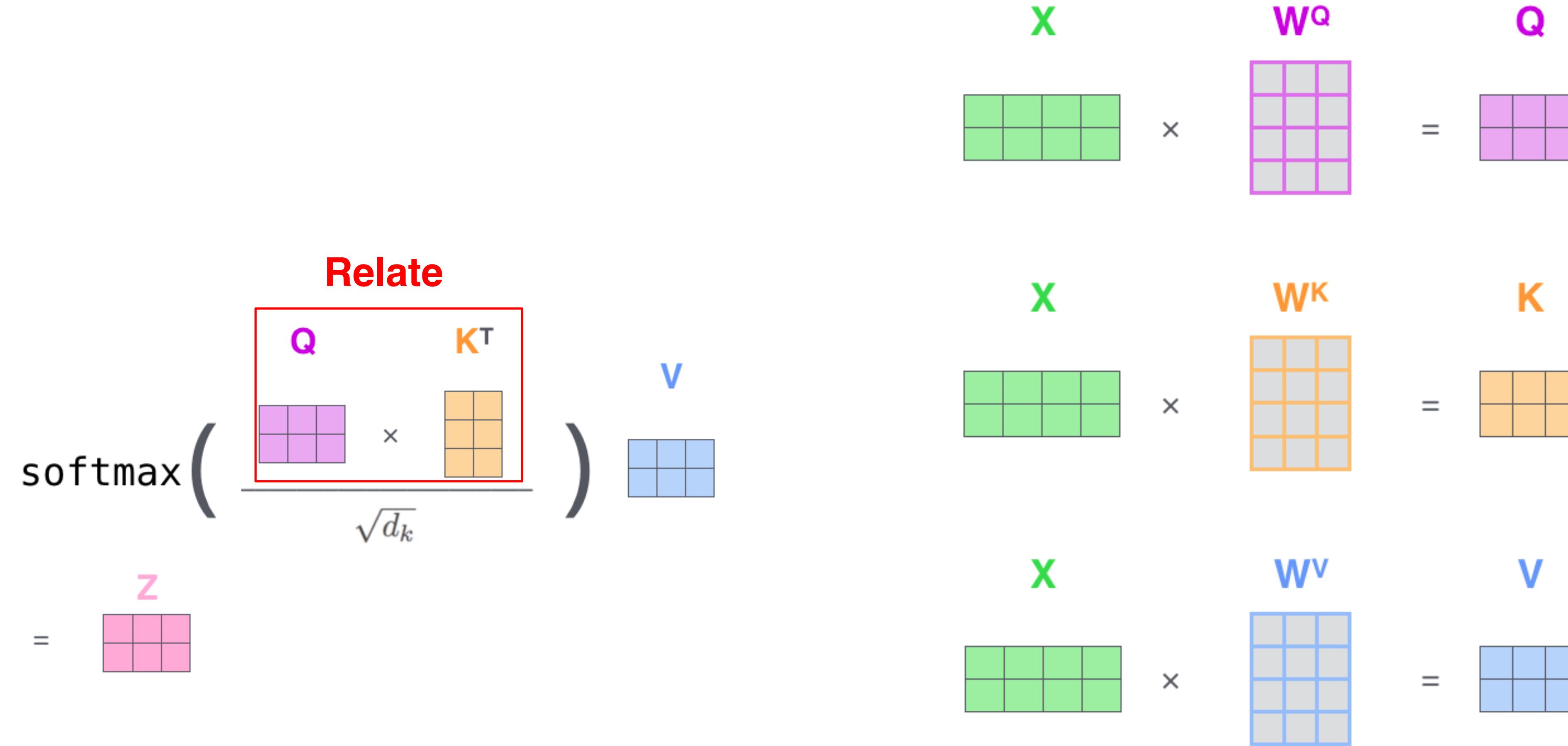
$$X \times W^K = K$$

The diagram illustrates the computation of the Key matrix  $K$ . It shows the input matrix  $X$  (green) being multiplied by the weight matrix  $W^K$  (orange) to produce the output matrix  $K$  (orange). The input matrix  $X$  is a 2x4 matrix, and the weight matrix  $W^K$  is a 4x4 matrix. The resulting matrix  $K$  is also a 2x4 matrix.

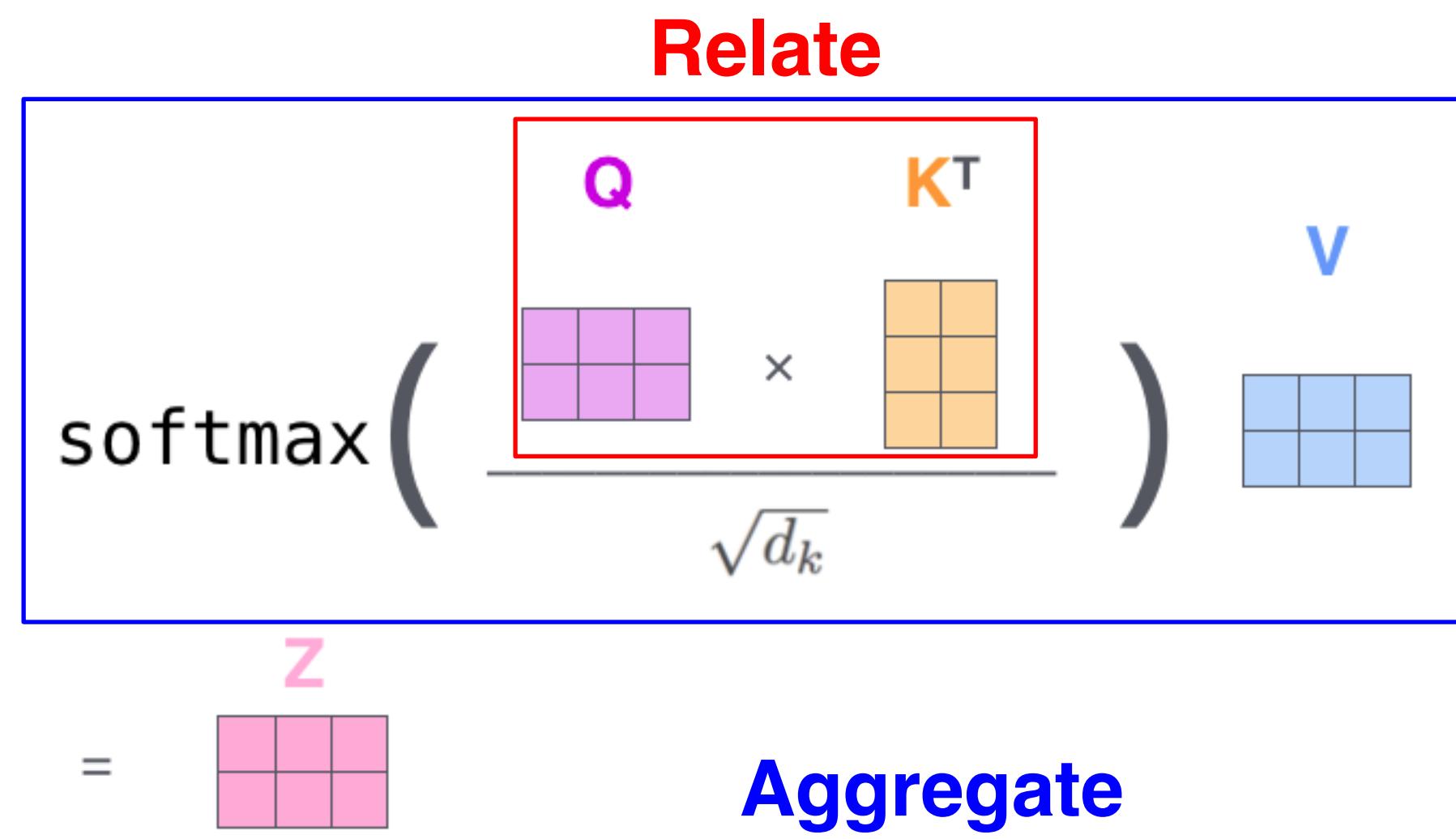
$$X \times W^V = V$$

The diagram illustrates the computation of the Value matrix  $V$ . It shows the input matrix  $X$  (green) being multiplied by the weight matrix  $W^V$  (blue) to produce the output matrix  $V$  (blue). The input matrix  $X$  is a 2x4 matrix, and the weight matrix  $W^V$  is a 4x4 matrix. The resulting matrix  $V$  is also a 2x4 matrix.

# Single-Headed Attention



# Single-Headed Attention



$$\begin{array}{ccc} X & W^Q & Q \\ \times & = & \\ \begin{matrix} \text{green} & \text{green} & \text{green} \\ \text{green} & \text{green} & \text{green} \end{matrix} & \begin{matrix} \text{purple} & \text{purple} & \text{purple} \\ \text{purple} & \text{purple} & \text{purple} \end{matrix} & \begin{matrix} \text{purple} & \text{purple} & \text{purple} \\ \text{purple} & \text{purple} & \text{purple} \end{matrix} \\ X & W^K & K \\ \times & = & \\ \begin{matrix} \text{green} & \text{green} & \text{green} \\ \text{green} & \text{green} & \text{green} \end{matrix} & \begin{matrix} \text{orange} & \text{orange} & \text{orange} \\ \text{orange} & \text{orange} & \text{orange} \end{matrix} & \begin{matrix} \text{orange} & \text{orange} & \text{orange} \\ \text{orange} & \text{orange} & \text{orange} \end{matrix} \\ X & W^V & V \\ \times & = & \\ \begin{matrix} \text{green} & \text{green} & \text{green} \\ \text{green} & \text{green} & \text{green} \end{matrix} & \begin{matrix} \text{blue} & \text{blue} & \text{blue} \\ \text{blue} & \text{blue} & \text{blue} \end{matrix} & \begin{matrix} \text{blue} & \text{blue} & \text{blue} \\ \text{blue} & \text{blue} & \text{blue} \end{matrix} \end{array}$$

# Multihead Attention

- 1) This is our input sentence\*
- 2) We embed each word\*



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



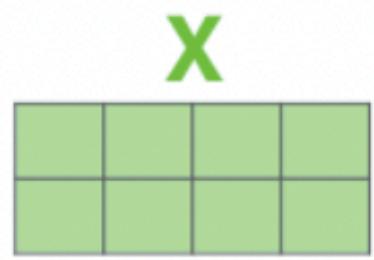
# Multihead Attention

1) This is our  
input sentence\*

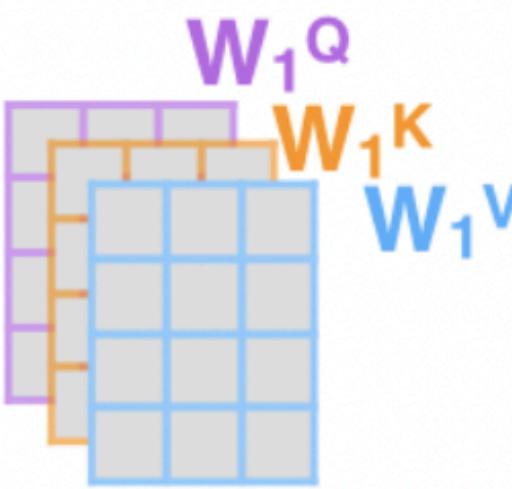
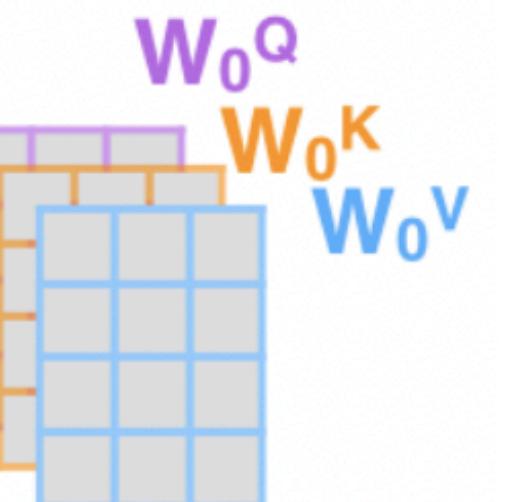
2) We embed  
each word\*

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

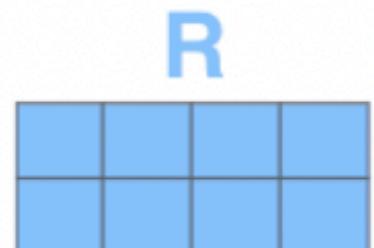
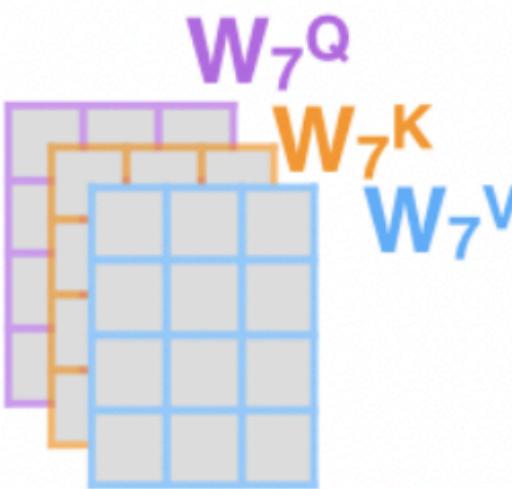
Thinking  
Machines



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



...



