Self-Attention and Transformer

Xiaolong Wang

Previous classes

- RNN for time sequence processing

- Video understanding with temporal convolution, 3D CNN

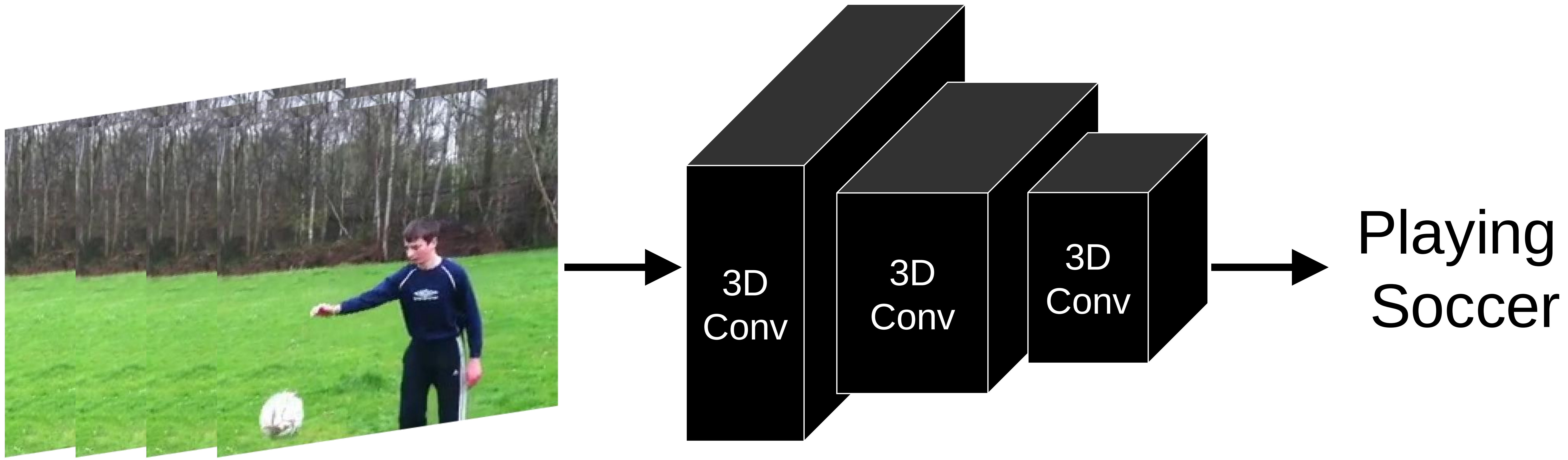
This Class

- Non-local Neural Network for Videos

- Self-Attention and Transformer for NLP

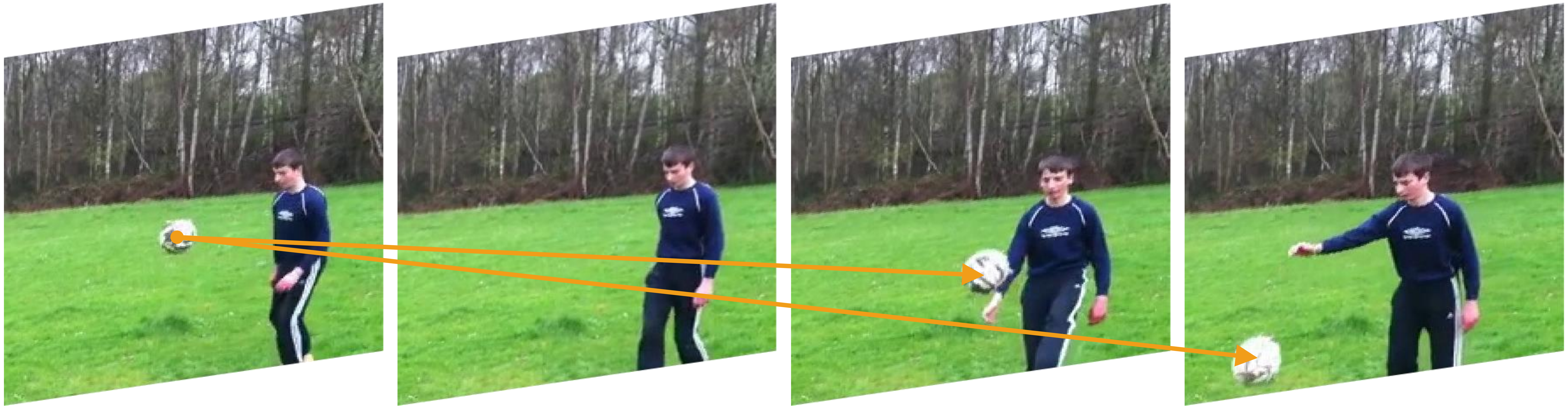
Non-local Neural Network for Videos

Video Recognition



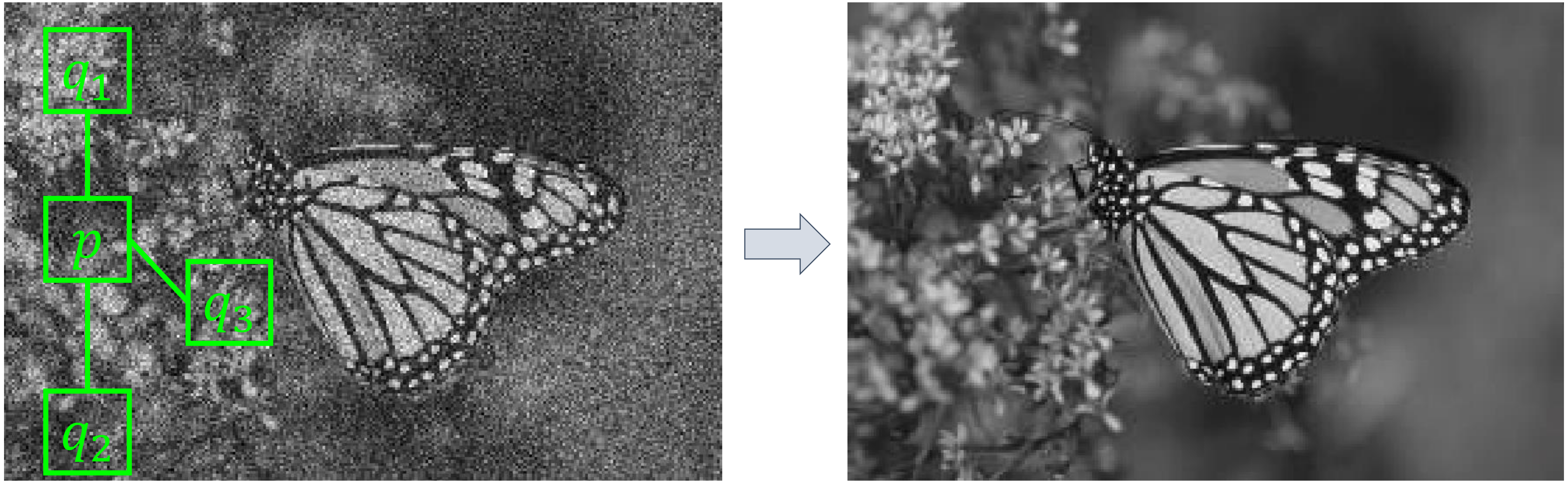
Reasoning for Action Recognition

Long-rang explicit reasoning



Wang et al., 2018.

Non-local Means

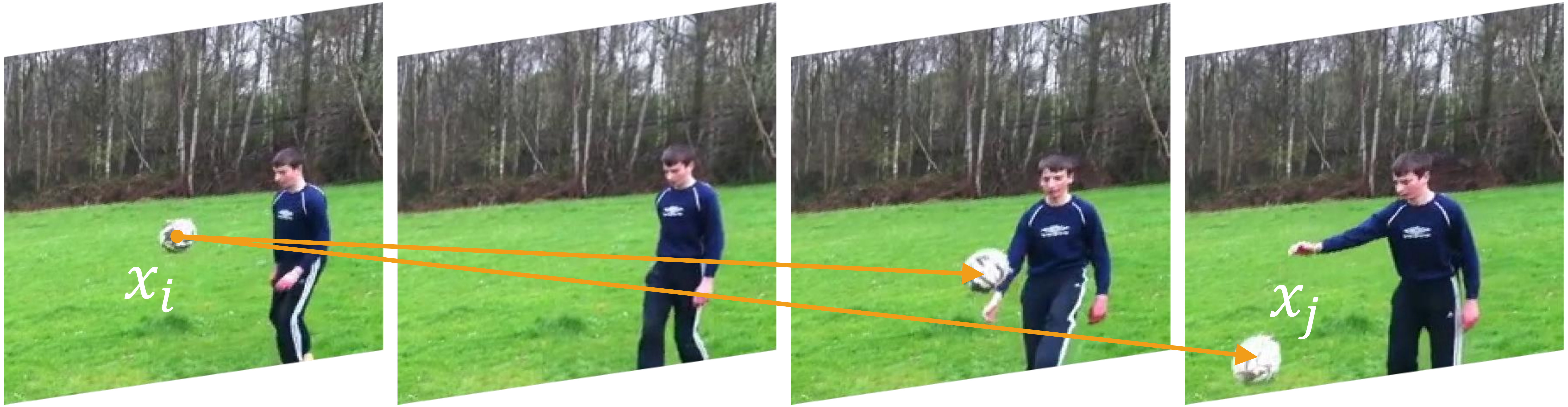


Buades et al., 2005.

Non-local Operator

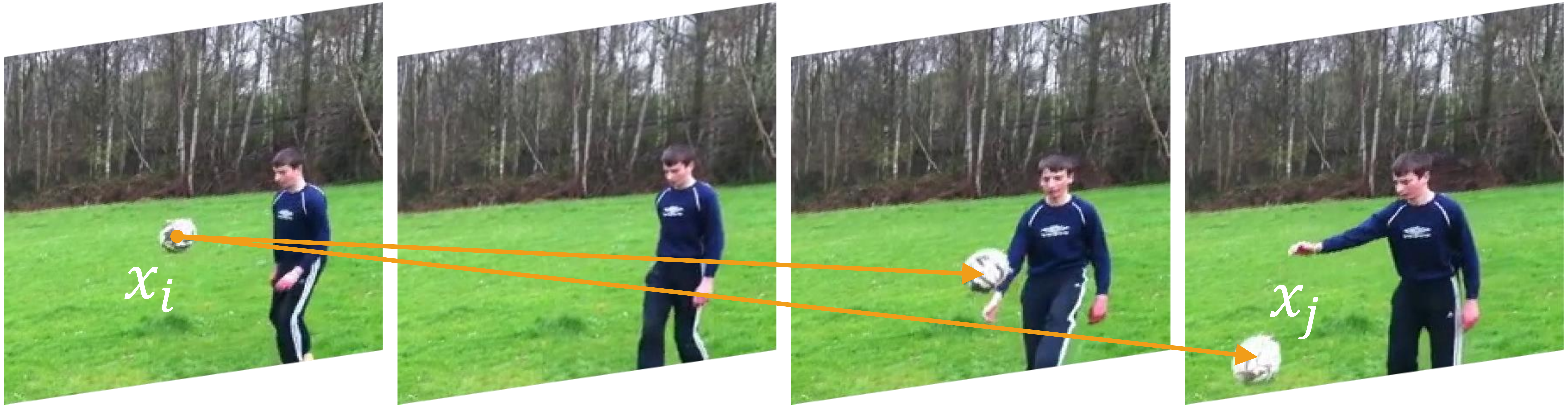
Operation in feature space

Can be embedded into any ConvNets



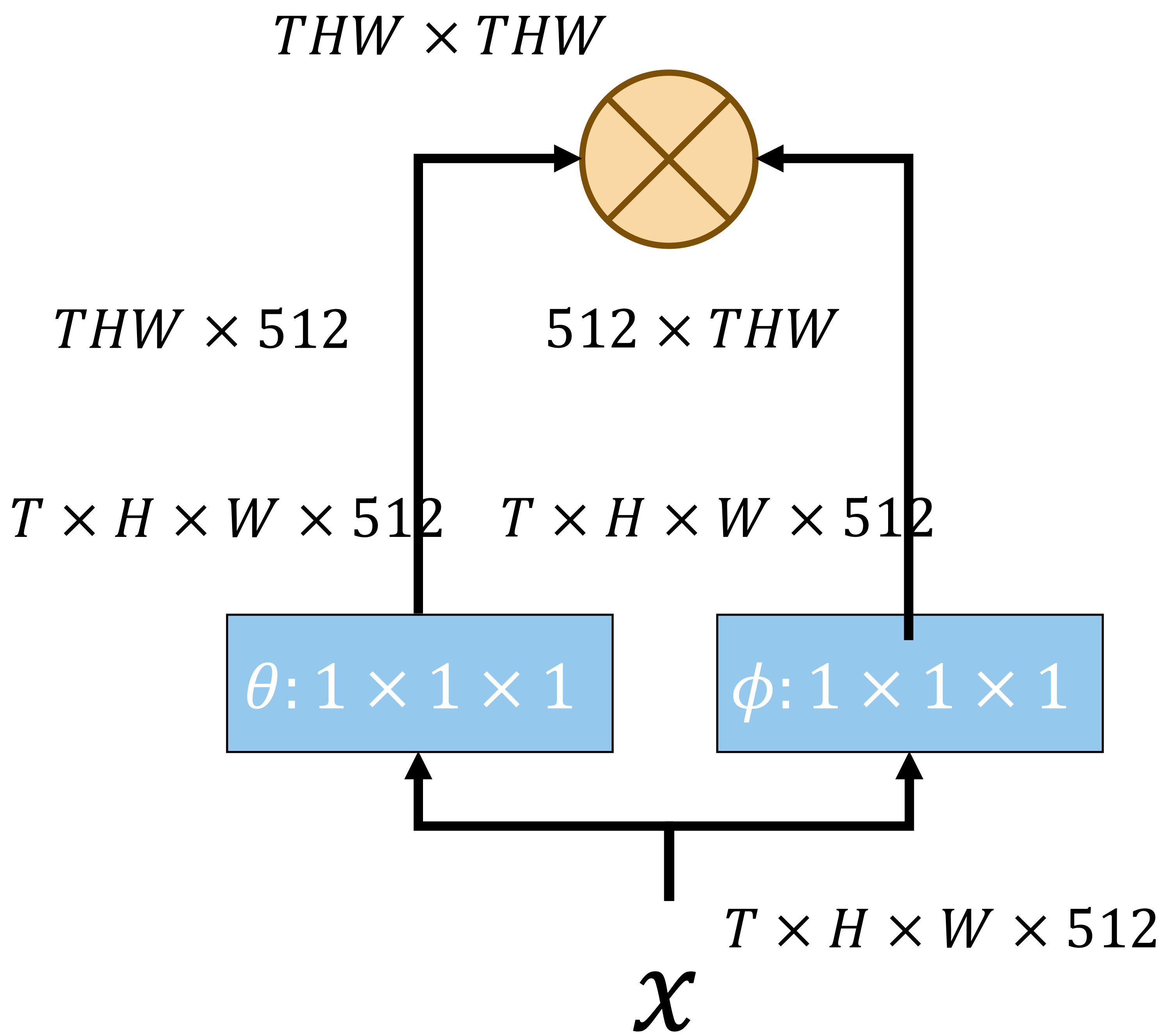
Non-local Operator

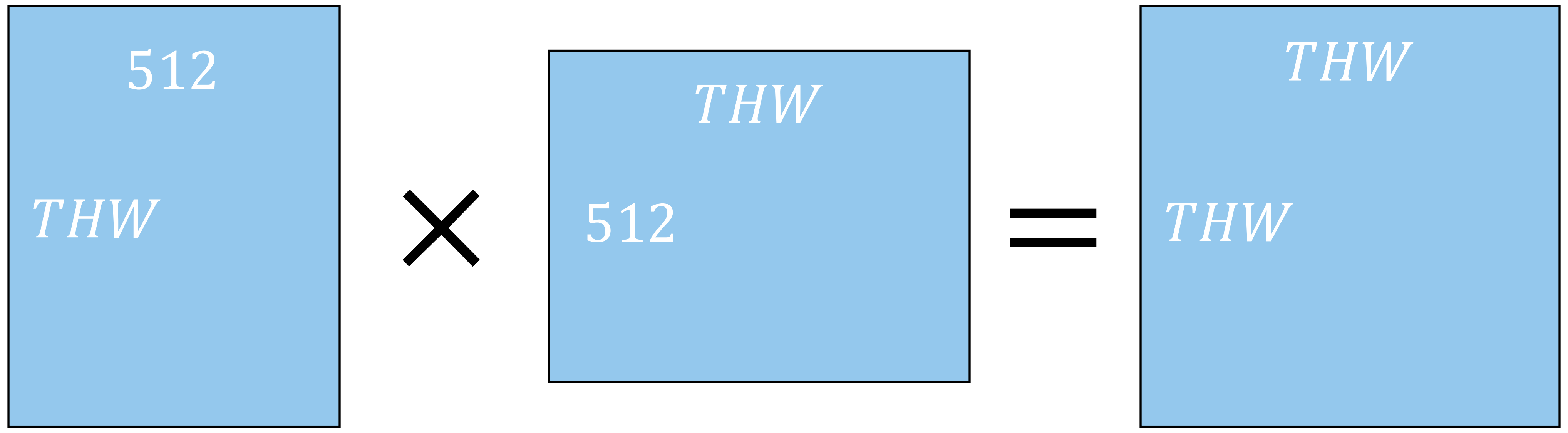
$$y_i = \frac{1}{C(x)} \sum_{\forall j} \underbrace{f(x_i, x_j)}_{\text{Affinity}} \; \underbrace{g(x_j)}_{\text{Features}}$$



Non-local Operator

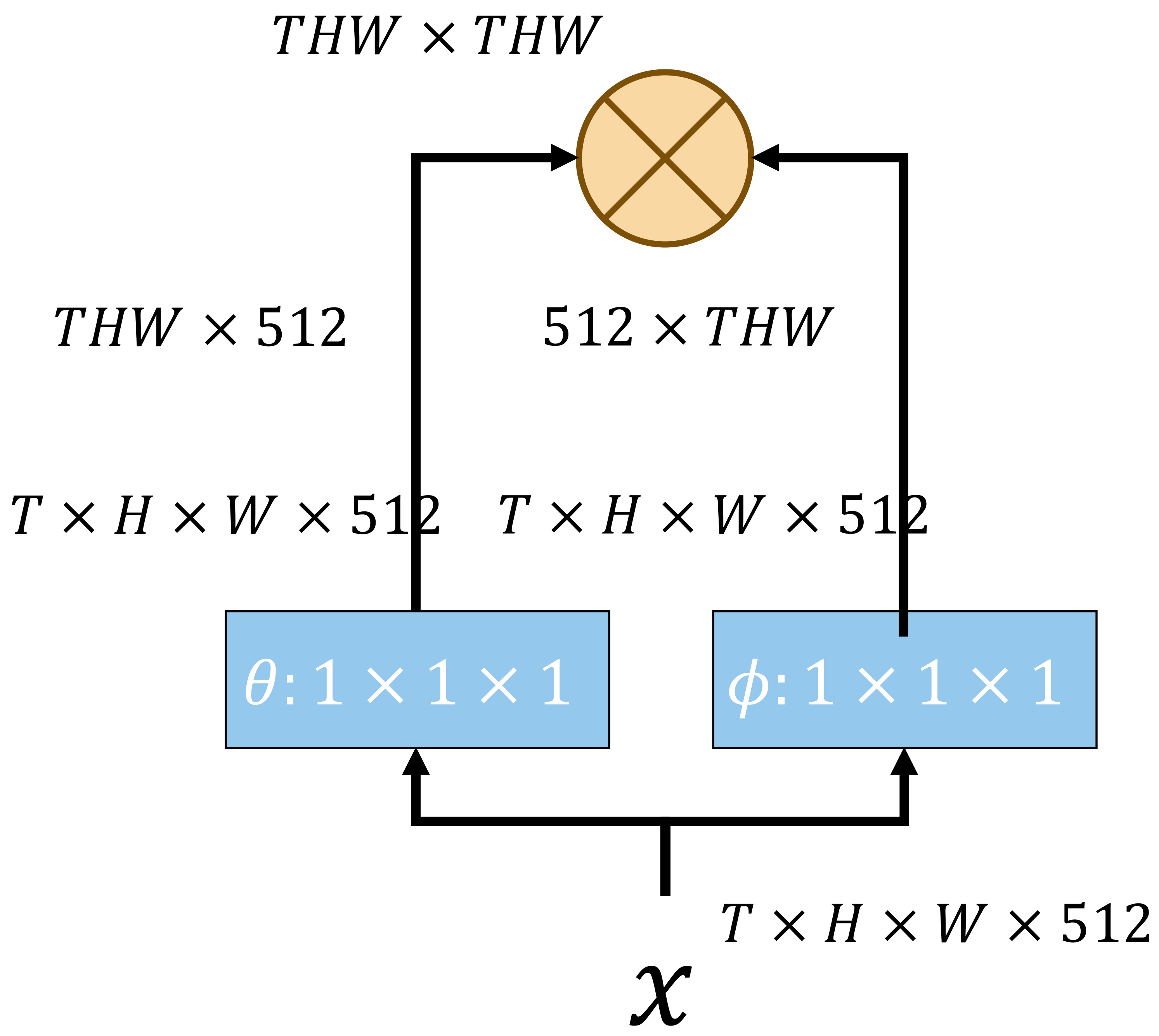
$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j)\ g(x_j)$$

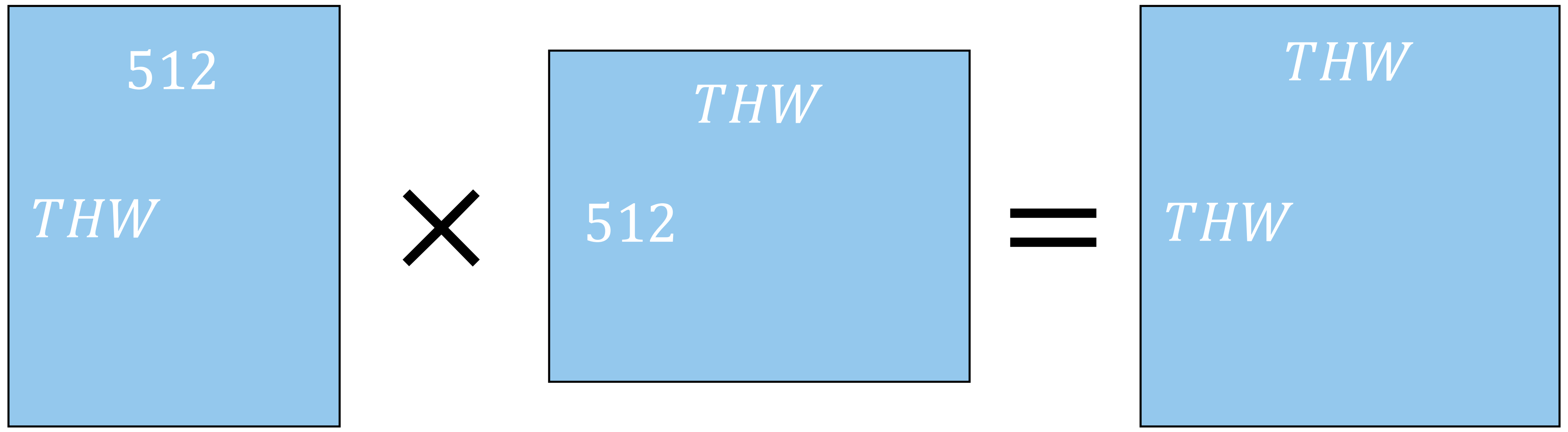




Non-local Operator

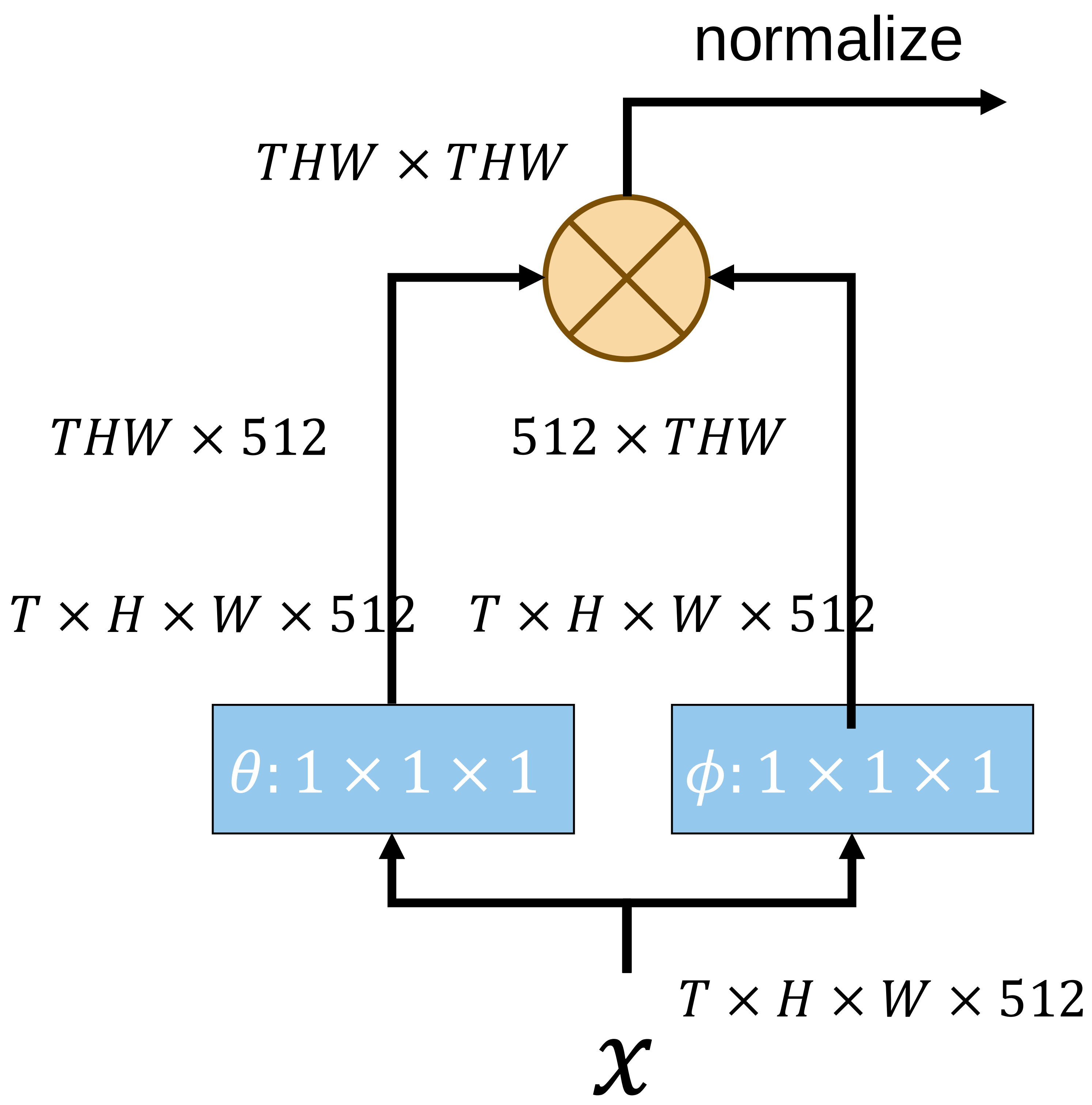
$$y_i = \frac{1}{C(x)} \sum_{\forall j} f\left(x_i, x_j\right)\ g(x_j)$$