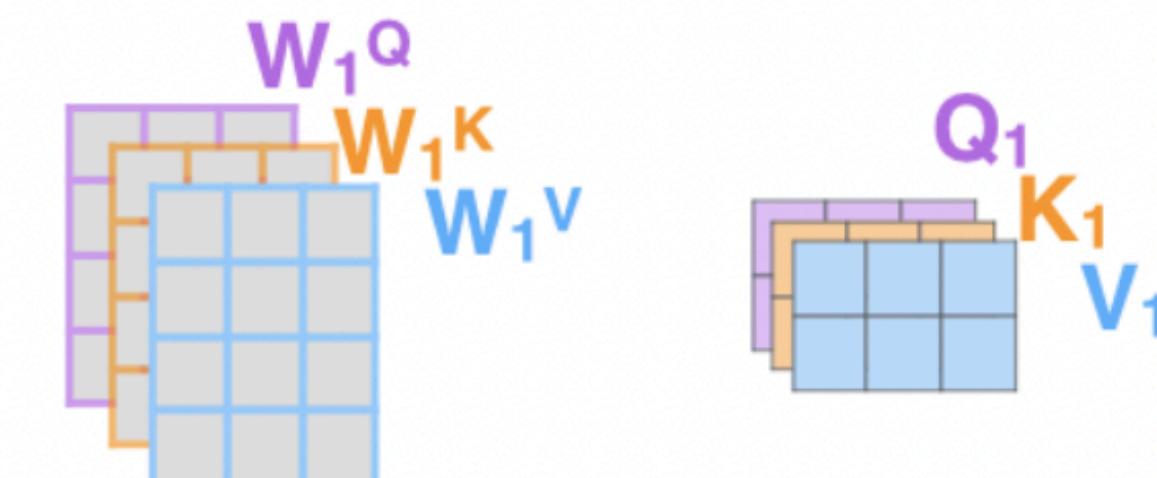
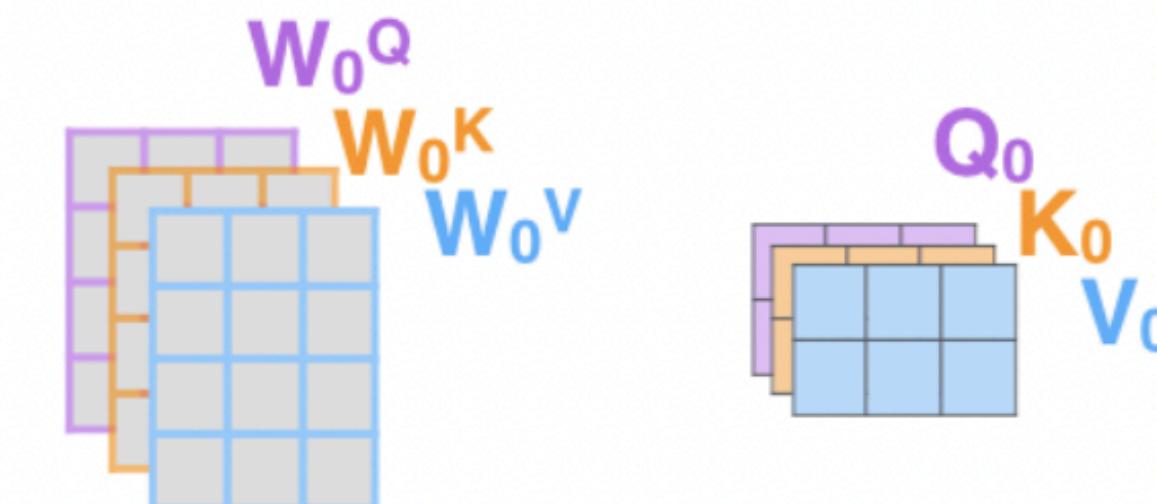
# Multihead Attention

1) This is our  
input sentence\*

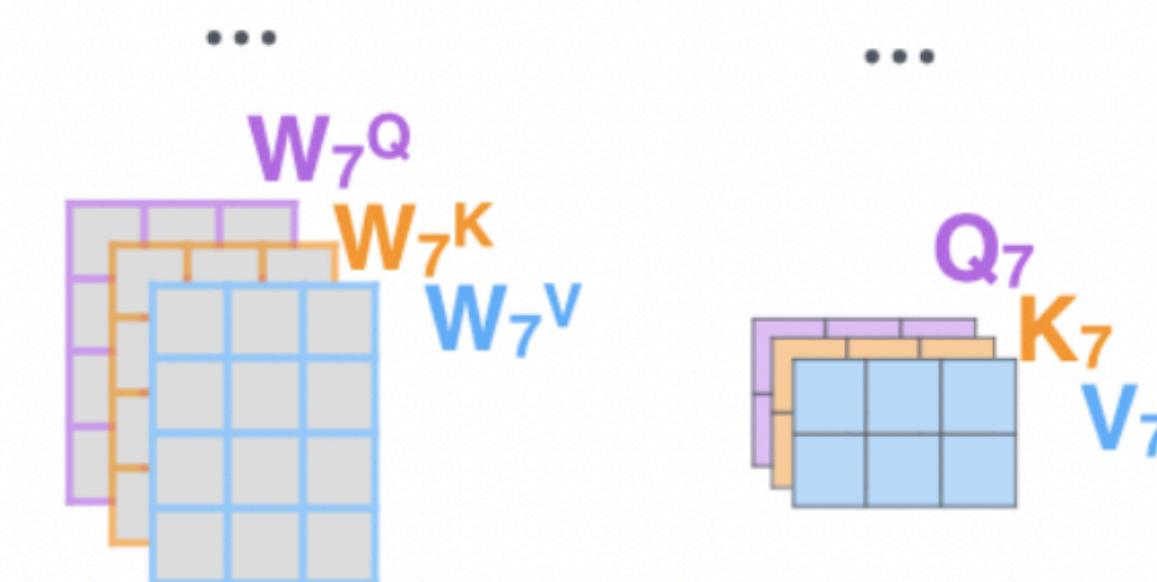
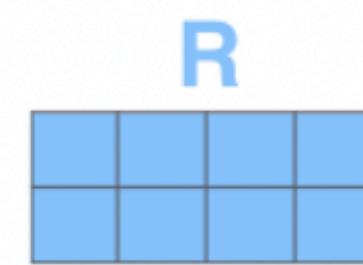
2) We embed  
each word\*

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

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

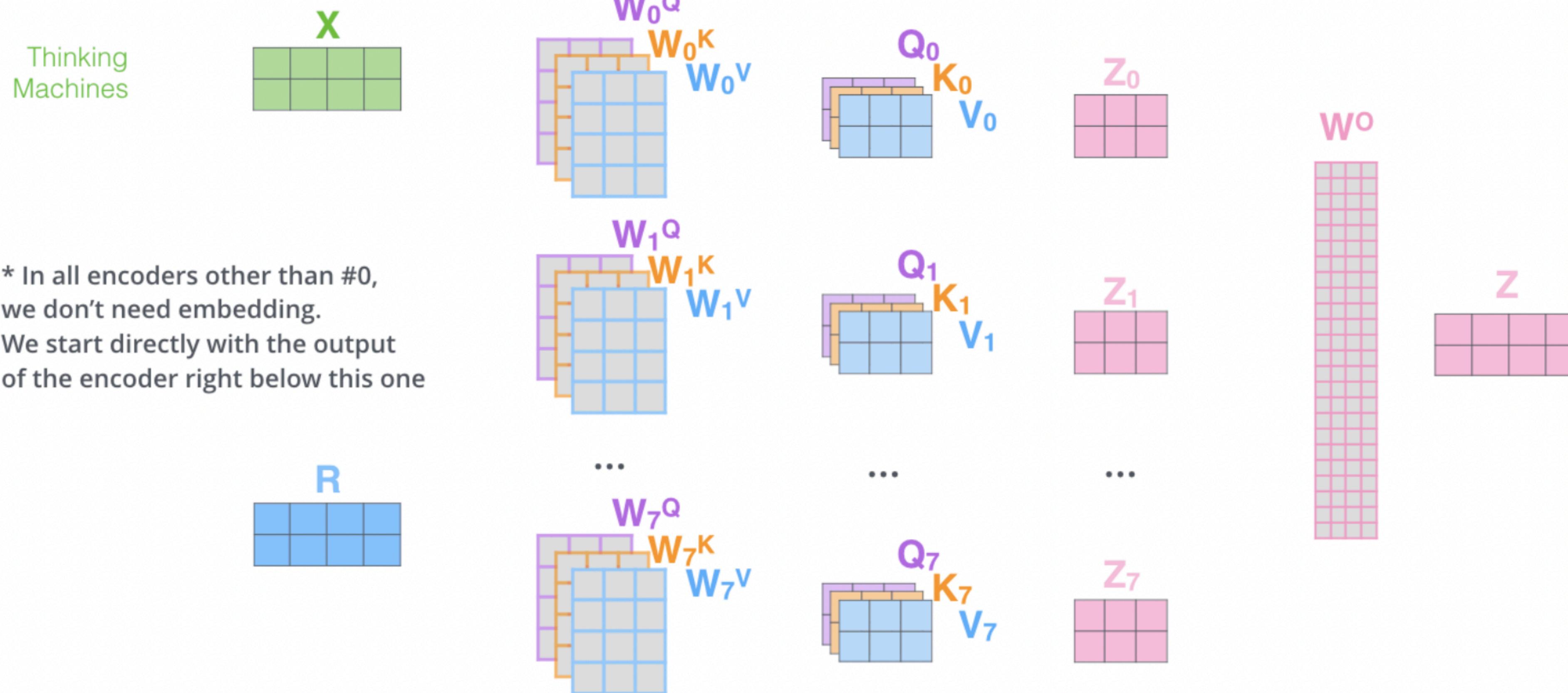


\* In all encoders other than #0,  
we don't need embedding.  
We start directly with the output  
of the encoder right below this one



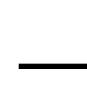
# Multihead Attention

- 1) This is our input sentence\* each word\*
- 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