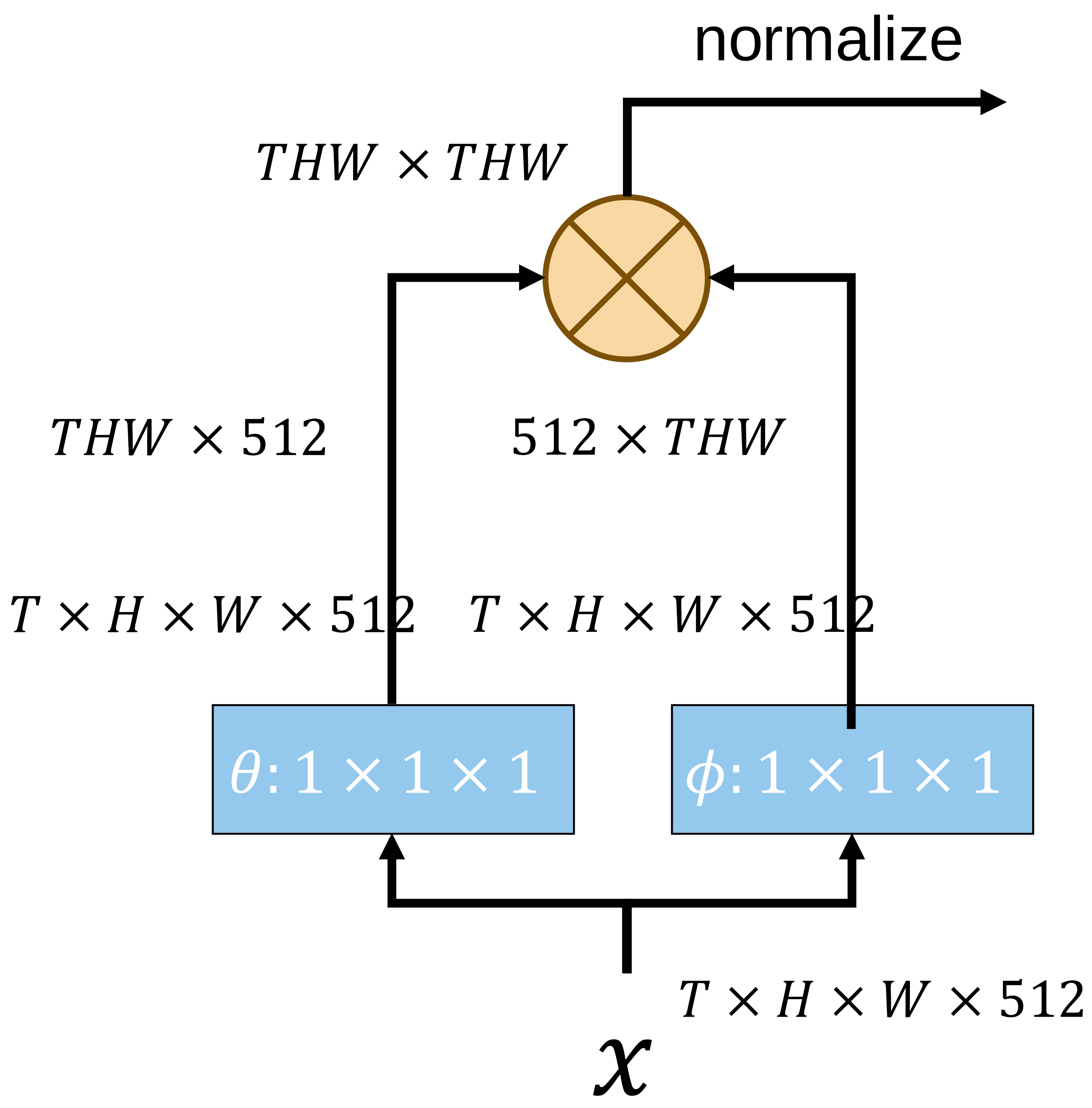
Non-local Operator

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f\left(x_i, x_j\right)\ g(x_j)$$



Non-local Operator

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j)\ g(x_j)$$



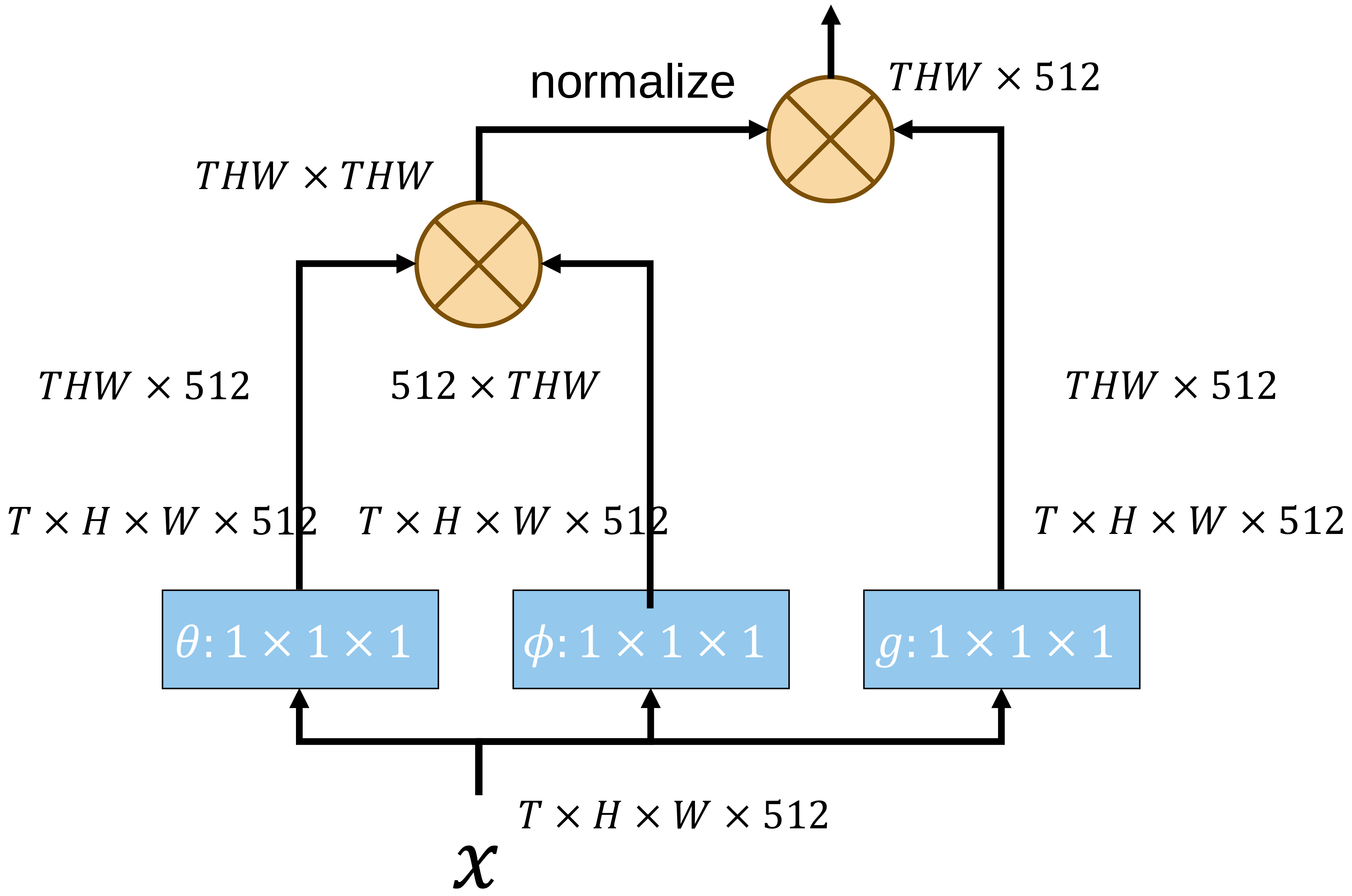
$$f(x_i, x_j) = \exp(x_i^T x_j)$$

$$C(x) = \sum_{\forall j} f(x_i, x_j)$$

$$\frac{f(x_i, x_j)}{C(x)} = \frac{\exp(x_i^T x_j)}{\sum_{\forall j} \exp(x_i^T x_j)}$$

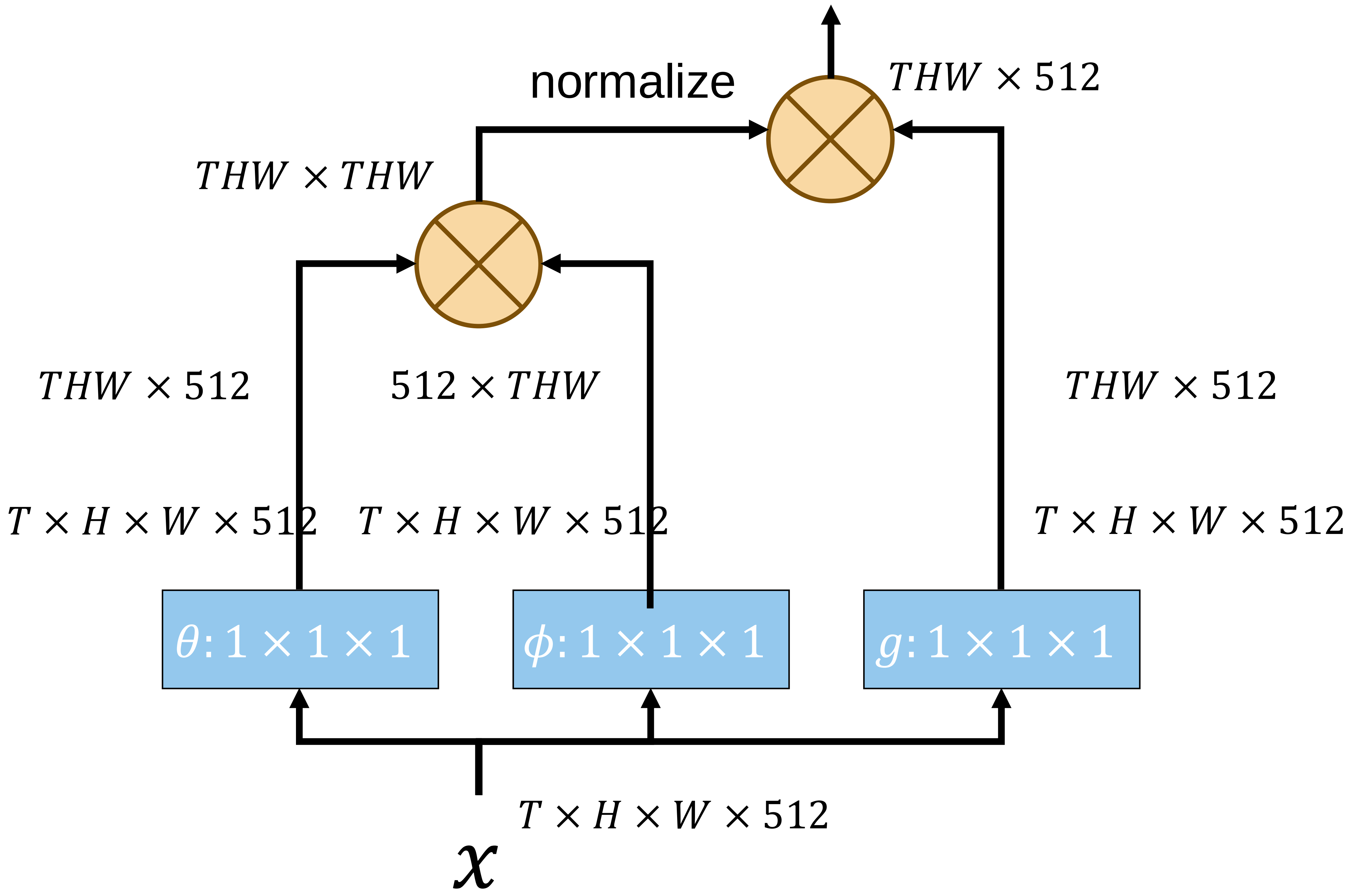
Non-local Operator

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j)\ g(x_j)$$



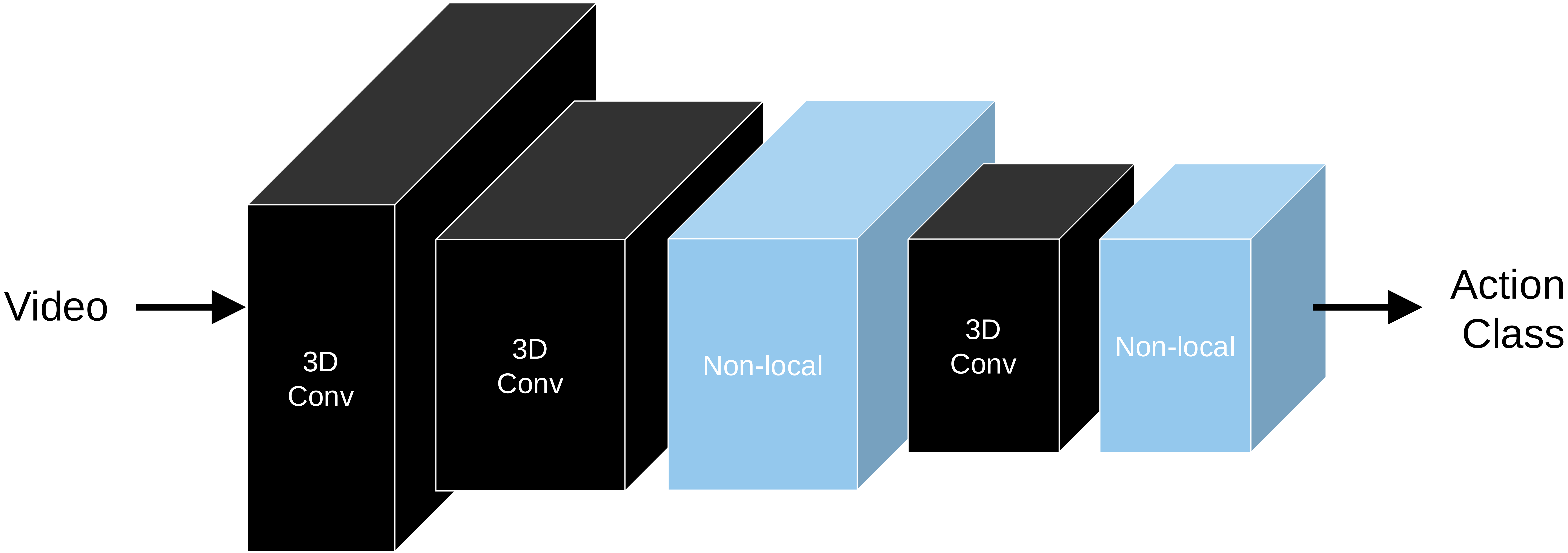
Non-local Operator

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f\left(x_i, x_j\right)\ g(x_j)$$