# Transformers in Computer Vision

April 2020



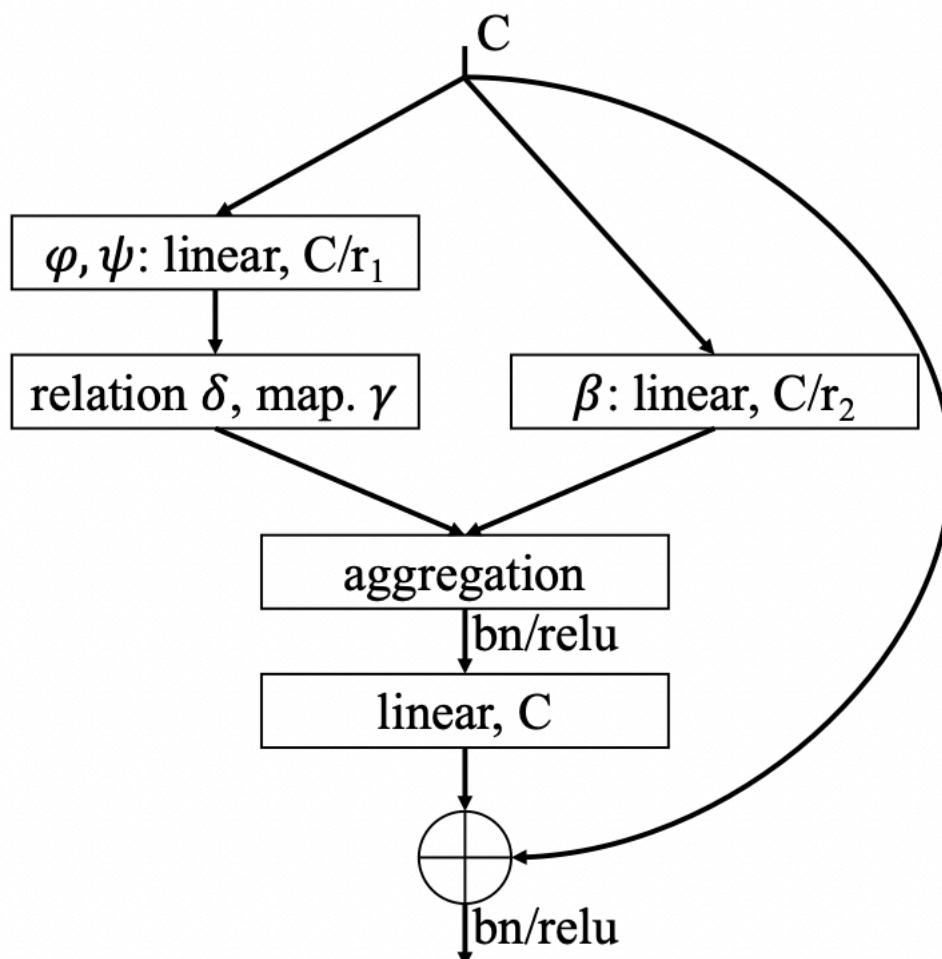
May 2020



October 2020



**Exploring Self Attention for  
Image Recognition**



**DetR**

**Vision Transformers**

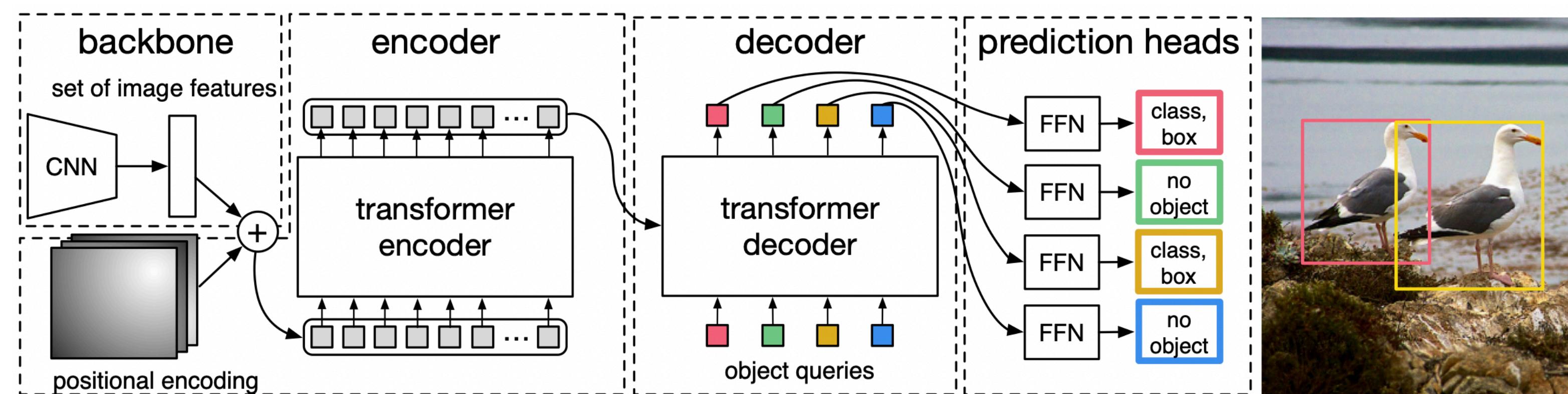
# Transformers in Computer Vision

April 2020

May 2020

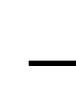
June 2021

**Exploring Self Attention for  
Image Recognition**



# Transformers in Computer Vision

April 2020



May 2020



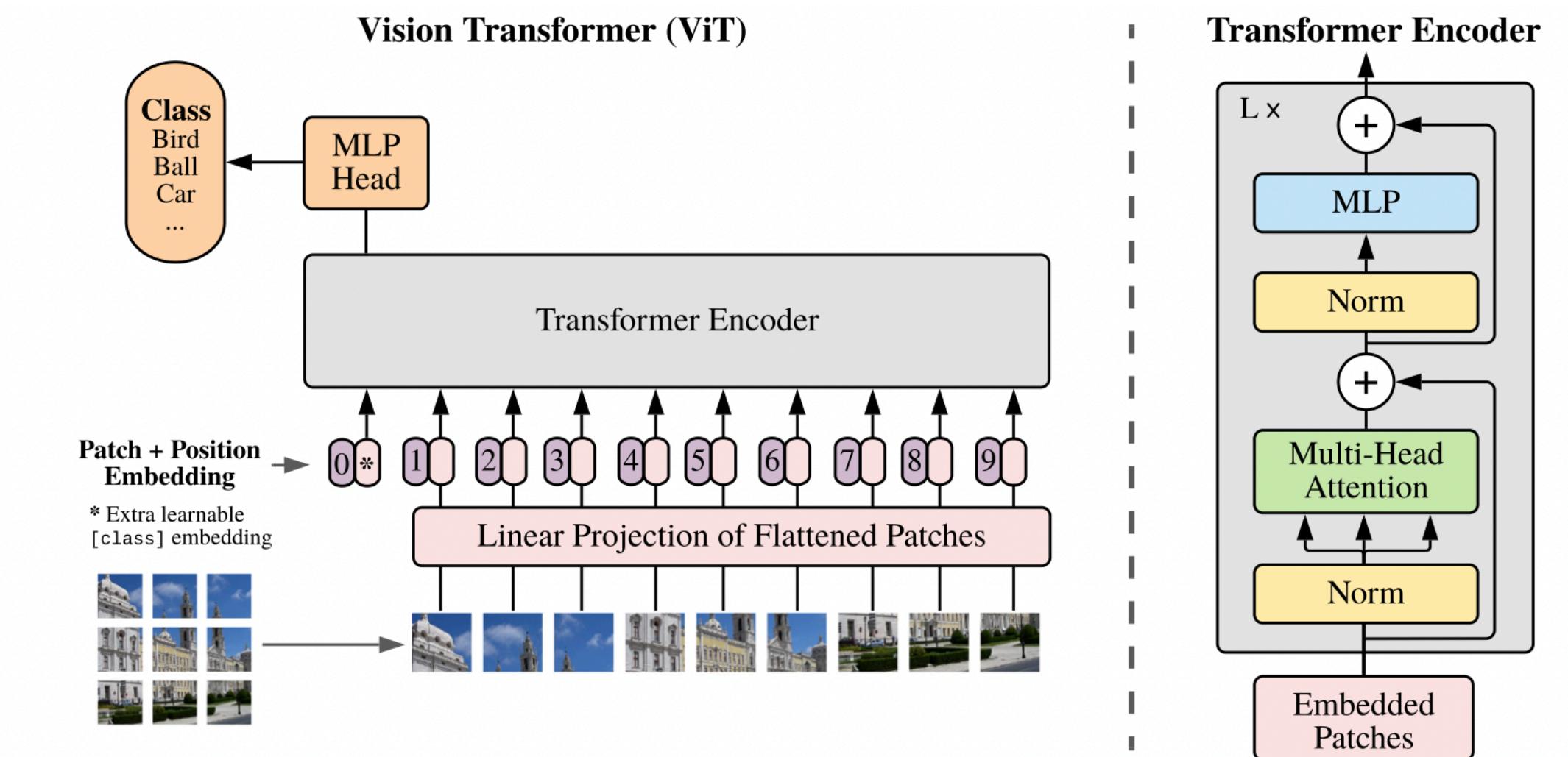
June 2021



**Exploring Self Attention for  
Image Recognition**

**DetR**

**Vision Transformers**

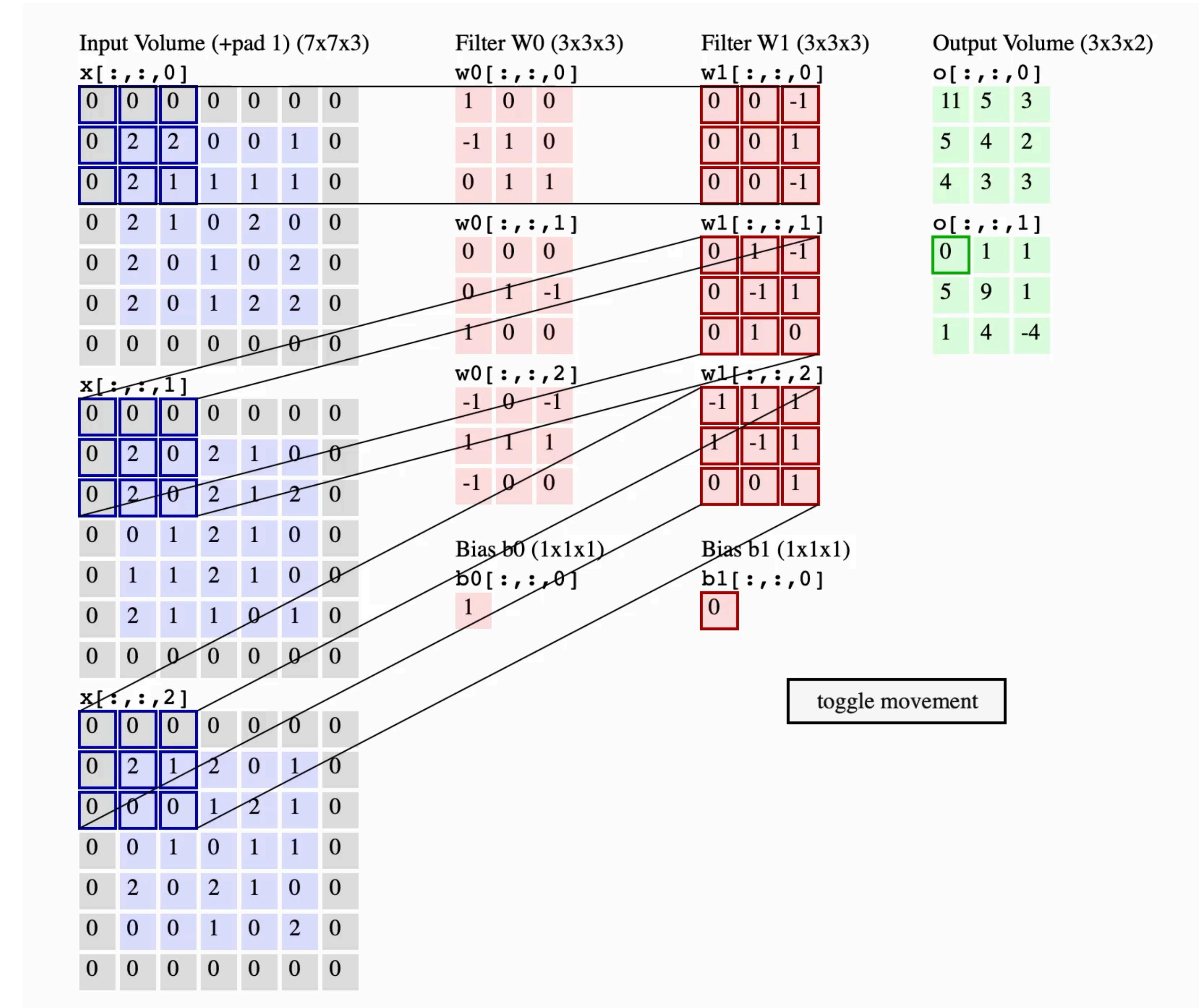


# Self Attention: A New Perspective

**Convolutional Networks**

**Aggregate Function**

**Fixed Weights**



# Self Attention: A New Perspective

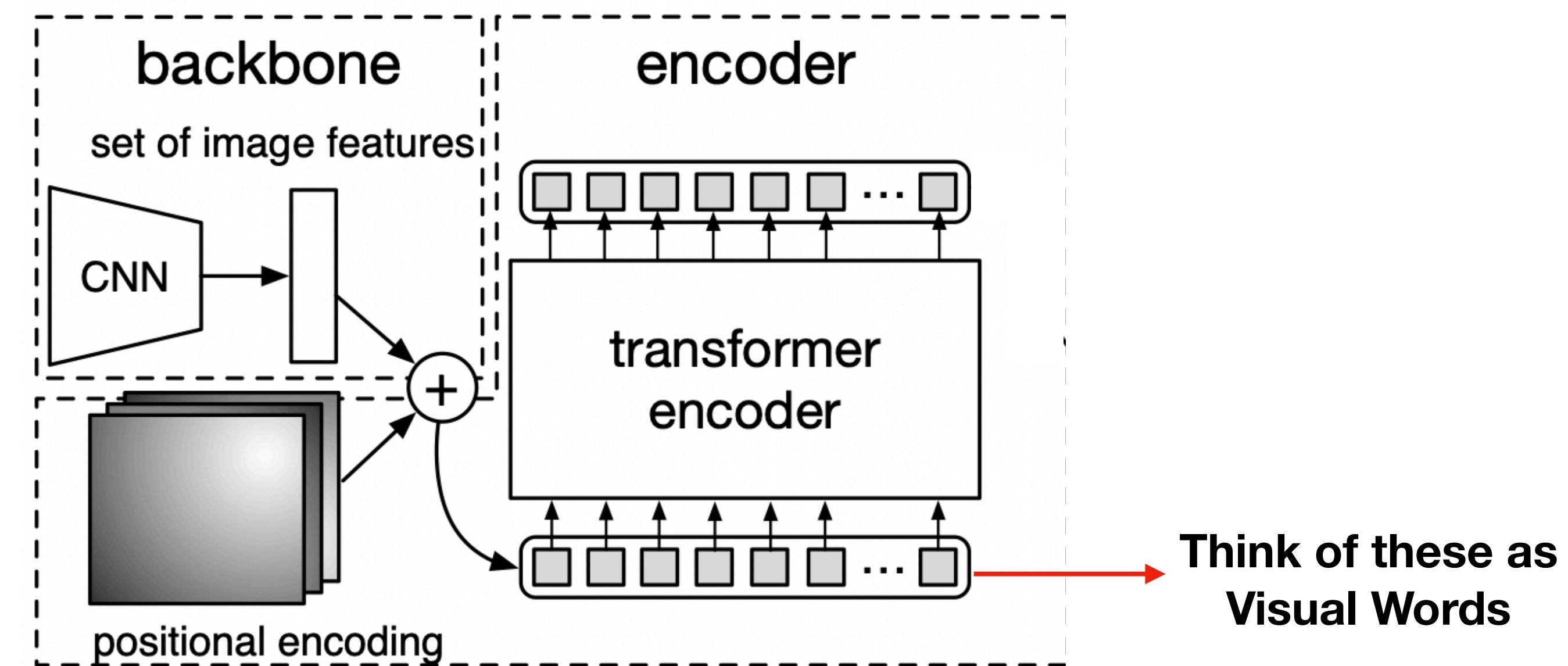
## Transformers

$$\text{softmax} \left( \frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}$$