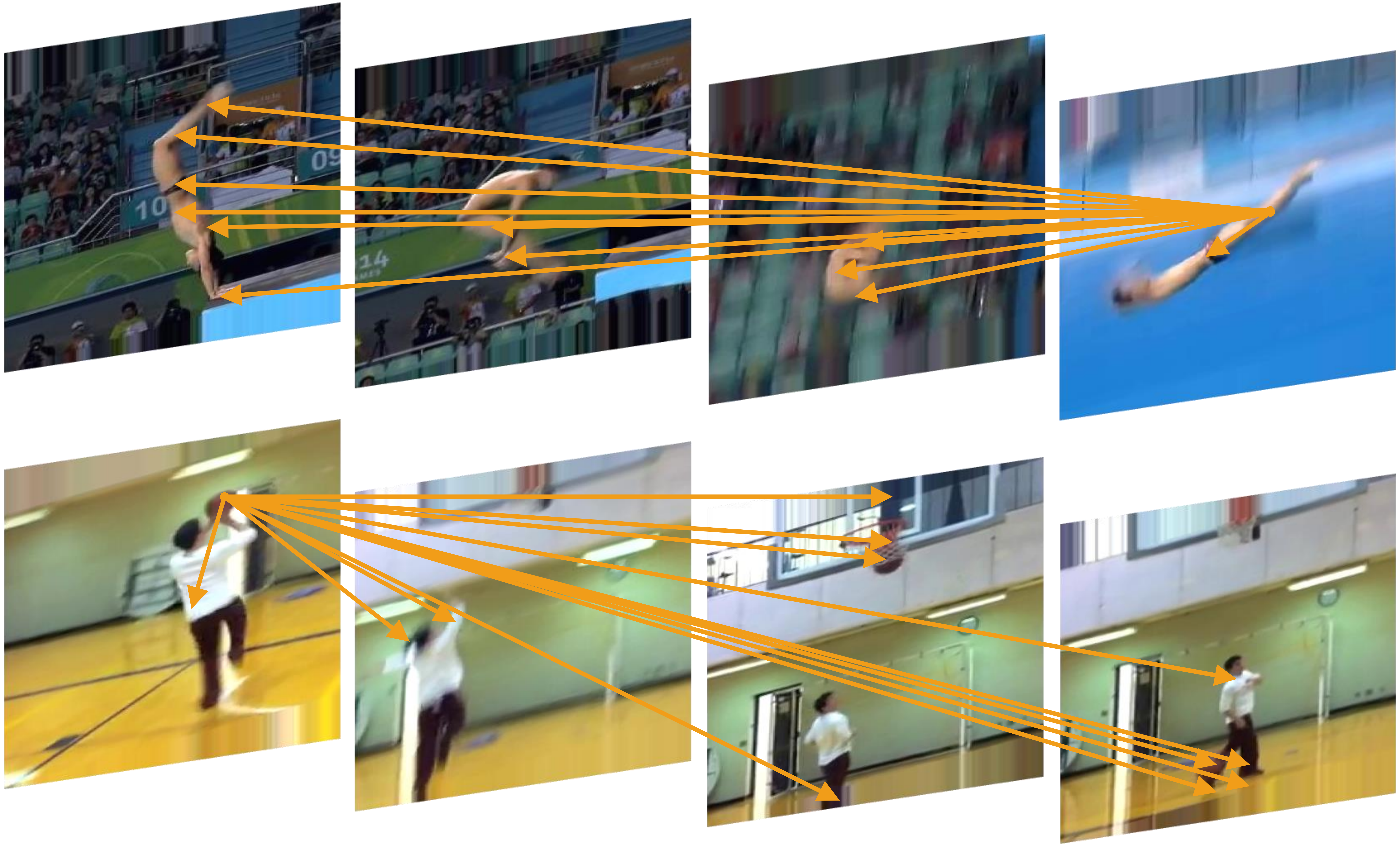


Non-local Operator as A Residual Block

$$z_i = y_i W + x_i$$



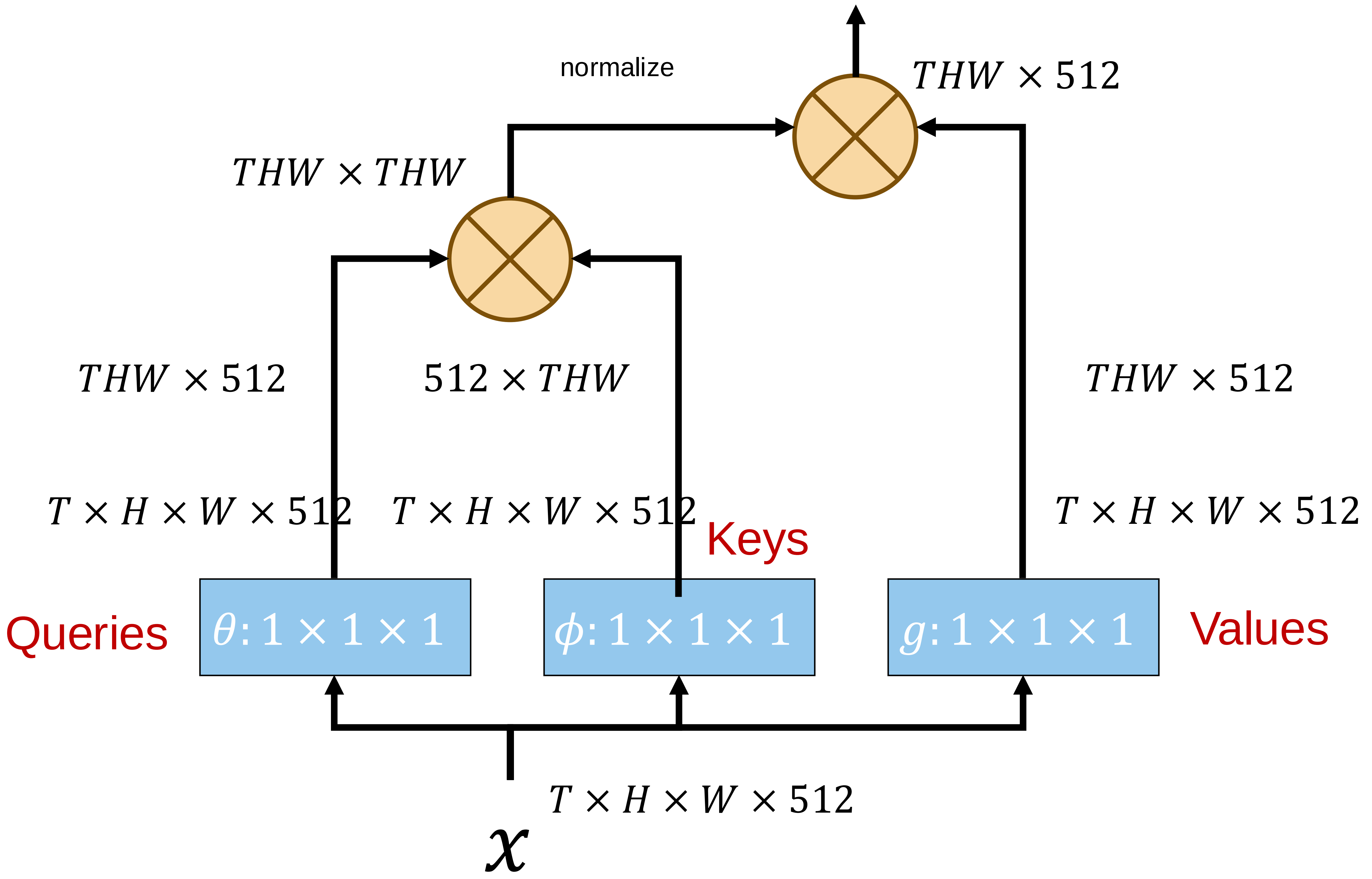
Examples



Self-Attention and Transformer for NLP

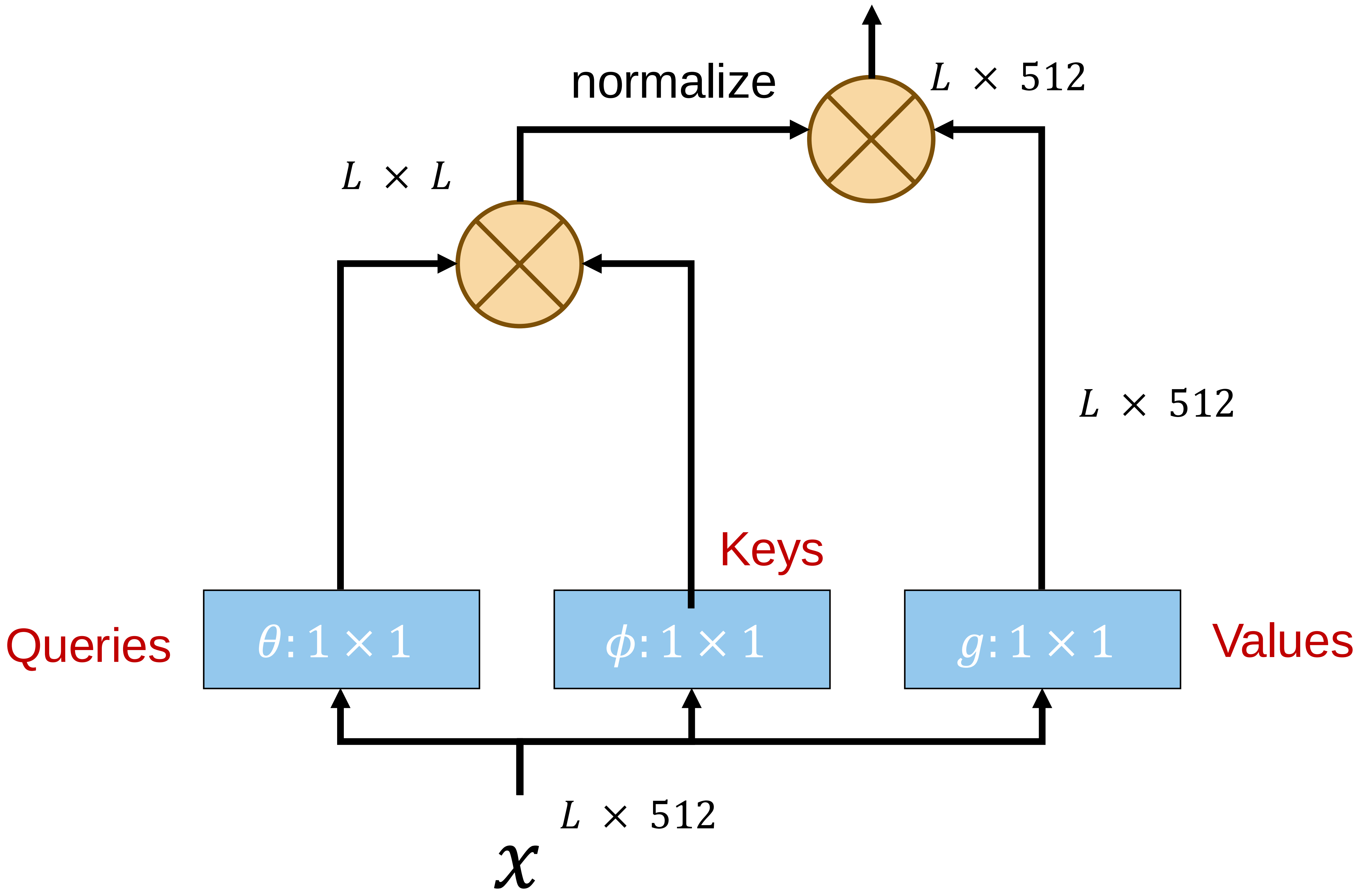
Self-Attention Operator

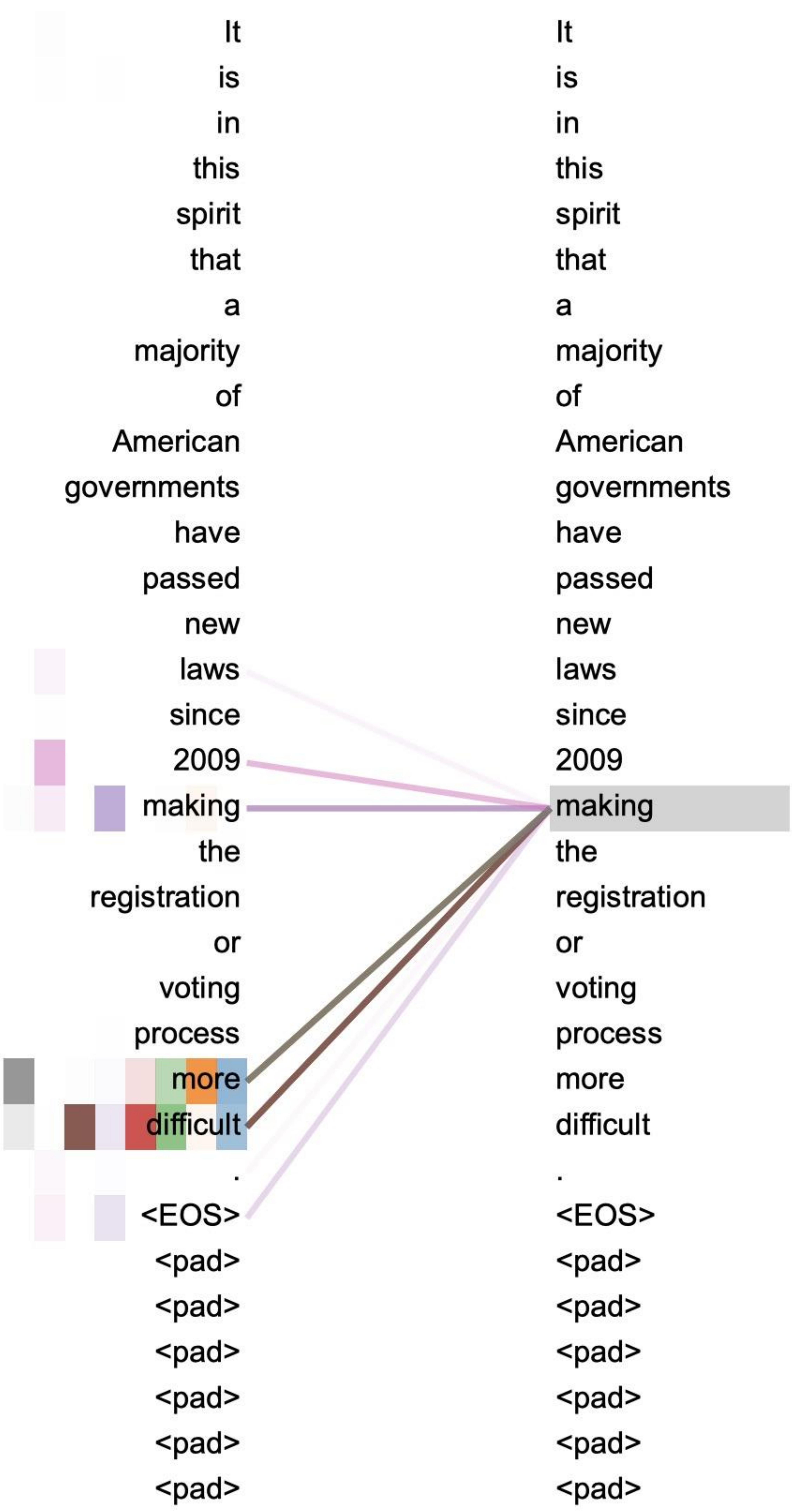
$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j)\ g(x_j)$$



Self-Attention Operator

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f\left(x_i, x_j\right)\ g(x_j)$$

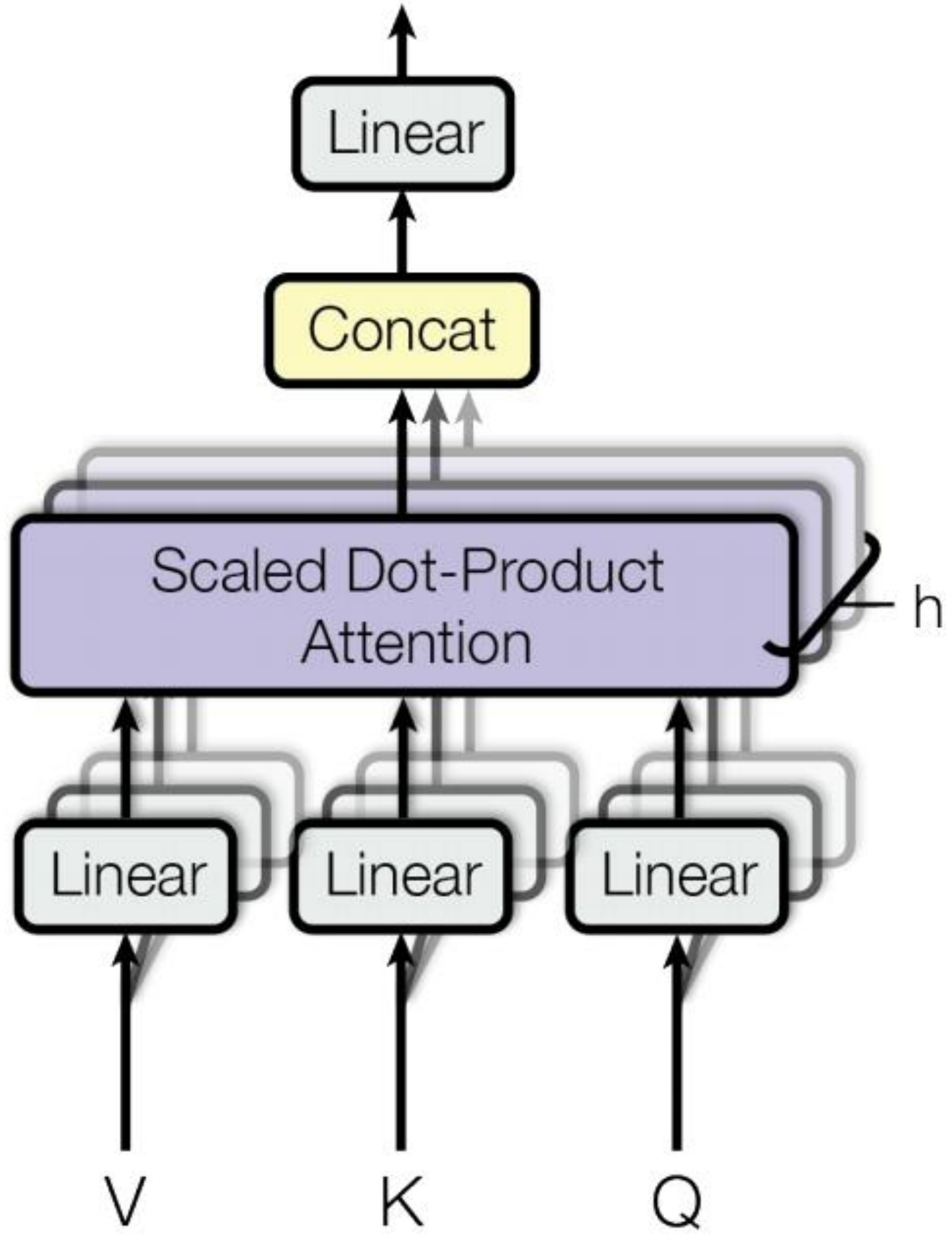




It
is
in
this
spirit
that
a
majority
of
American
governments
have
passed
new
laws
since
2009
making
the
registration
or
voting
process
more
difficult
.
<EOS>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>

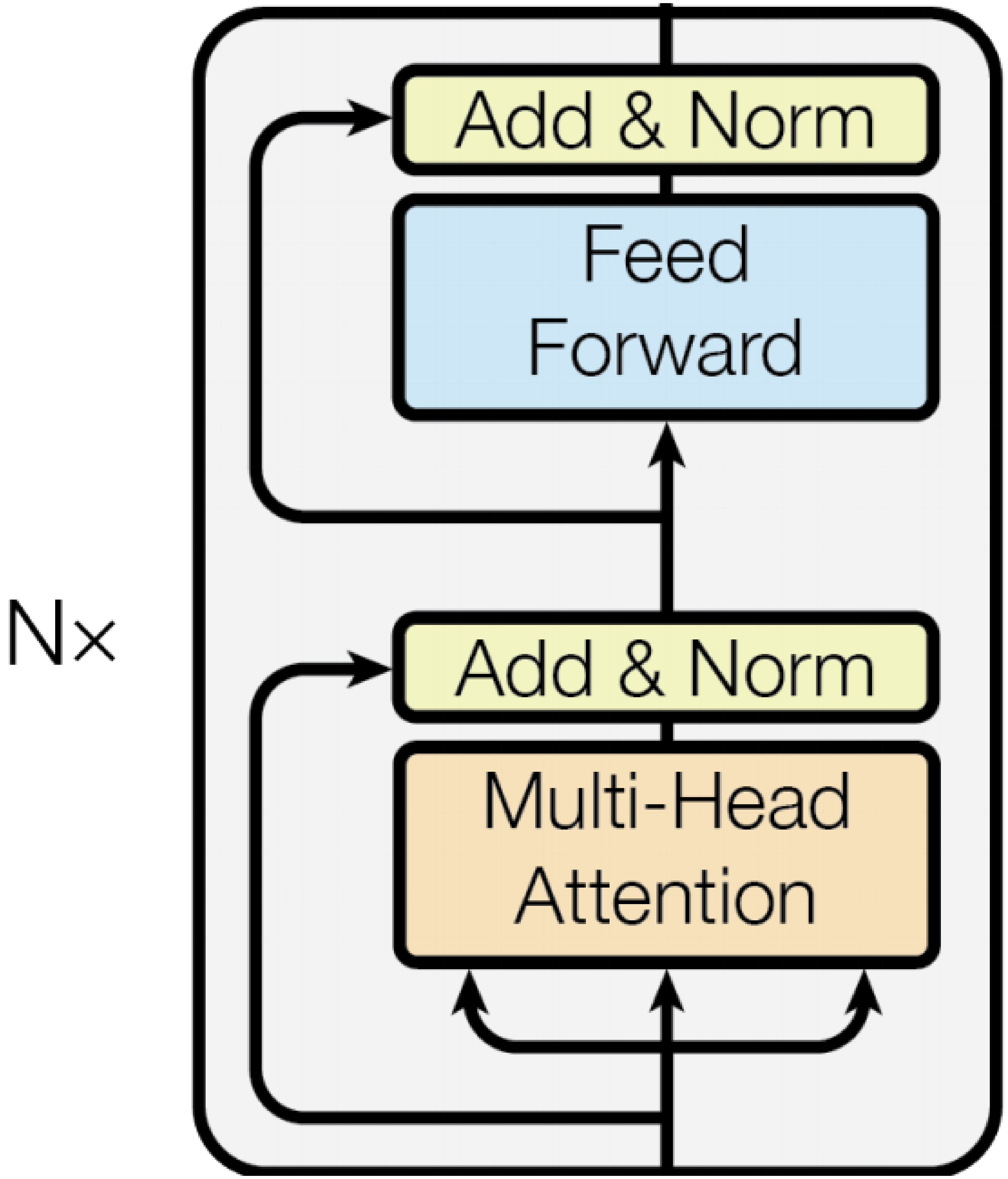
Multi-head attention

- Run h attention models in parallel on top of different linearly projected versions of Q, K, V; concatenate and linearly project the results
- Intuition: enables model to attend to different kinds of information at different positions



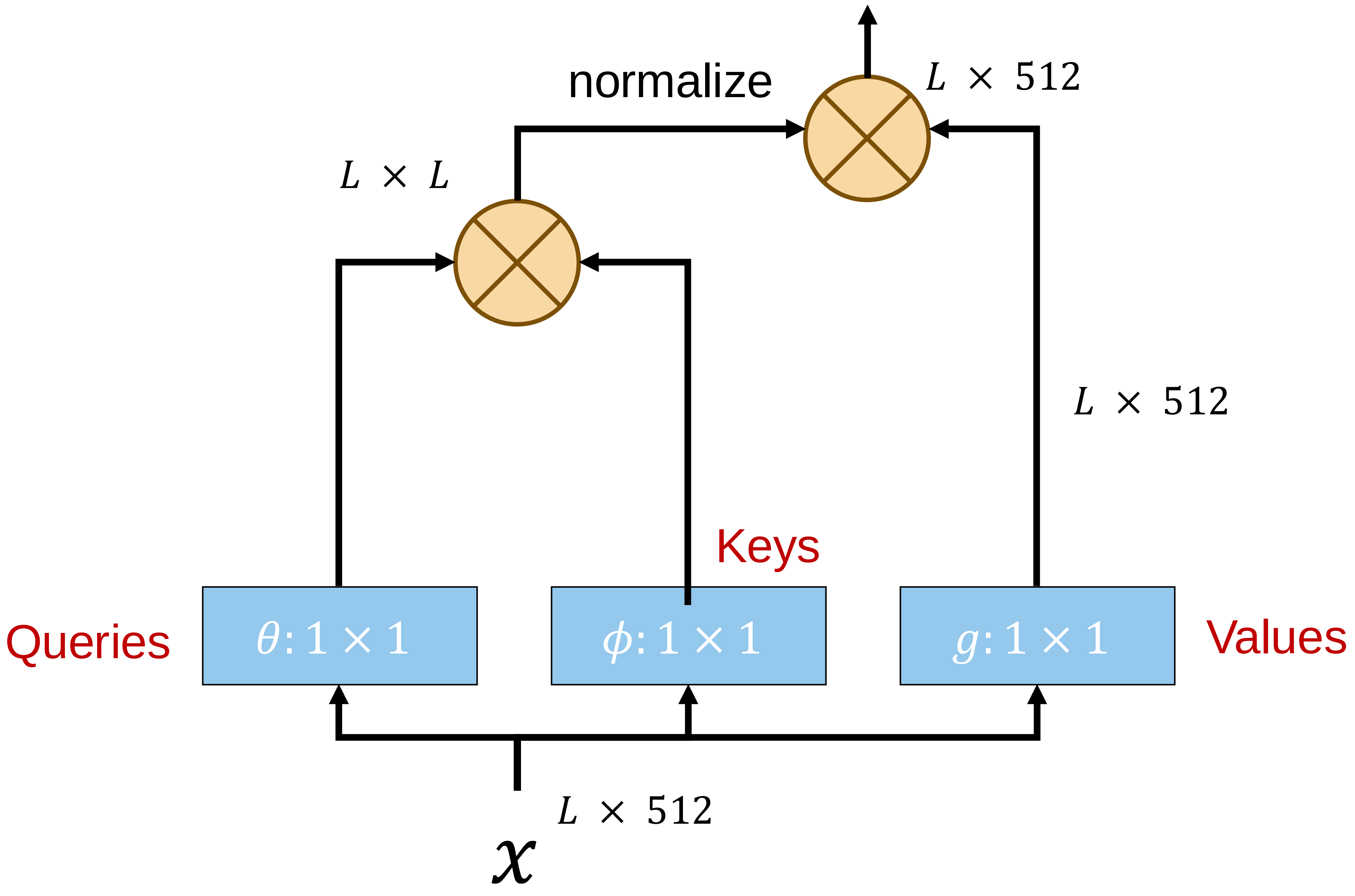
Transformer blocks

- A **Transformer** is a sequence of transformer blocks
  - Vaswani et al.: N=12 blocks, embedding dimension = 512, 6 attention heads
  - **Add & Norm:** residual connection followed by [layer normalization]
  - **Feedforward:** two linear layers with ReLUs in between, applied independently to each vector
- Attention is the only interaction between inputs!



Positional encoding

Self-attention does not encode the order of the inputs.



$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j)\ g(x_j)$$

Positional encoding

To give transformer information about ordering of tokens, add function of position (based on sines and cosines) to every input

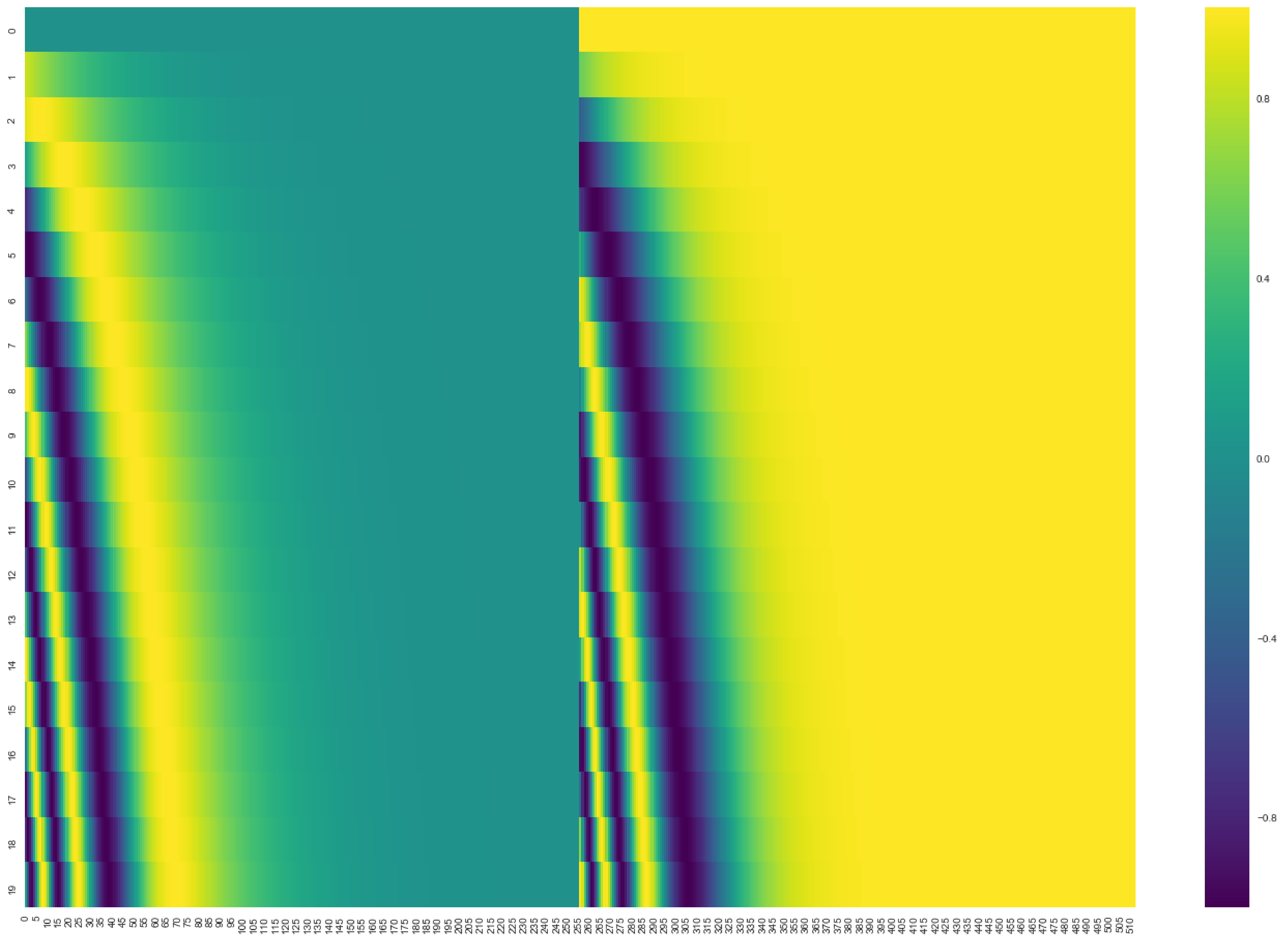
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

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

Positional encoding

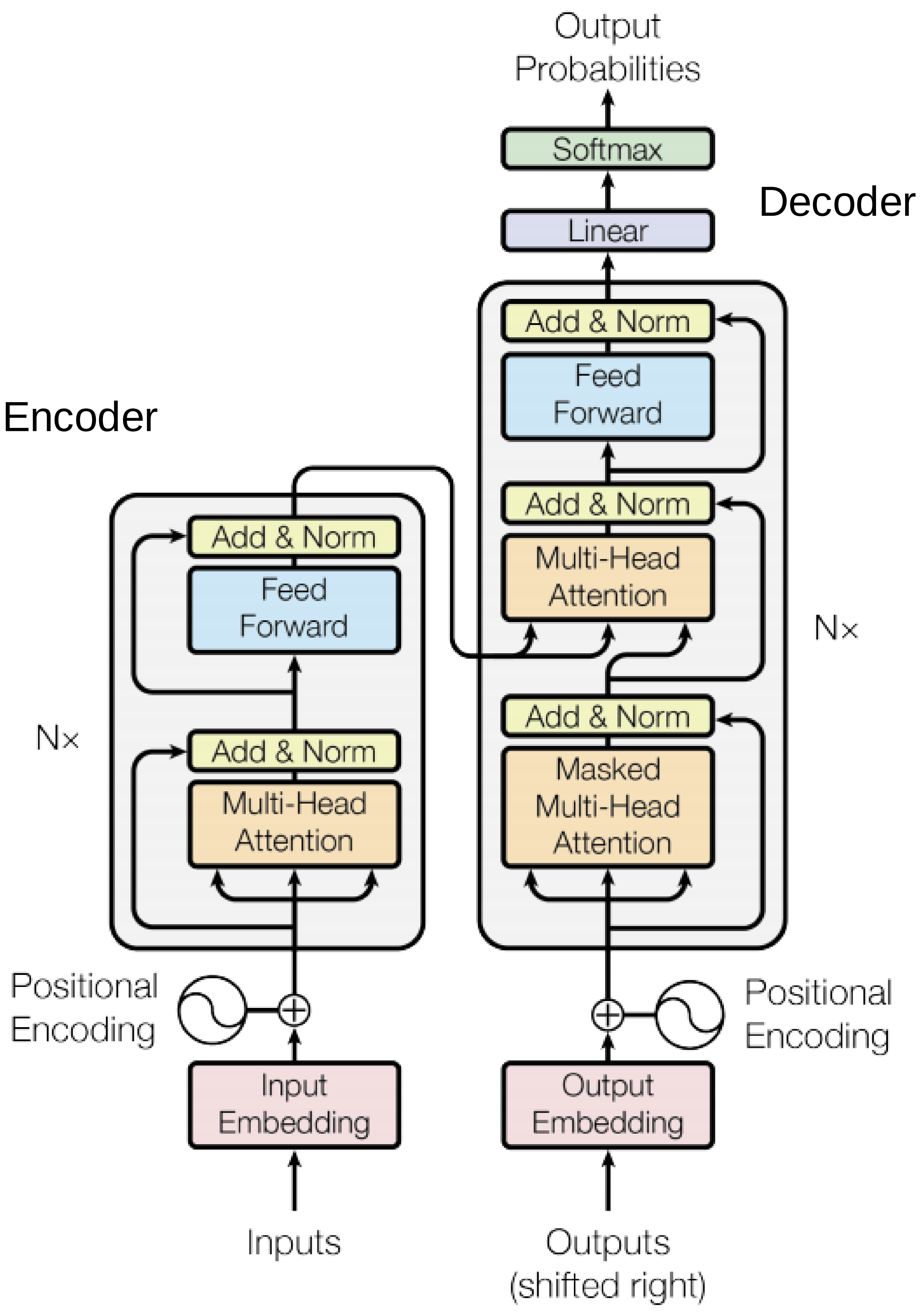
To give transformer information about ordering of tokens, add function of position (based on sines and cosines) to every input