=  $\mathbf{z}$

## Aggregate Function

Content Adaptive Weights



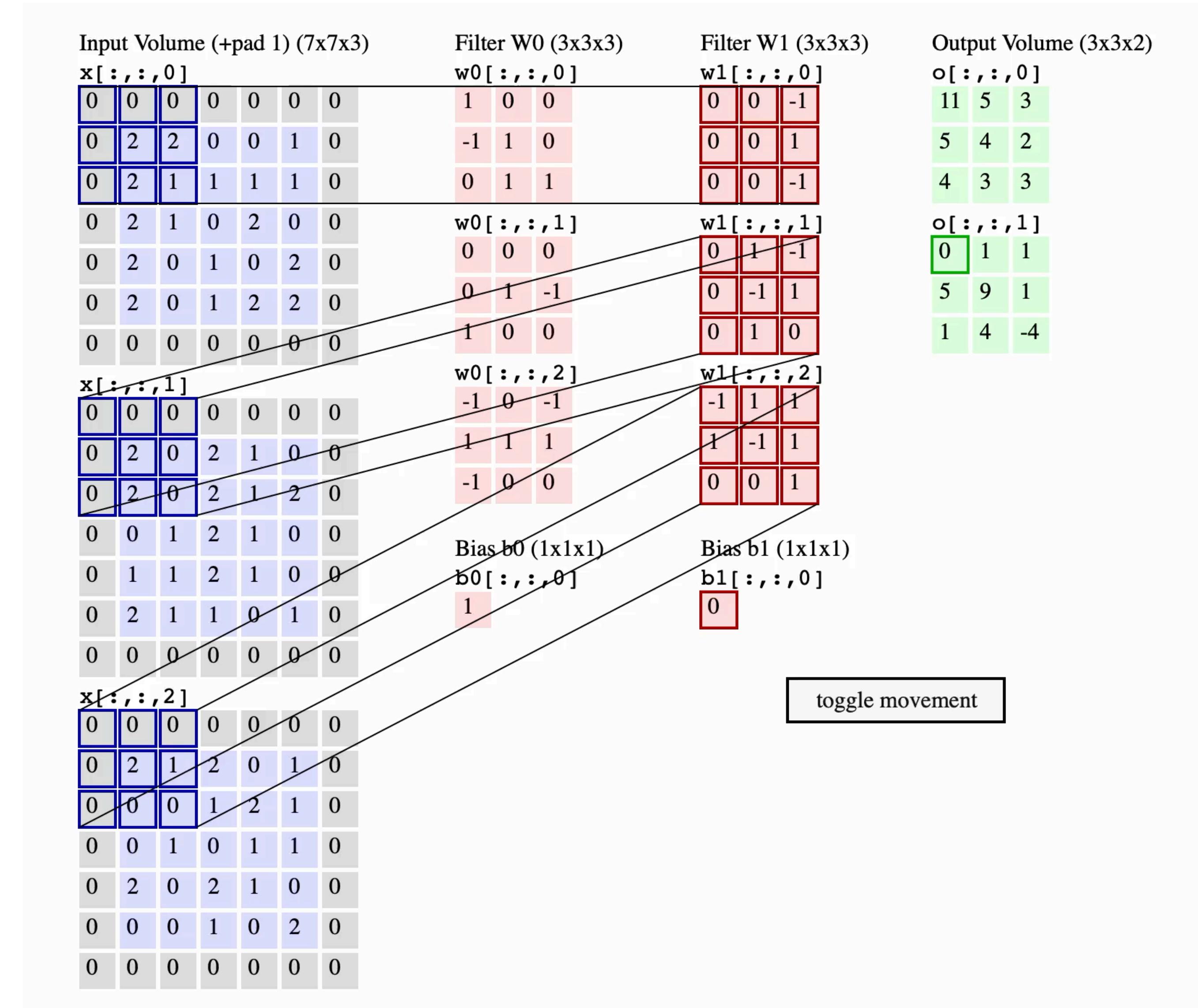
Tokens: Flattened CNN features

# Self Attention: A New Perspective

Convolutional Networks

Aggregate Function

Local

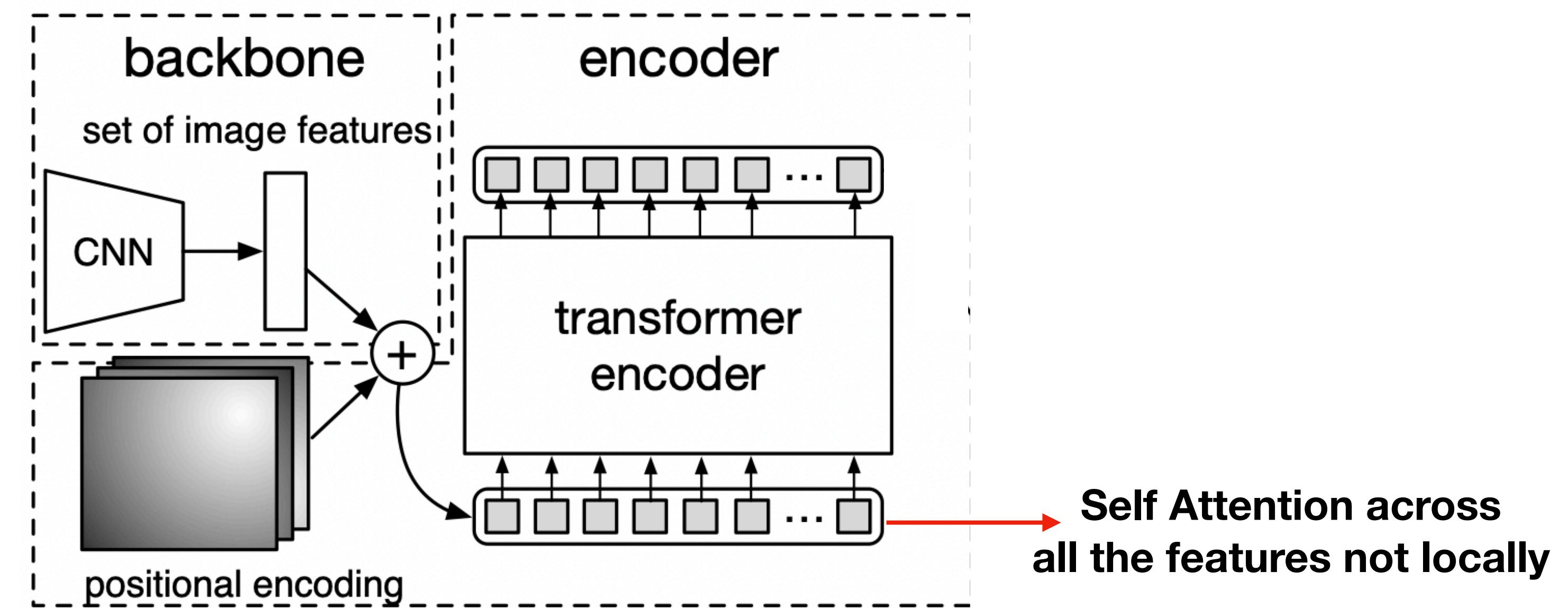


# Self Attention: A New Perspective

**Transformers**

**Aggregate Function**

**Global Context**

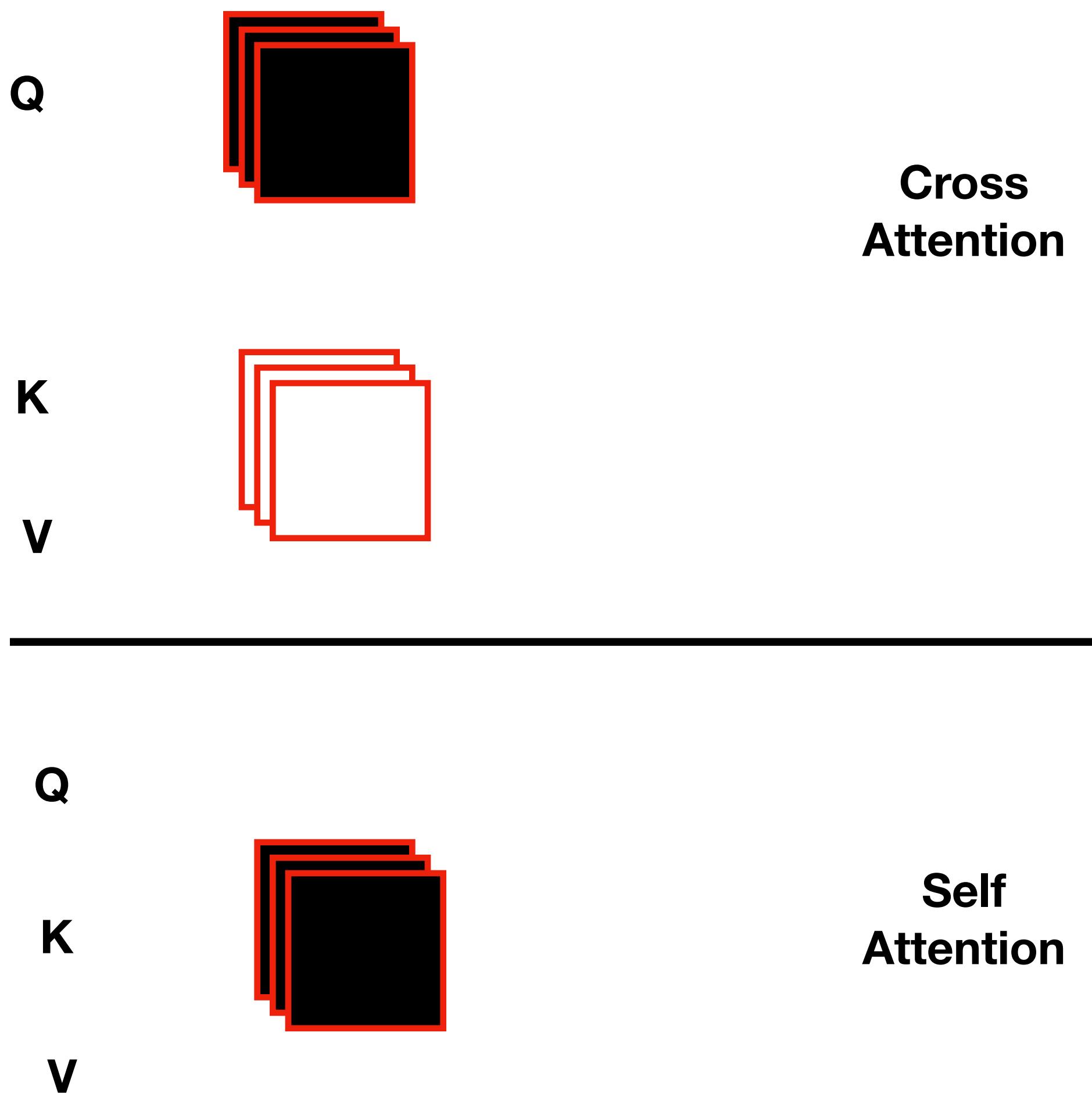


**Tokens: Flattened CNN features**

# Self Attention vs Cross Attention

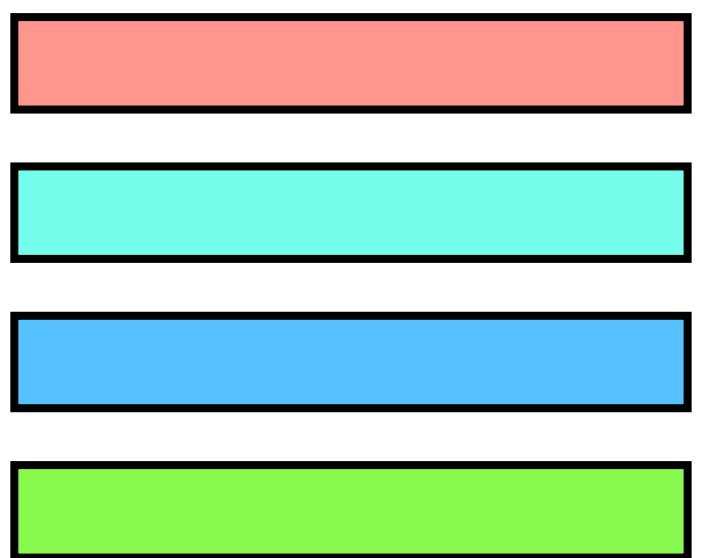
$$\text{softmax} \left( \frac{\begin{matrix} Q \\ \times \\ K^T \end{matrix}}{\sqrt{d_k}} \right) V$$

$$= \begin{matrix} Z \\ \begin{matrix} \text{MultiHeaded Attention in both} \end{matrix} \end{matrix}$$