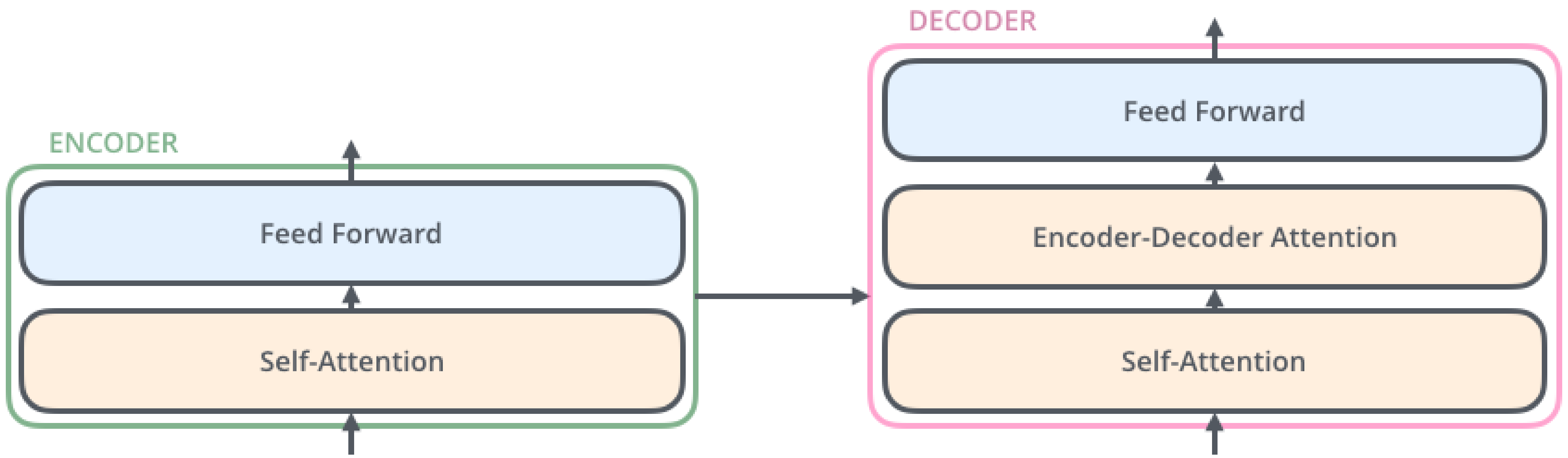
position



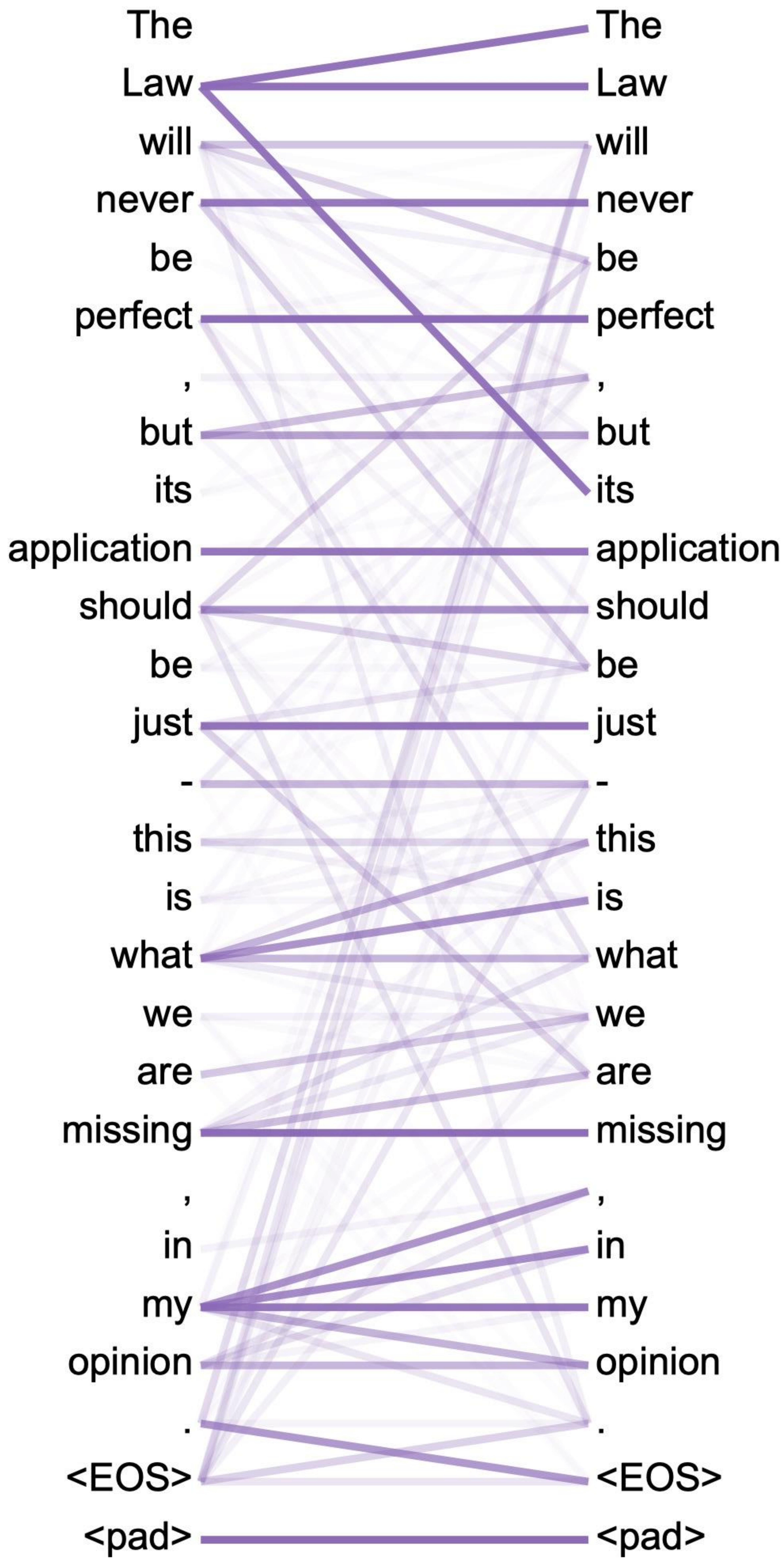
Embedding dimension

Transformer architecture: Zooming back





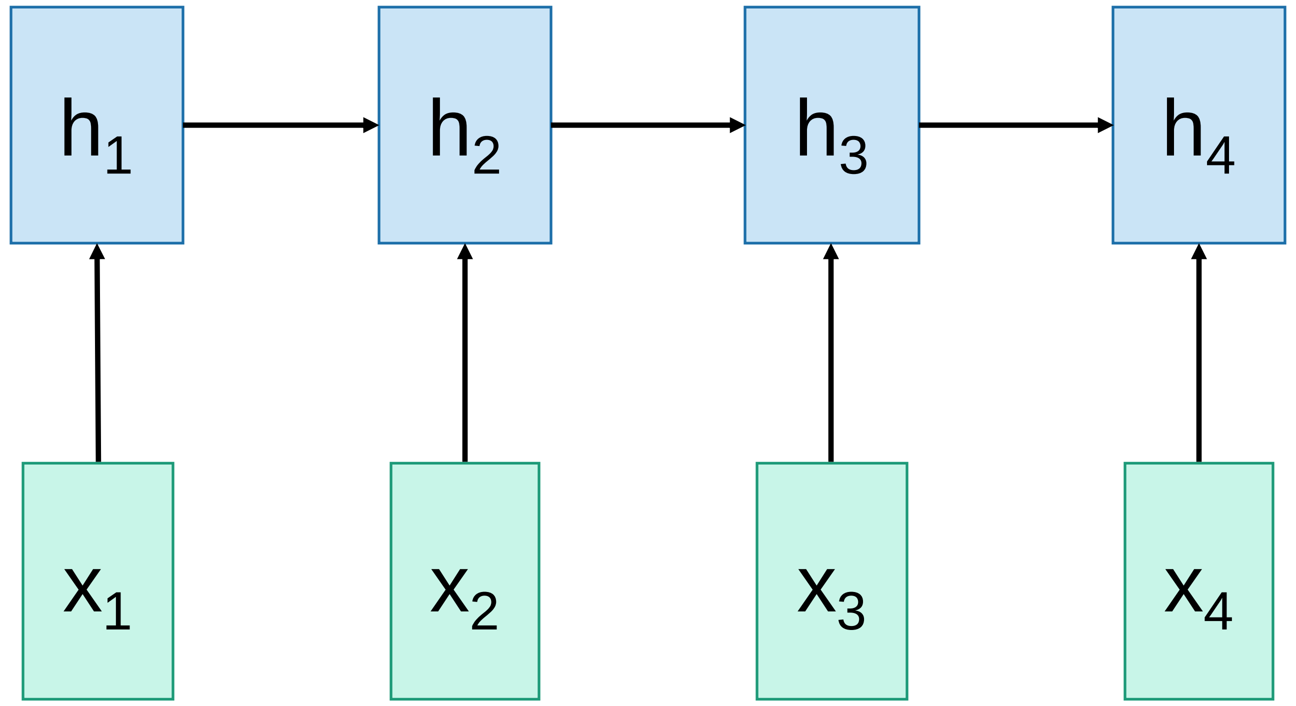
A. Vaswani et al., [Attention is all you need](), NeurIPS 2017



The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>
The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>

Different ways of processing sequences

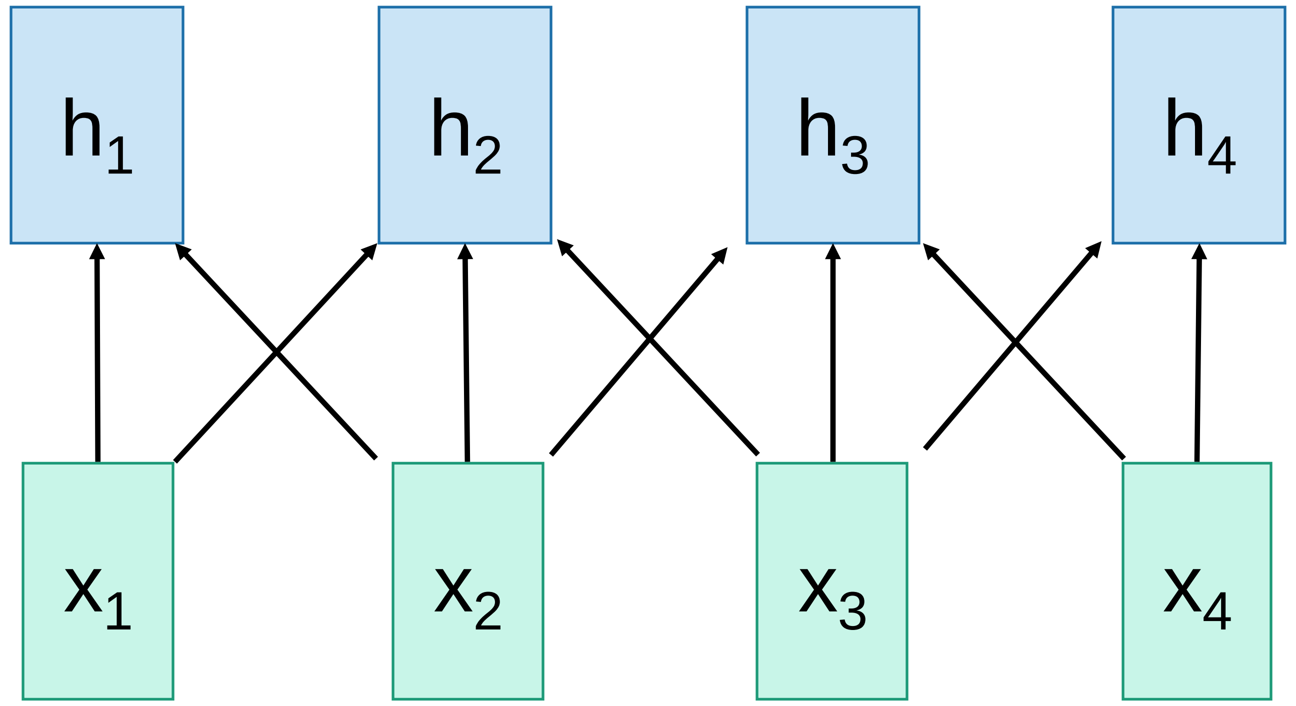
RNN



Works on **ordered sequences**

- Pros: Good at long sequences: the last hidden vector encapsulates the whole sequence
- Cons: Not parallelizable: need to compute hidden states sequentially

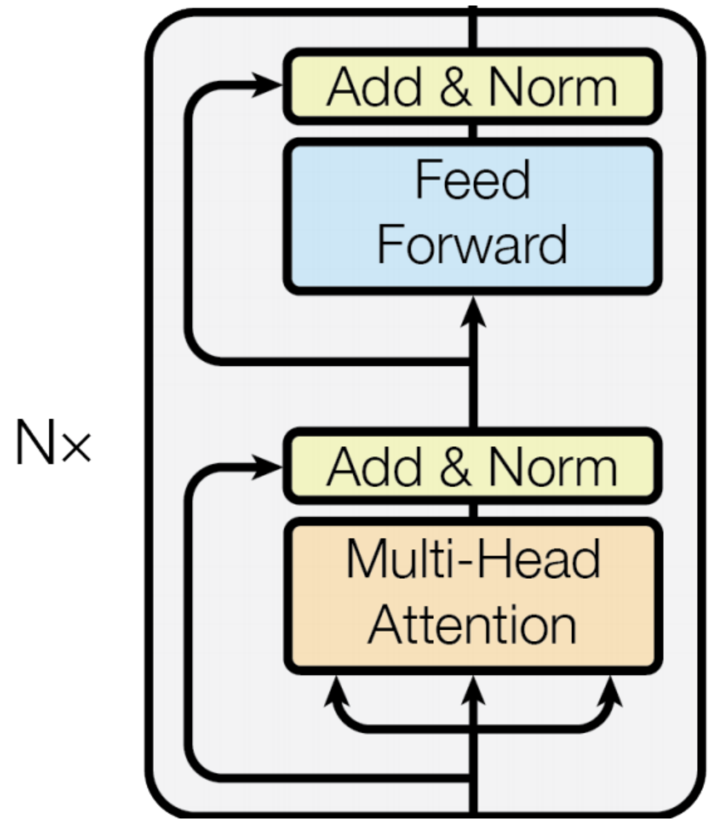
1D convolutional network



Works on **multidimensional grids**

- Con: Bad at long sequences: Need to stack many conv layers for outputs to "see" the whole sequence
- Pro: Highly parallel: Each output can be computed in parallel