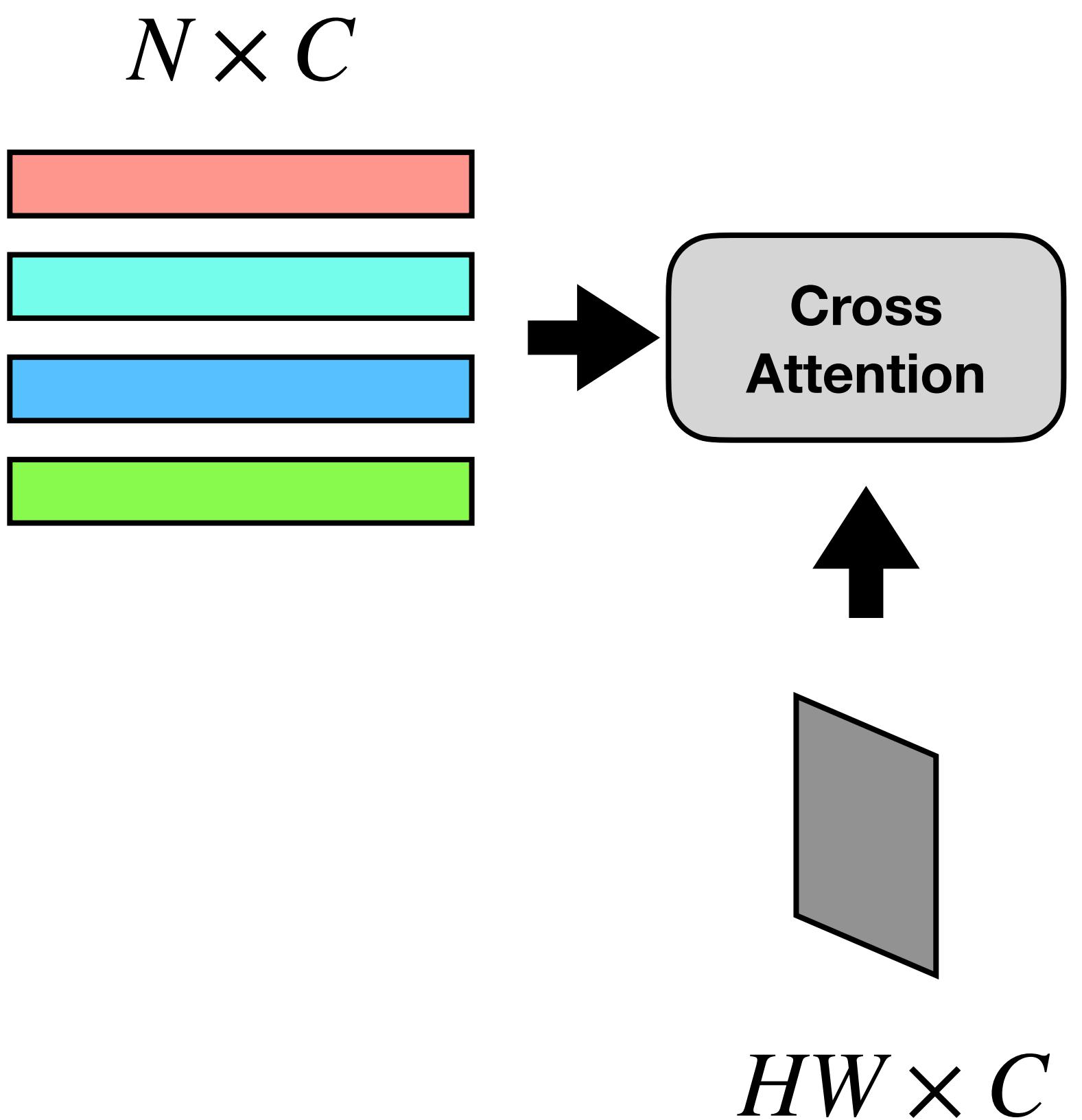
# Cross Attention

$N \times C$

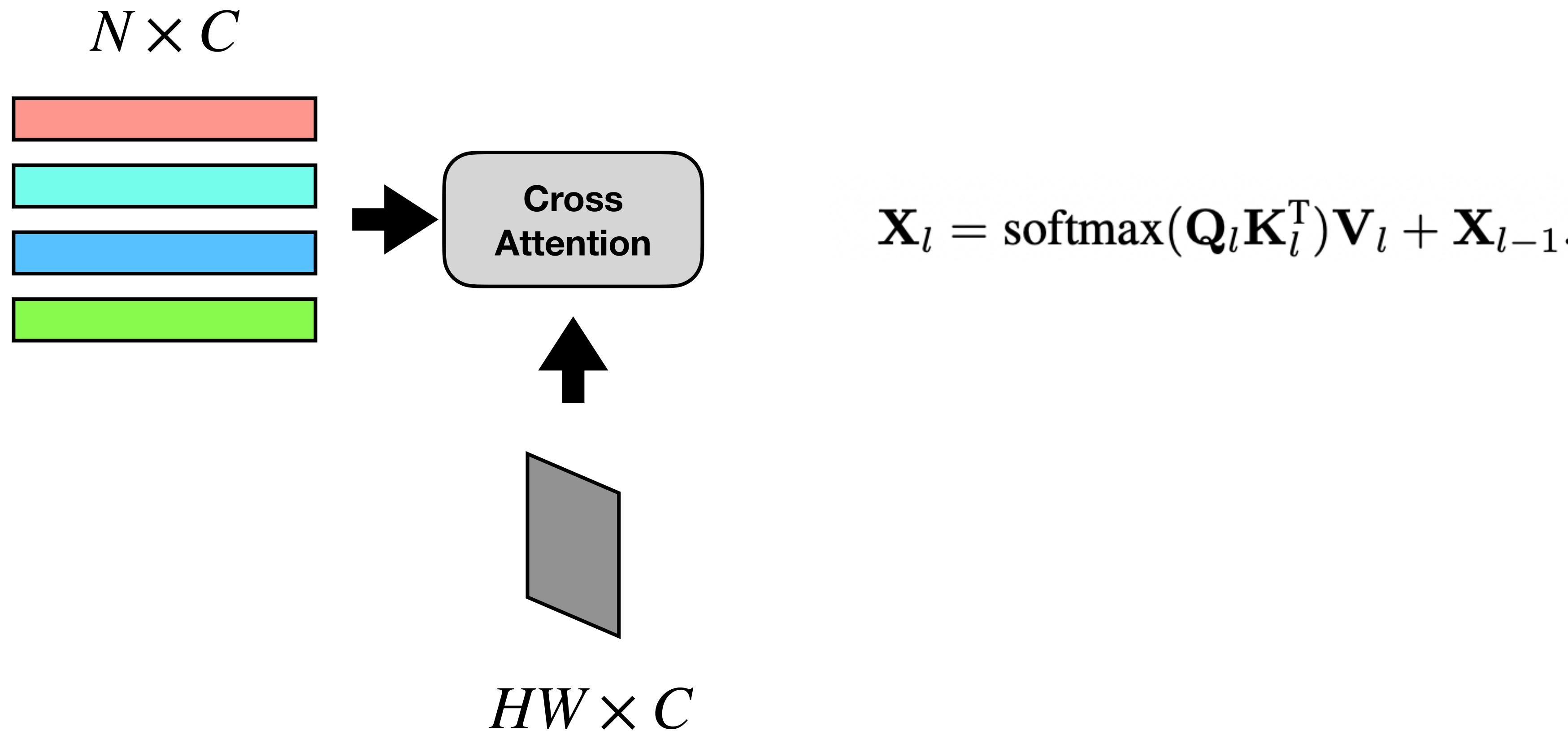


**Learnable Queries**

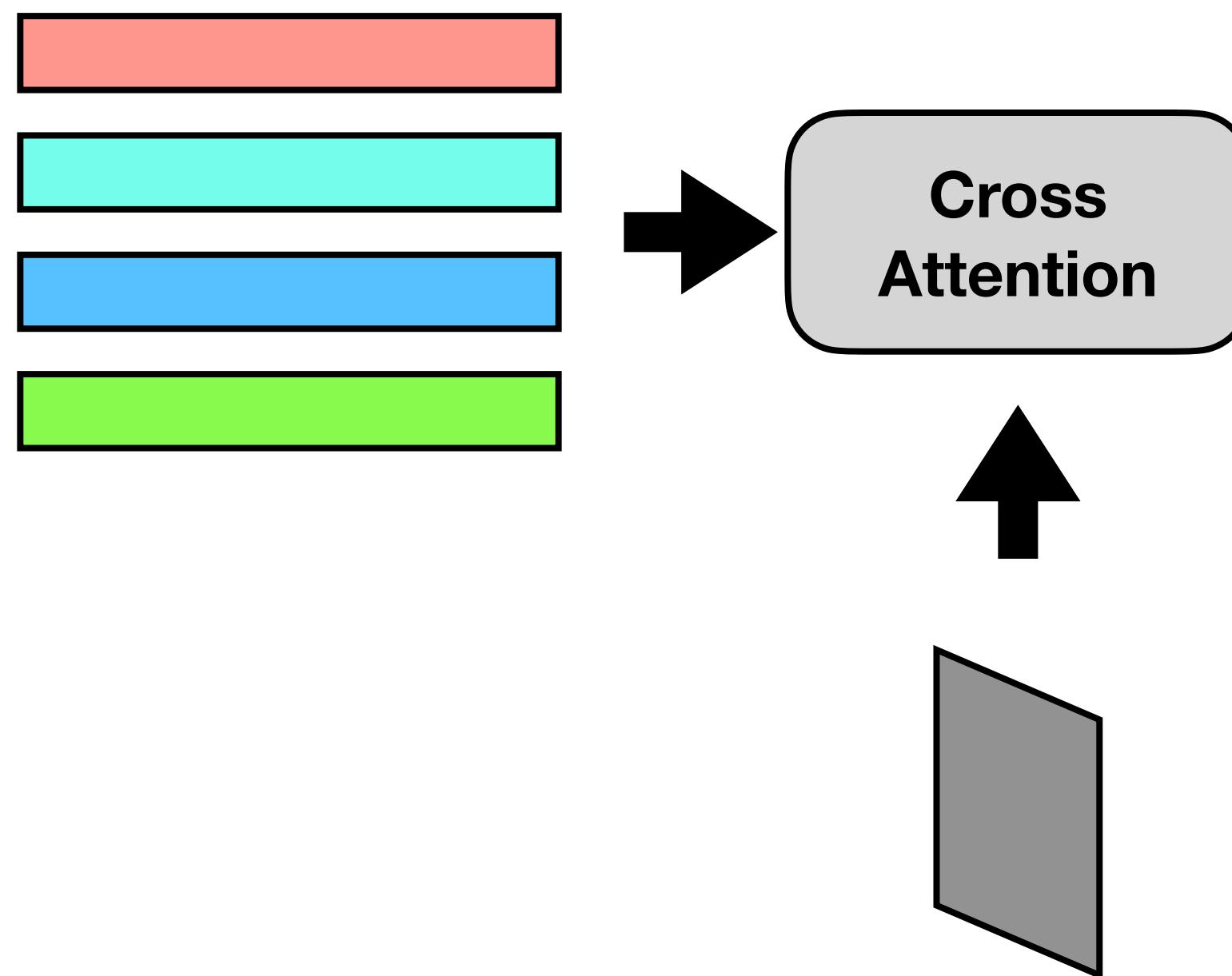
# Cross Attention



# Cross Attention



# Cross Attention



$$\mathbf{X}_l = \text{softmax}(\mathbf{Q}_l \mathbf{K}_l^T) \mathbf{V}_l + \mathbf{X}_{l-1}.$$

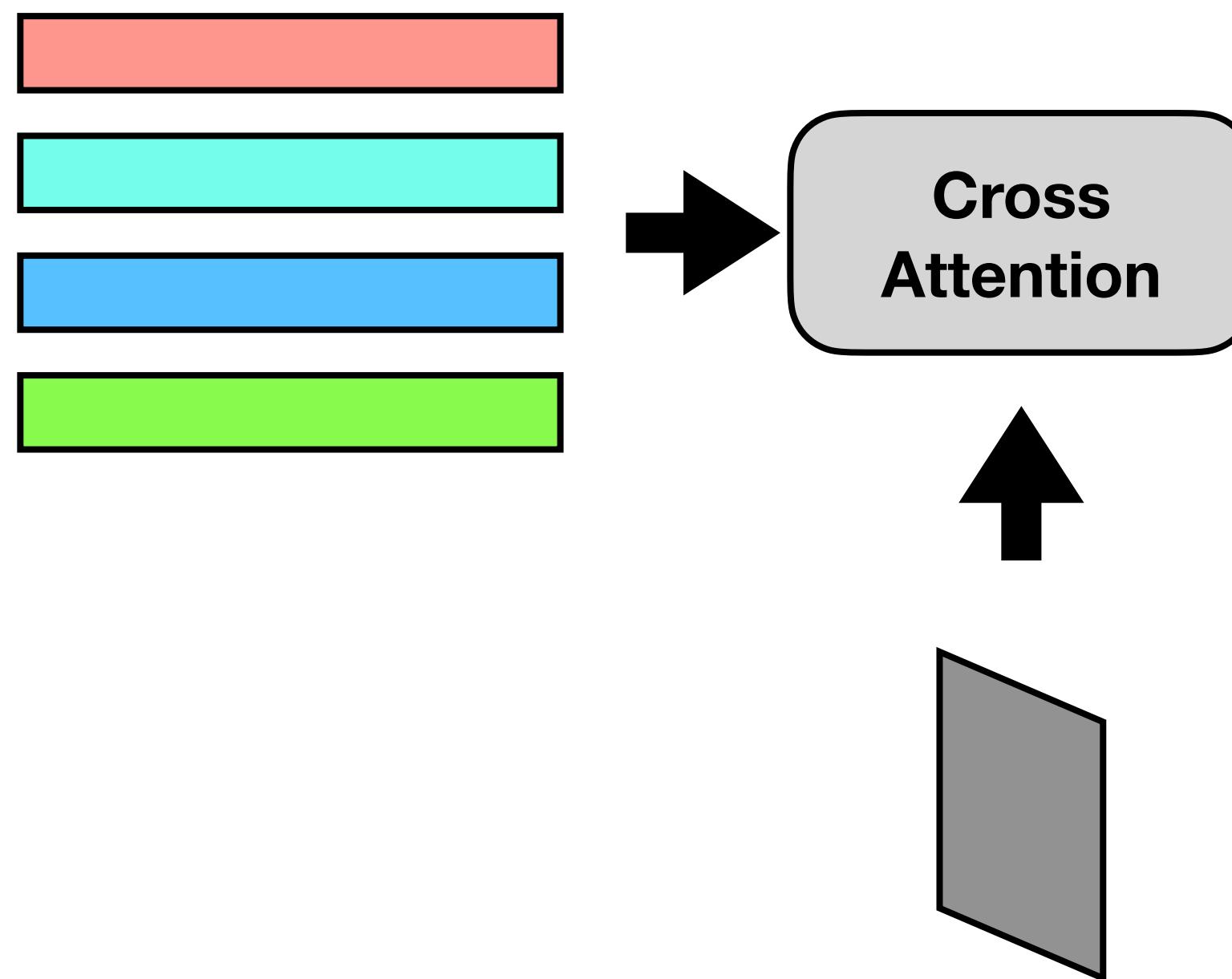
$N \times C \quad C \times HW$

$N \times HW$

**Attention Maps**

**Relate**

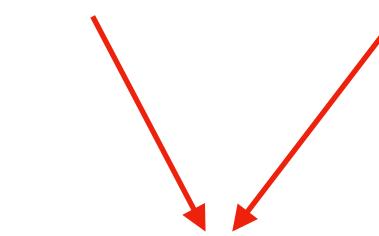
# Cross Attention



$$\mathbf{X}_l = \text{softmax}(\mathbf{Q}_l \mathbf{K}_l^T) \mathbf{V}_l + \mathbf{X}_{l-1}.$$

$N \times HW \quad HW \times C$

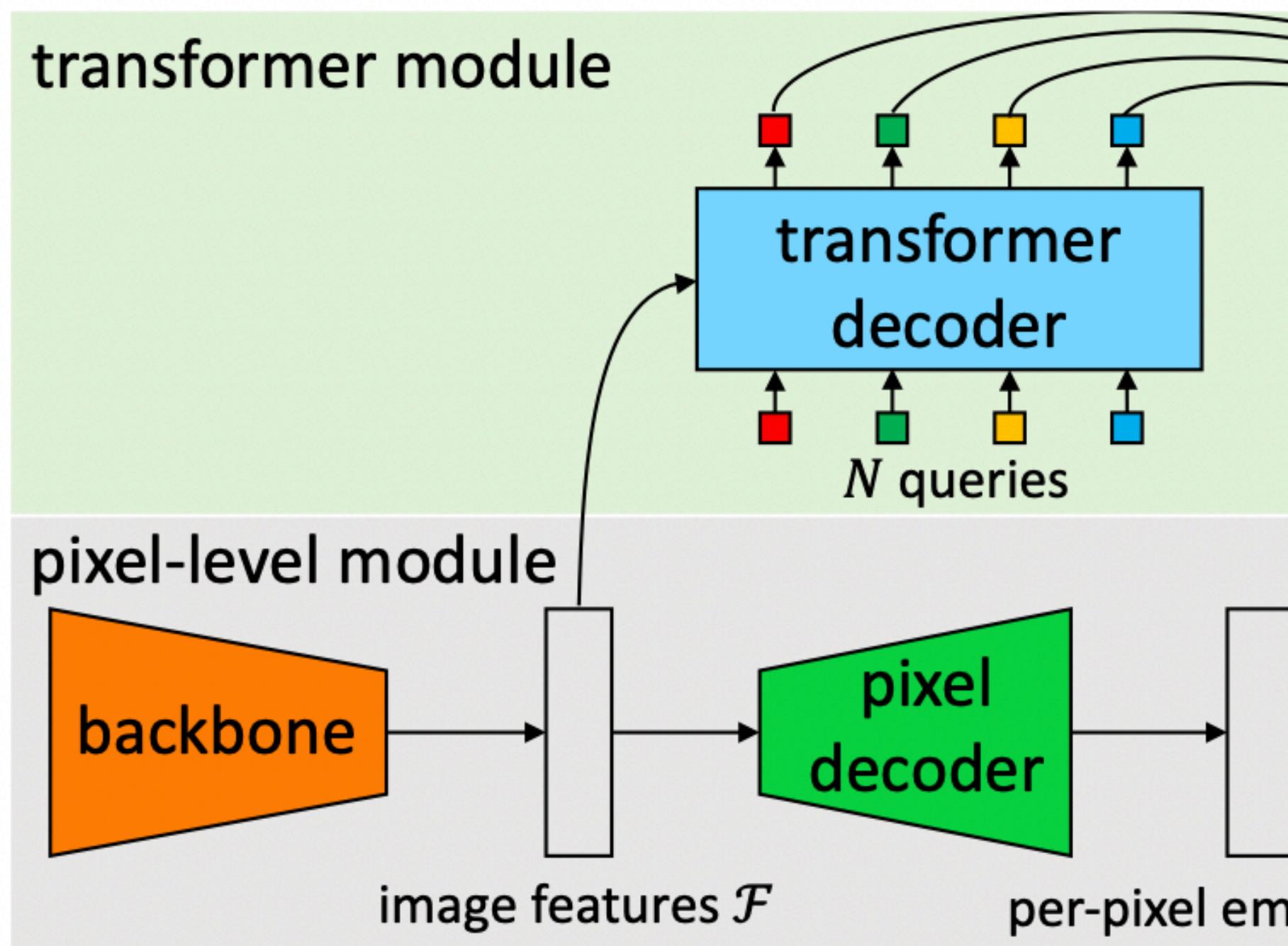
**Attention Maps**



$N \times C$

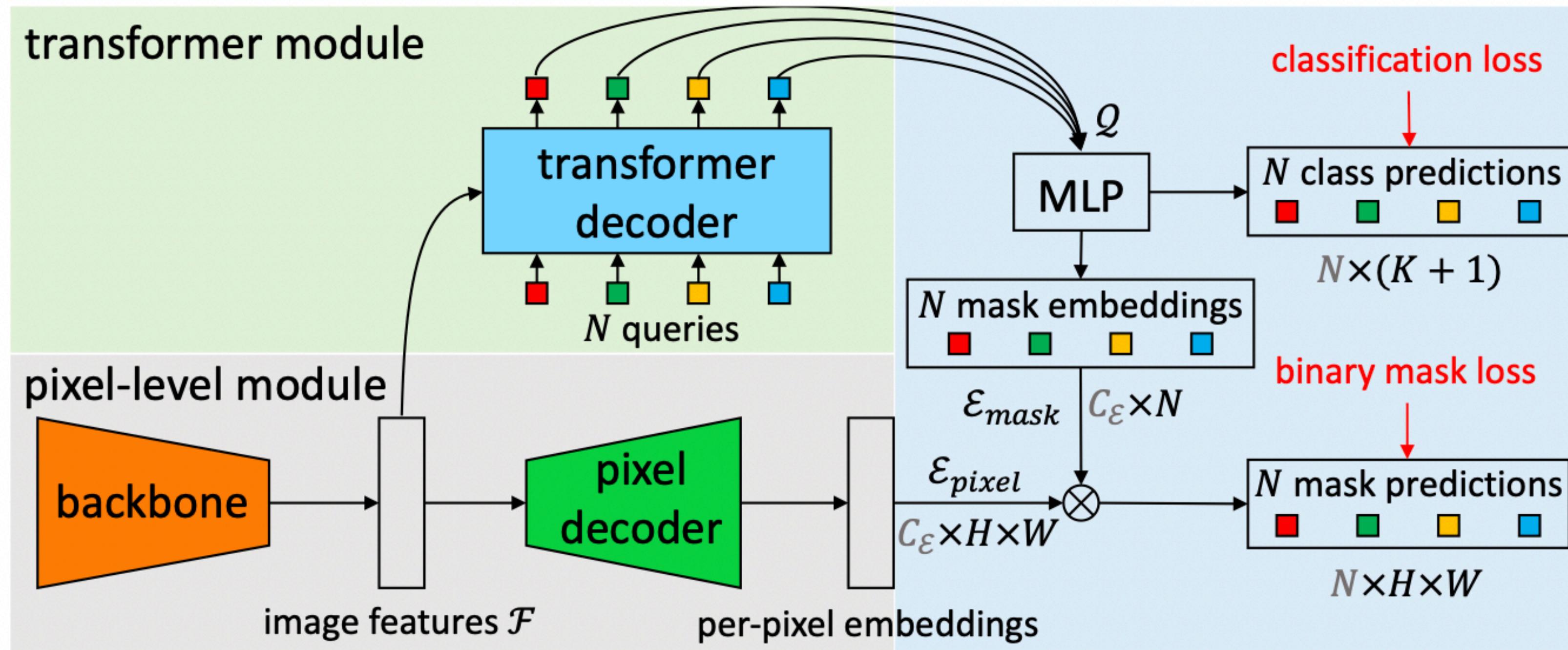
**Aggregate**

# MaskFormer



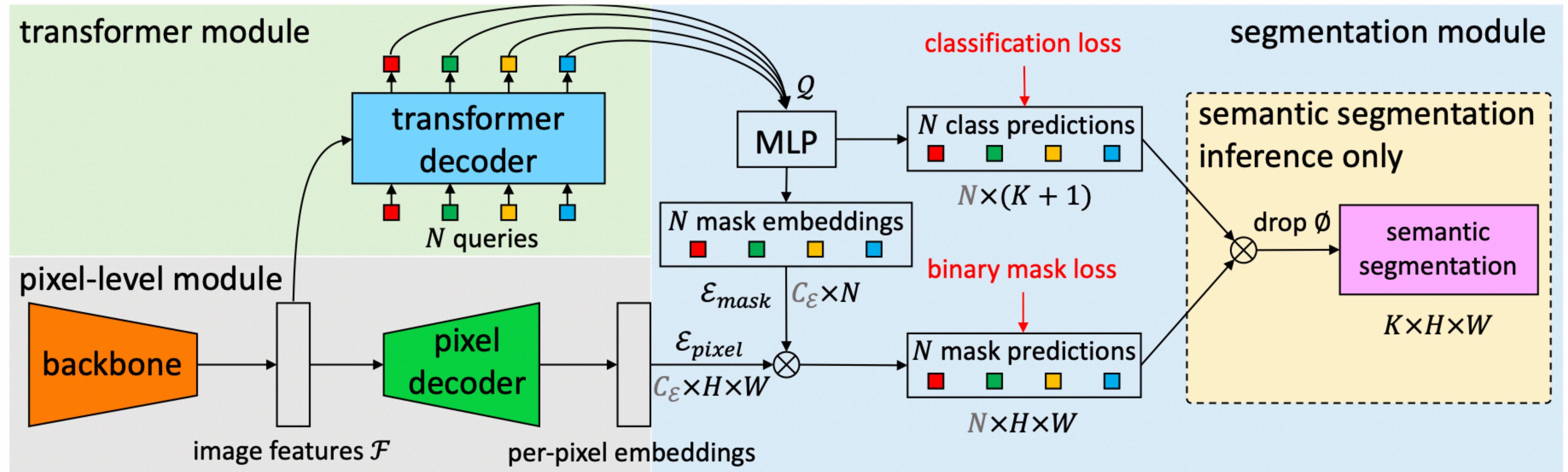
Cheng, Bowen, Alex Schwing, and Alexander Kirillov. "Per-pixel classification is not all you need for semantic segmentation." *Advances in Neural Information Processing Systems* 34 (2021): 17864-17875.

# MaskFormer



Cheng, Bowen, Alex Schwing, and Alexander Kirillov. "Per-pixel classification is not all you need for semantic segmentation." *Advances in Neural Information Processing Systems* 34 (2021): 17864-17875.

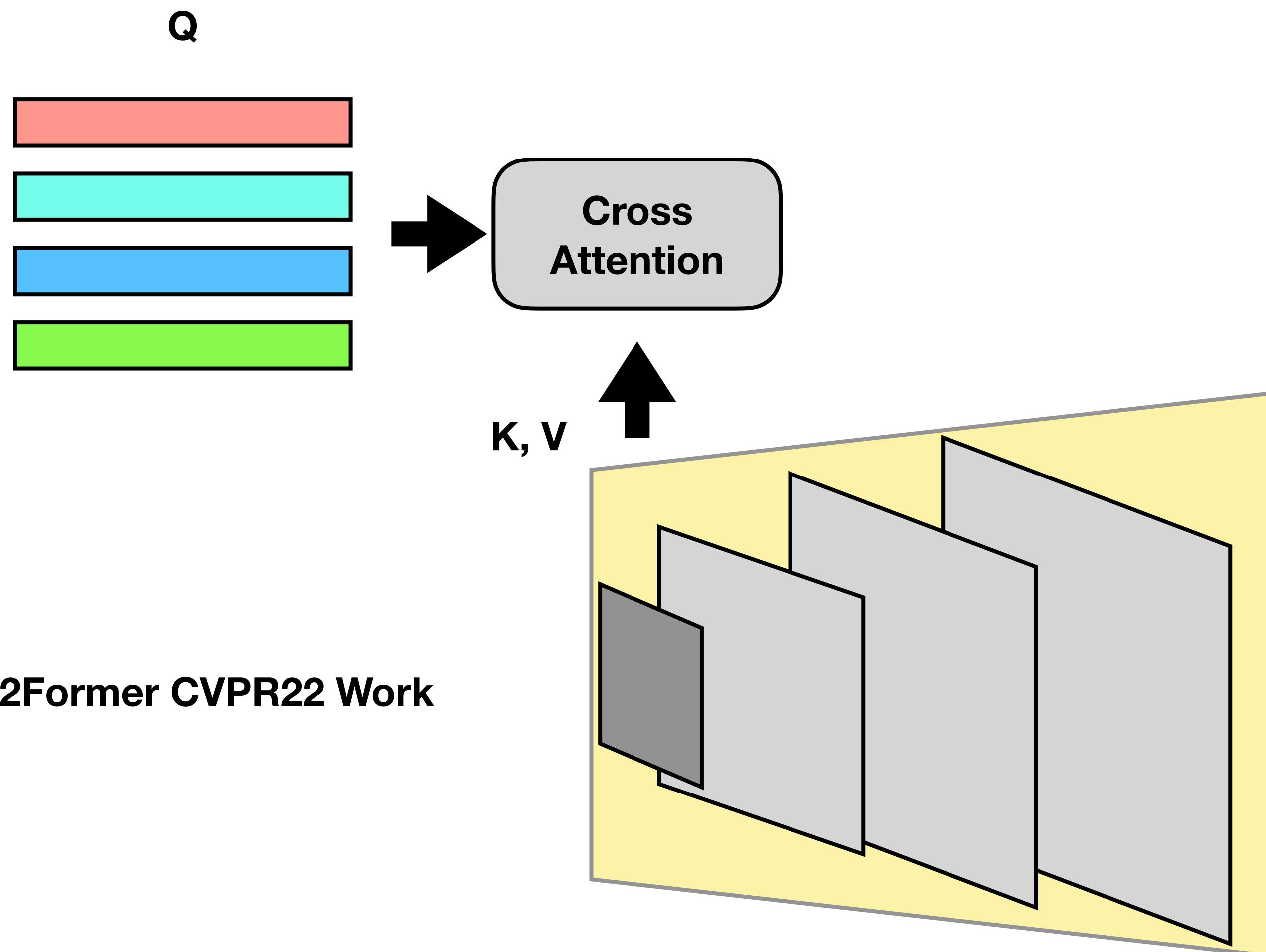
# MaskFormer



Cheng, Bowen, Alex Schwing, and Alexander Kirillov. "Per-pixel classification is not all you need for semantic segmentation." *Advances in Neural Information Processing Systems* 34 (2021): 17864-17875.

# Mask2Former

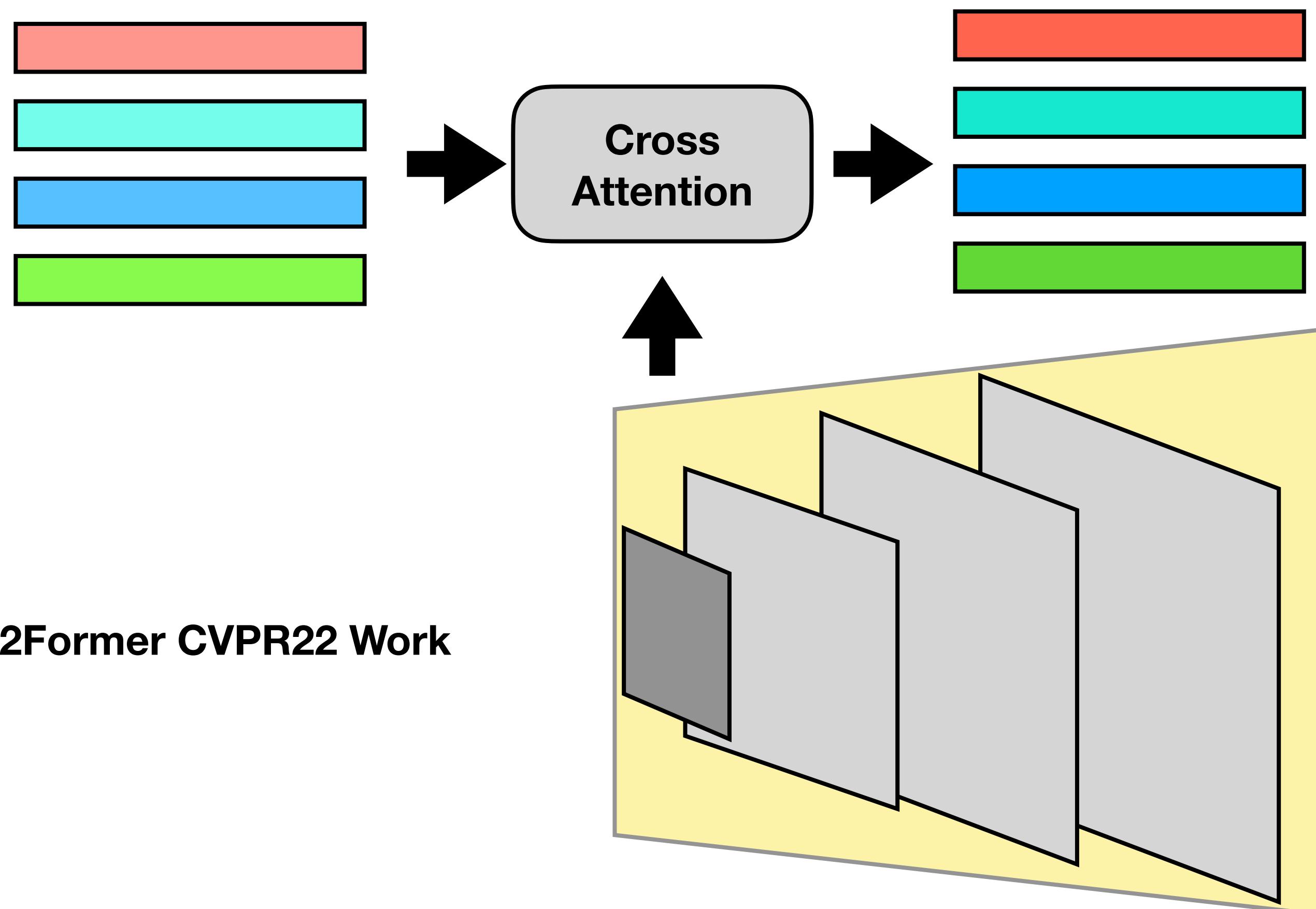
## Multiscale Transformer Decoder



**Mask2Former CVPR22 Work**

# Mask2Former

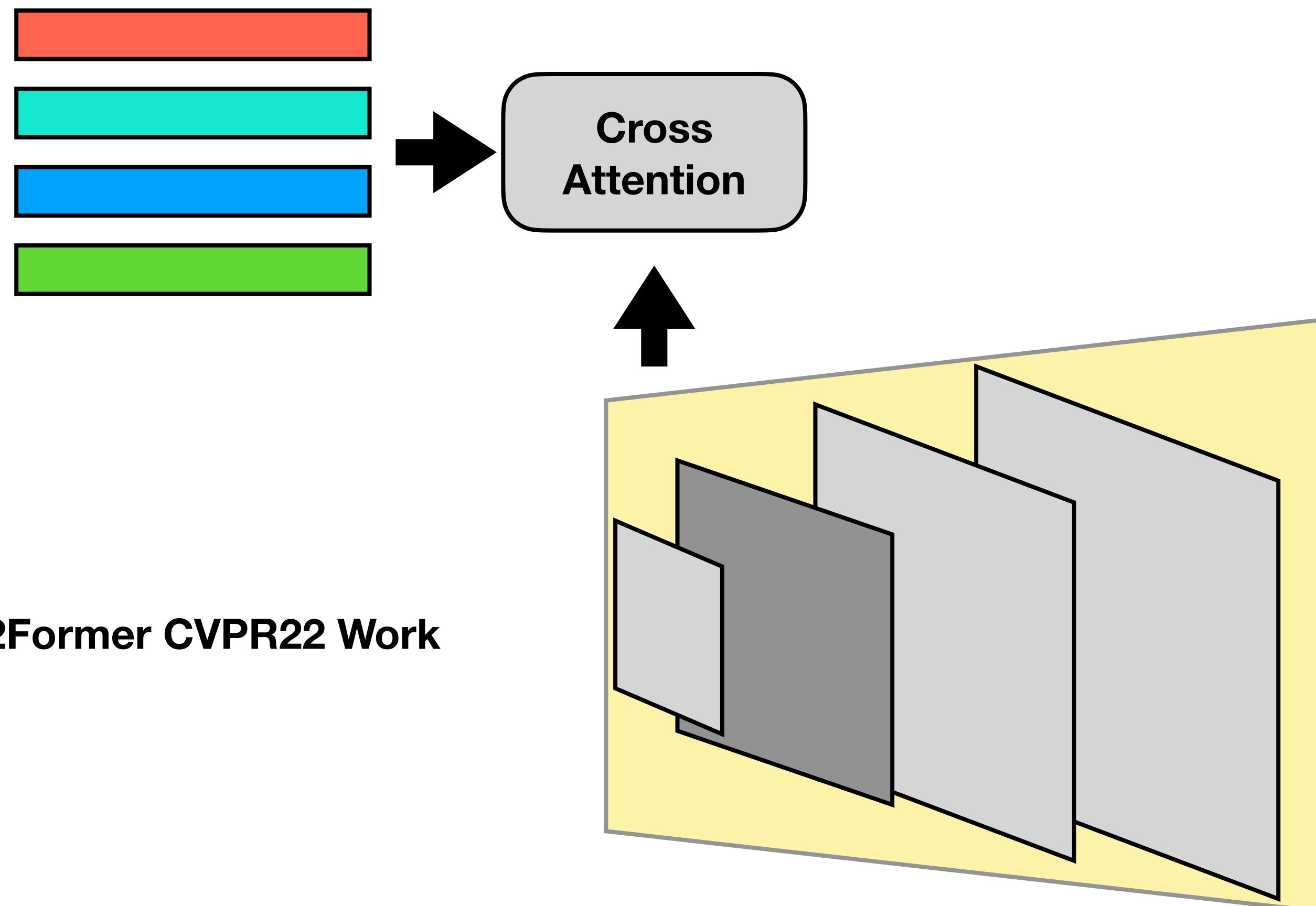
## Multiscale Transformer Decoder



**Mask2Former CVPR22 Work**

# Mask2Former

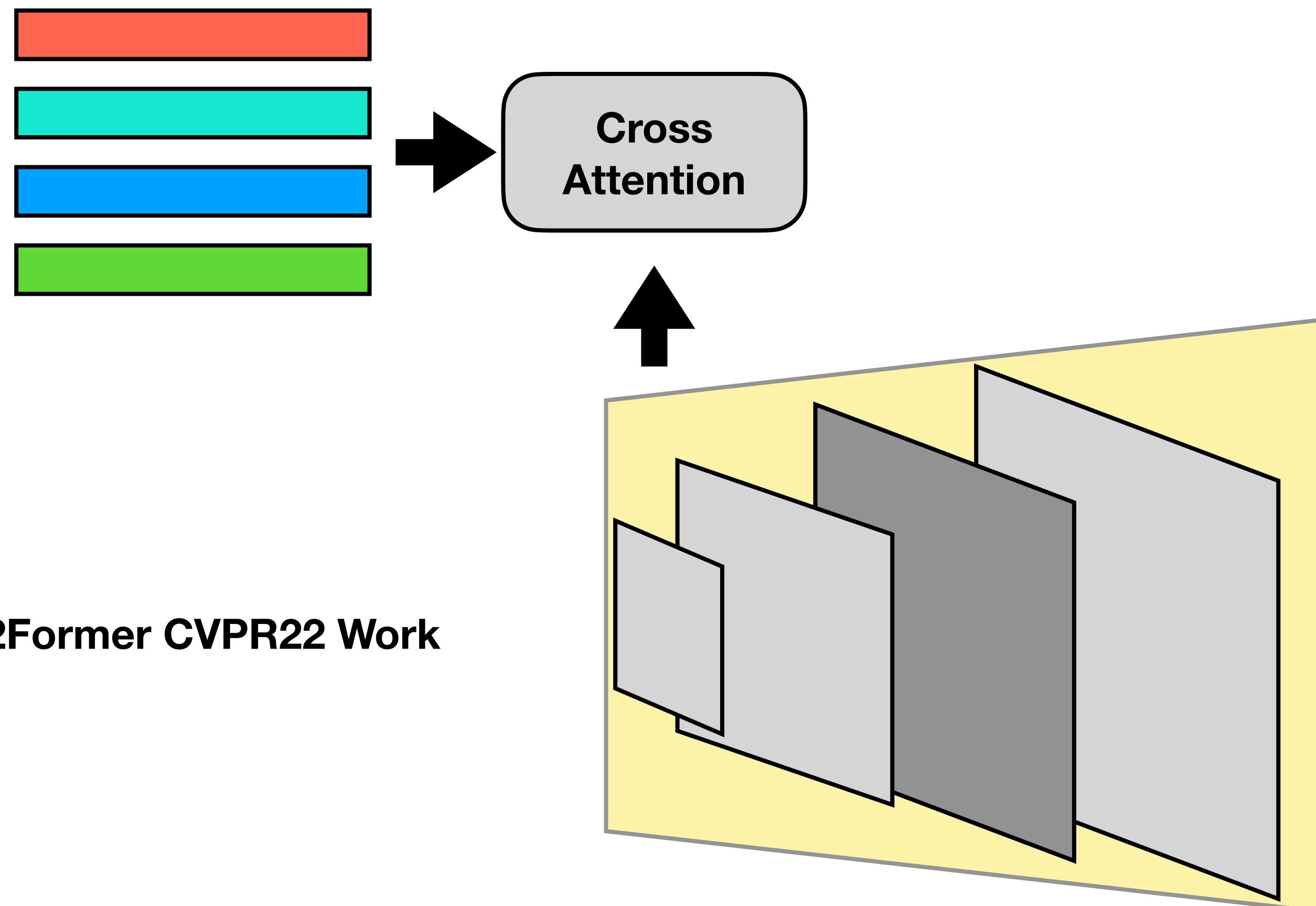
## Multiscale Transformer Decoder



**Mask2Former CVPR22 Work**

# Mask2Former

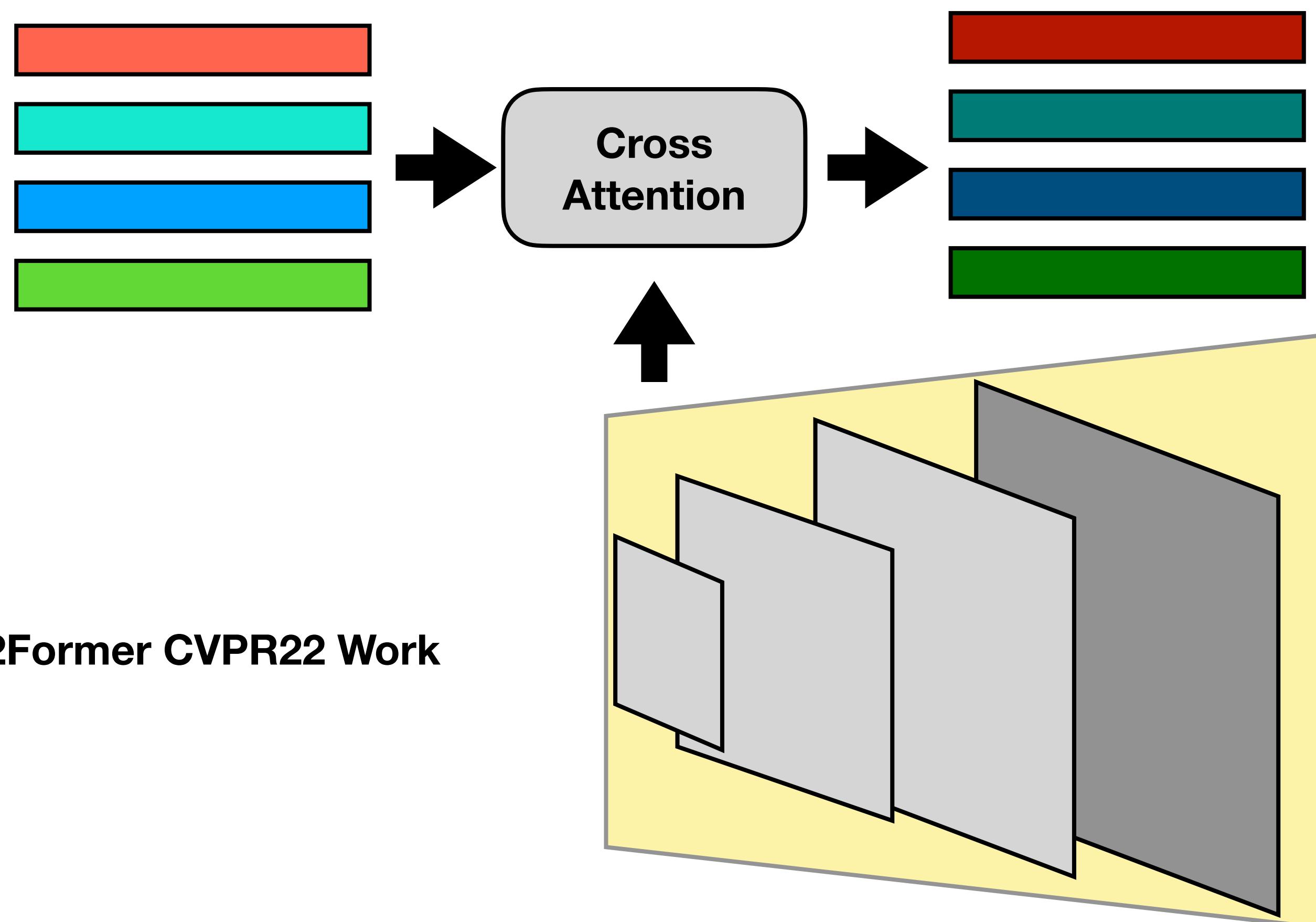
## Multiscale Transformer Decoder



**Mask2Former CVPR22 Work**

# Mask2Former

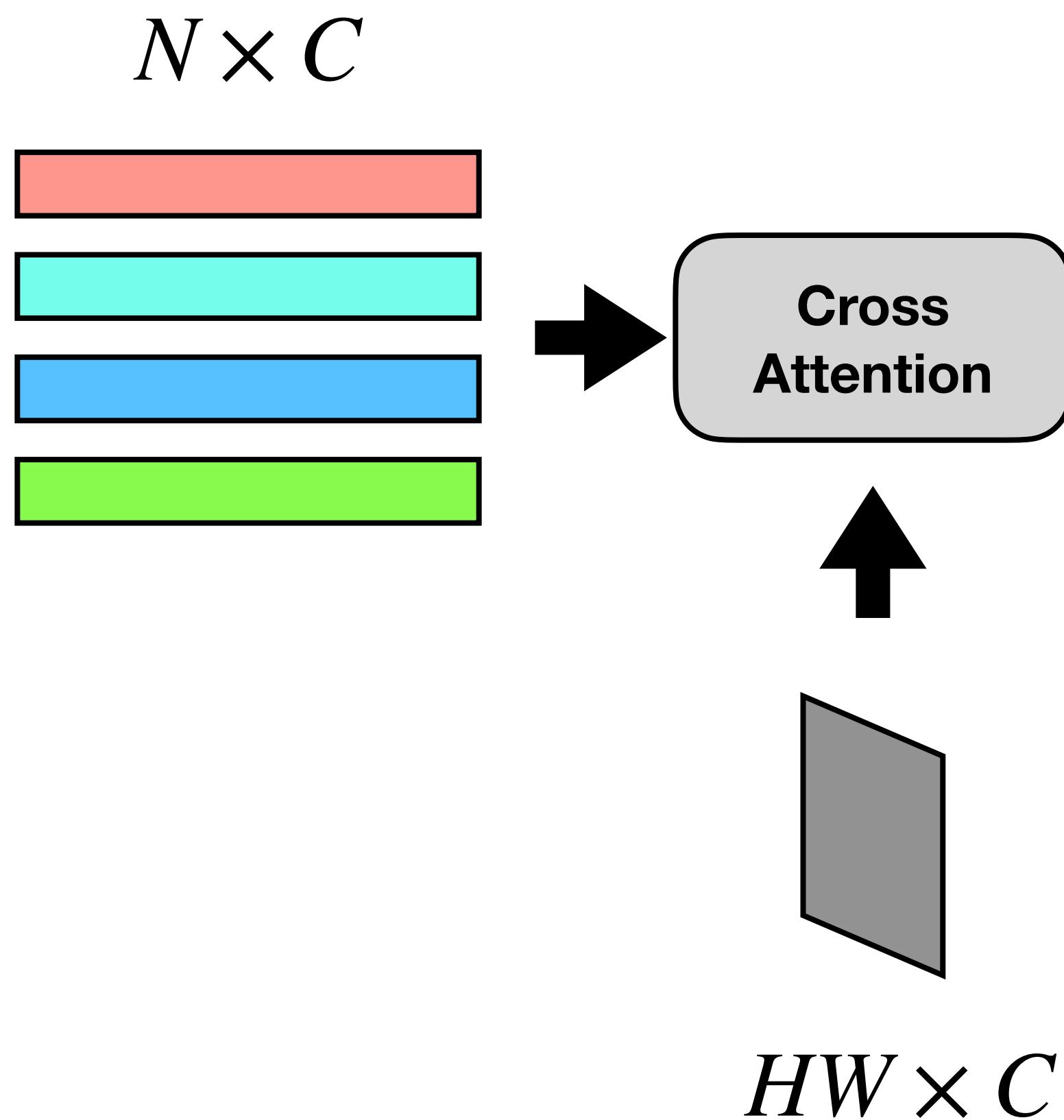
## Multiscale Transformer Decoder



**Mask2Former CVPR22 Work**

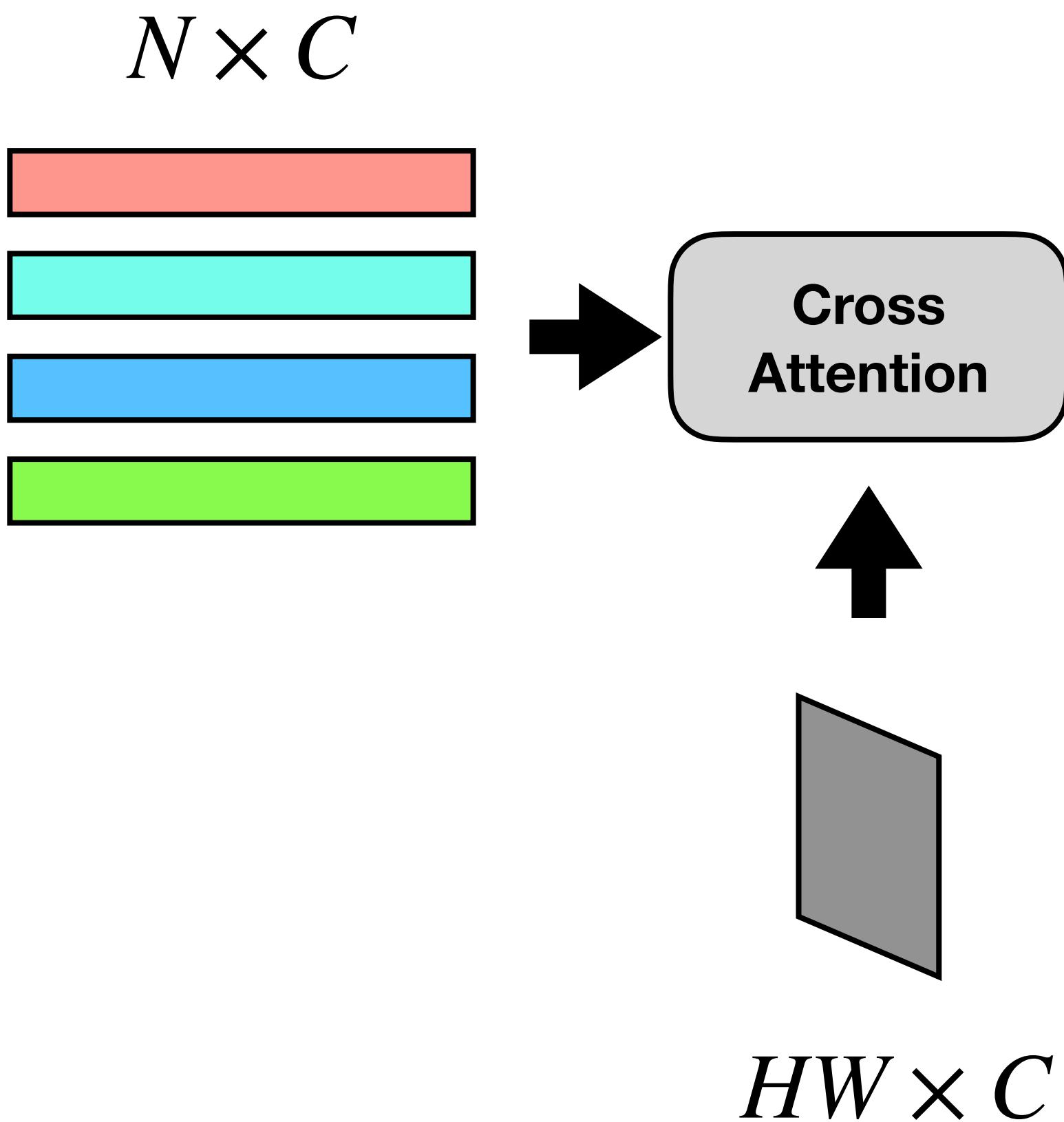
# Mask2Former

## Masked Attention



# Mask2Former

## Masked Attention

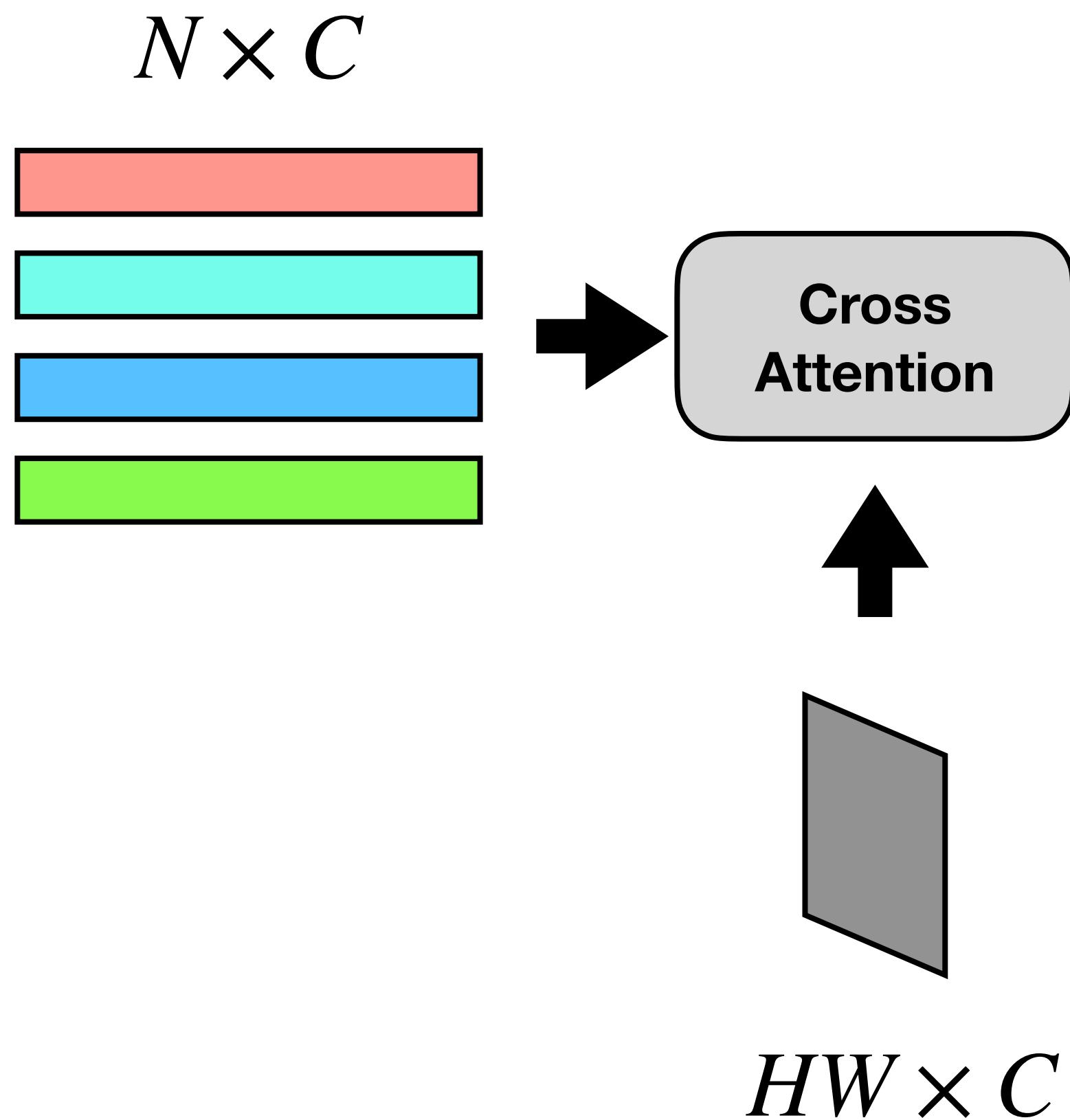


$$\mathbf{X}_l = \text{softmax}(\mathcal{M}_{l-1} + \mathbf{Q}_l \mathbf{K}_l^T) \mathbf{V}_l + \mathbf{X}_{l-1}.$$

$$\mathcal{M}_{l-1}(x, y) = \begin{cases} 0 & \text{if } \mathbf{M}_{l-1}(x, y) = 1 \\ -\infty & \text{otherwise} \end{cases}$$

# Mask2Former

## Masked Attention



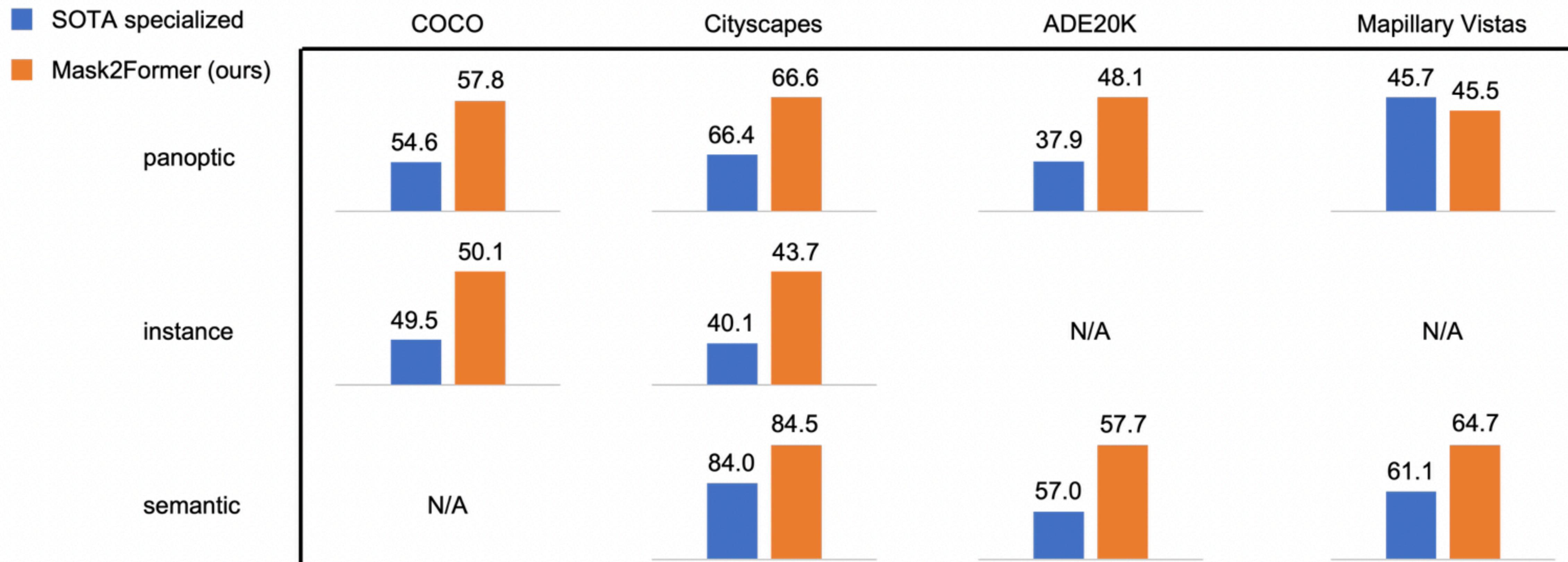
$M_{l-1}$   $N \times HW$   
Binary Mask Predictions {0, 1}

$$\mathbf{X}_l = \text{softmax}(\mathcal{M}_{l-1} + \mathbf{Q}_l \mathbf{K}_l^T) \mathbf{V}_l + \mathbf{X}_{l-1}.$$

$$\mathcal{M}_{l-1}(x, y) = \begin{cases} 0 & \text{if } \mathbf{M}_{l-1}(x, y) = 1 \\ -\infty & \text{otherwise} \end{cases}$$

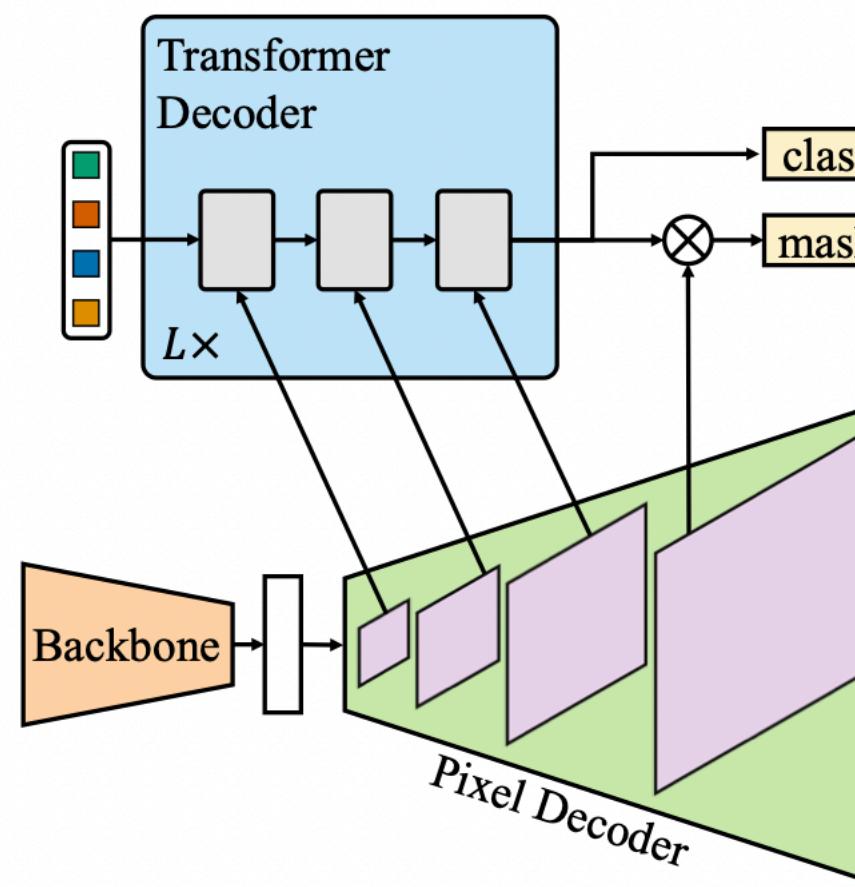
# Mask2Former

## Masked Cross Attention

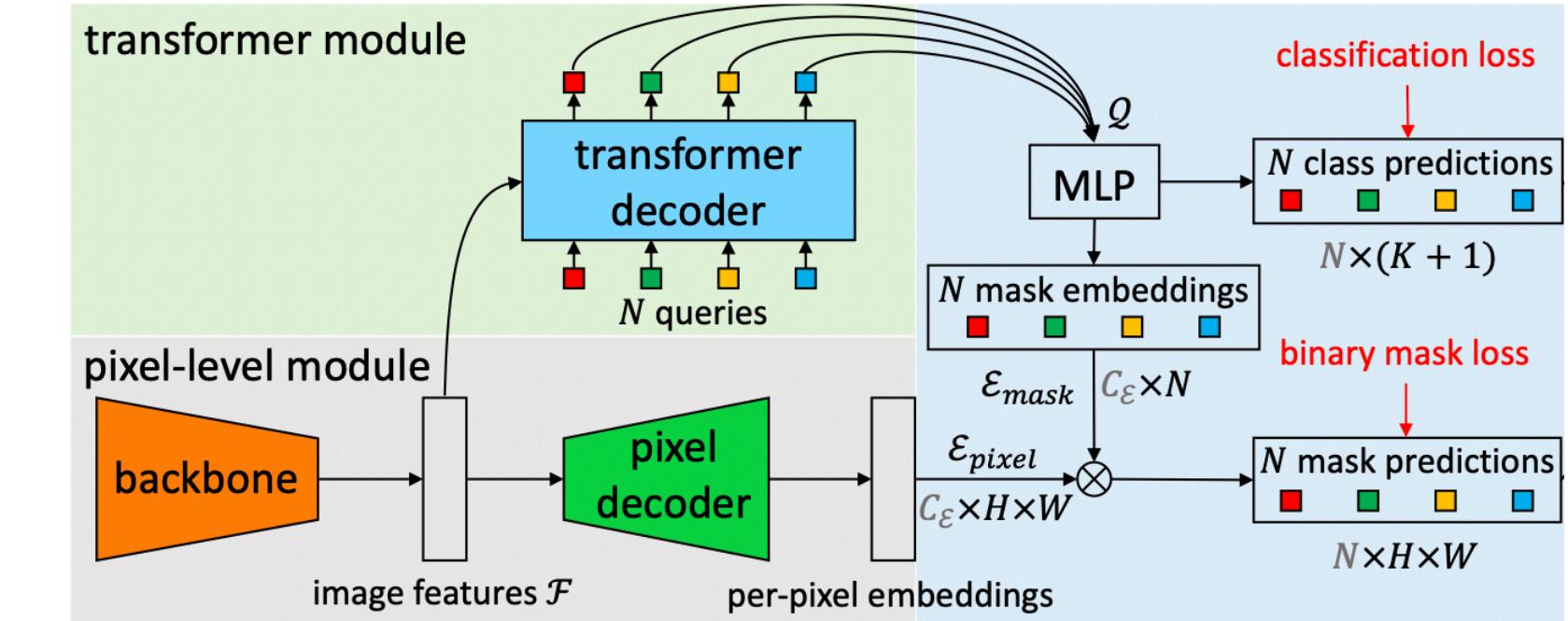


# Questions

**Mask2Former, CVPR'22**



**MaskFormer, NeurIPS'21**



**MED-VT, CVPR'23**