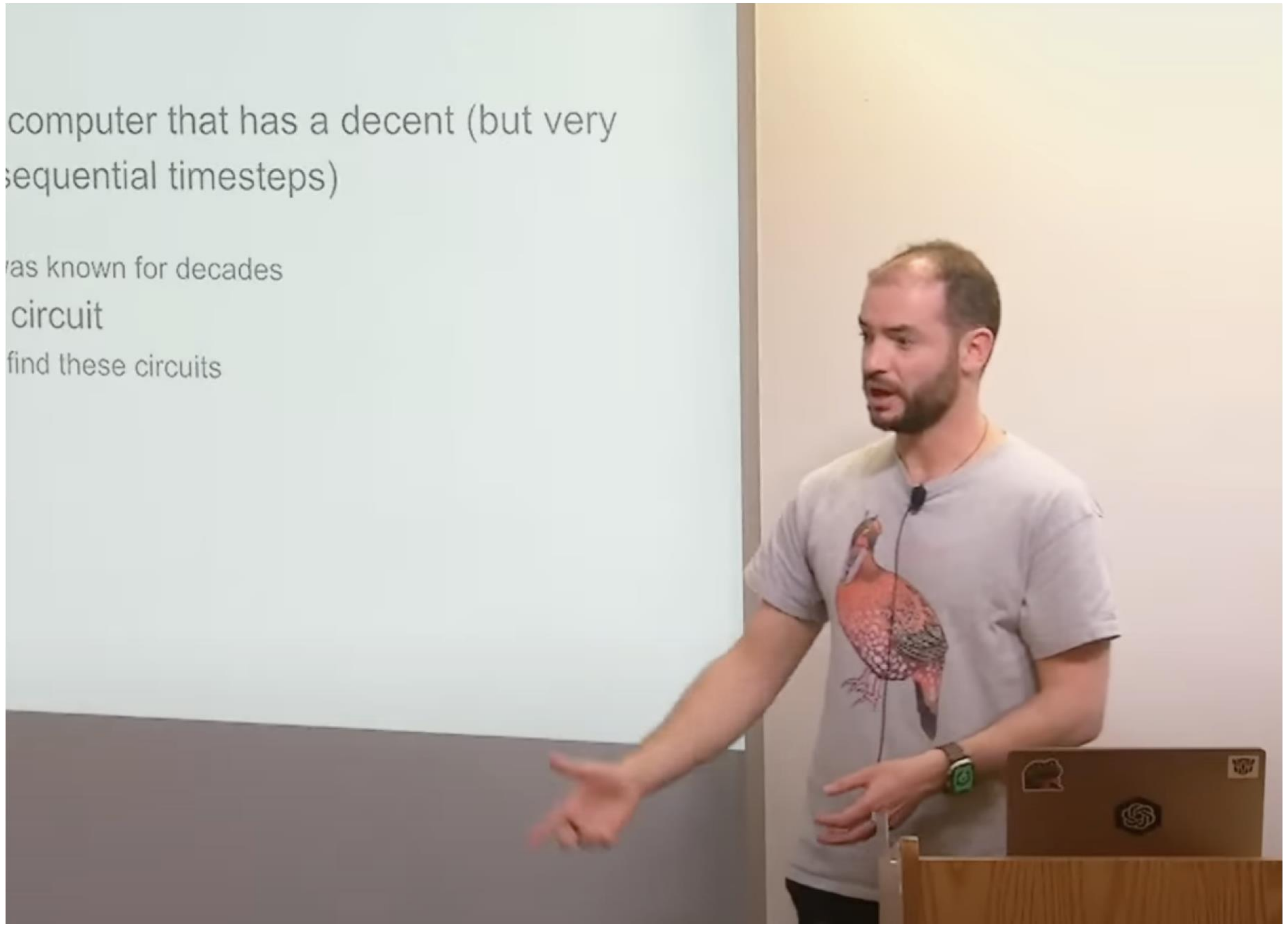
Transformer



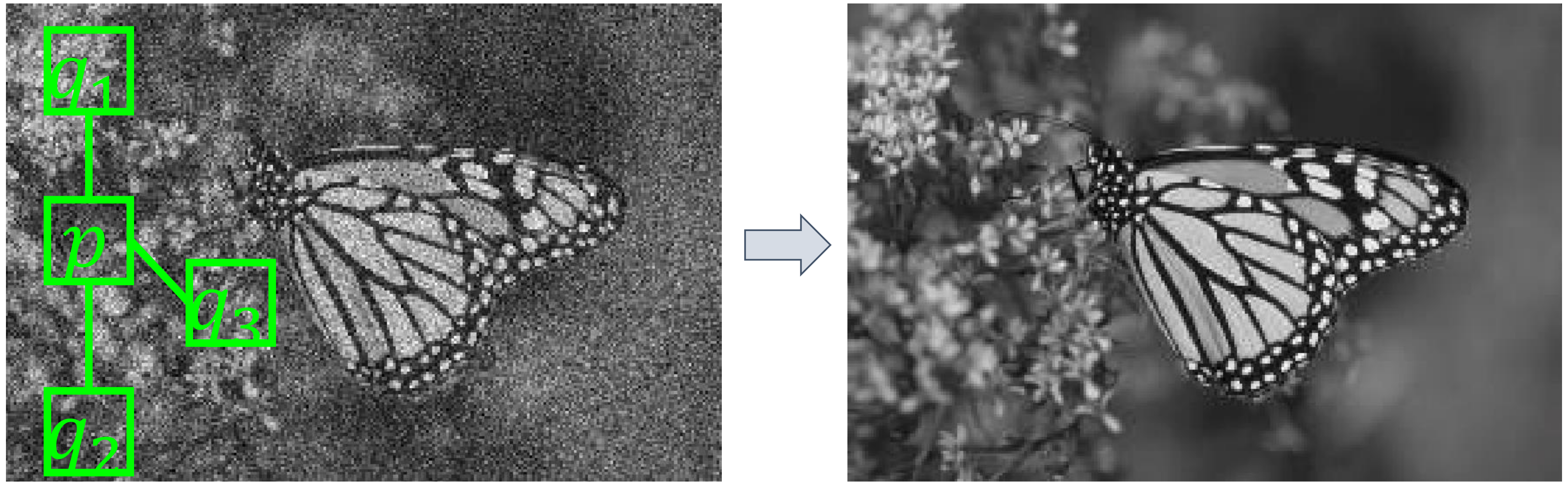
- Works on **sets of vectors**
- Pro: Good at long sequences: after one self-attention layer, each output "sees" all inputs!
- Pro: Highly parallel: Each output can be computed in parallel
- Con: Very memory-intensive

The RNN has a bottleneck, the hidden state. So it has a hard time implementing the transformer. However, had we found a way to engineer a very very large hidden state, perhaps it would become as good as a transformer again.

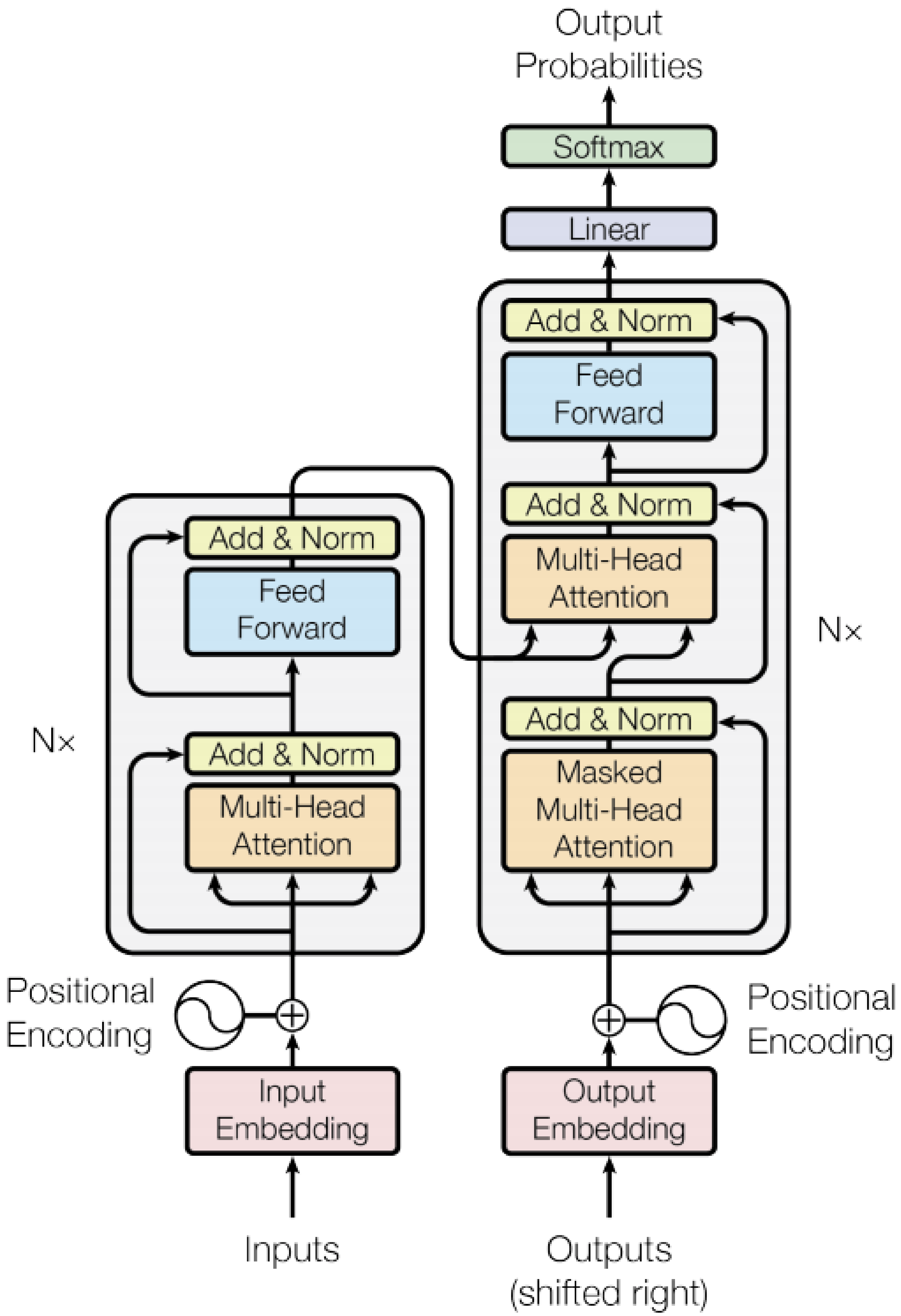
— Ilya Sutskever at Simons



It is all Non-local Means



Buades et al., 2005.



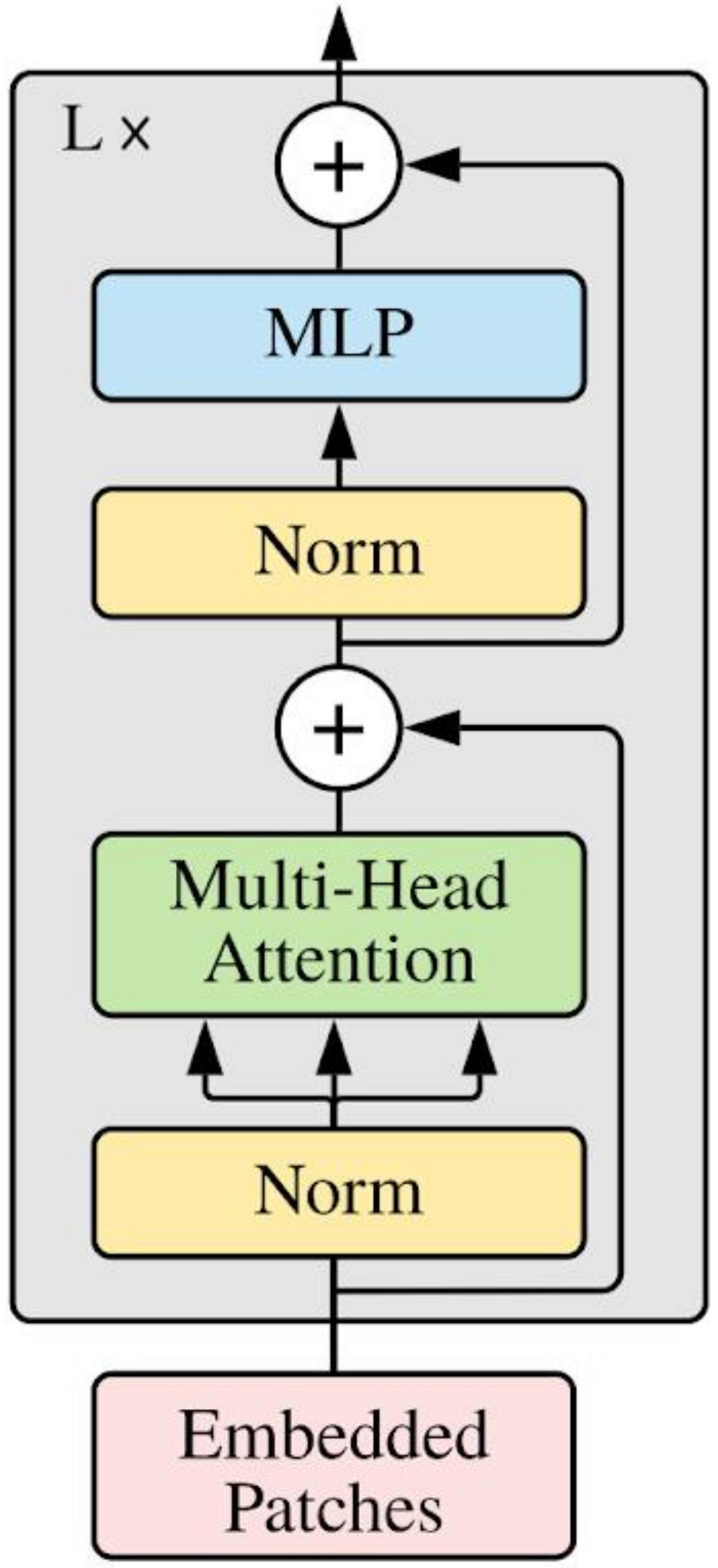
Vision Transformer

Xiaolong Wang

Vanilla ViT

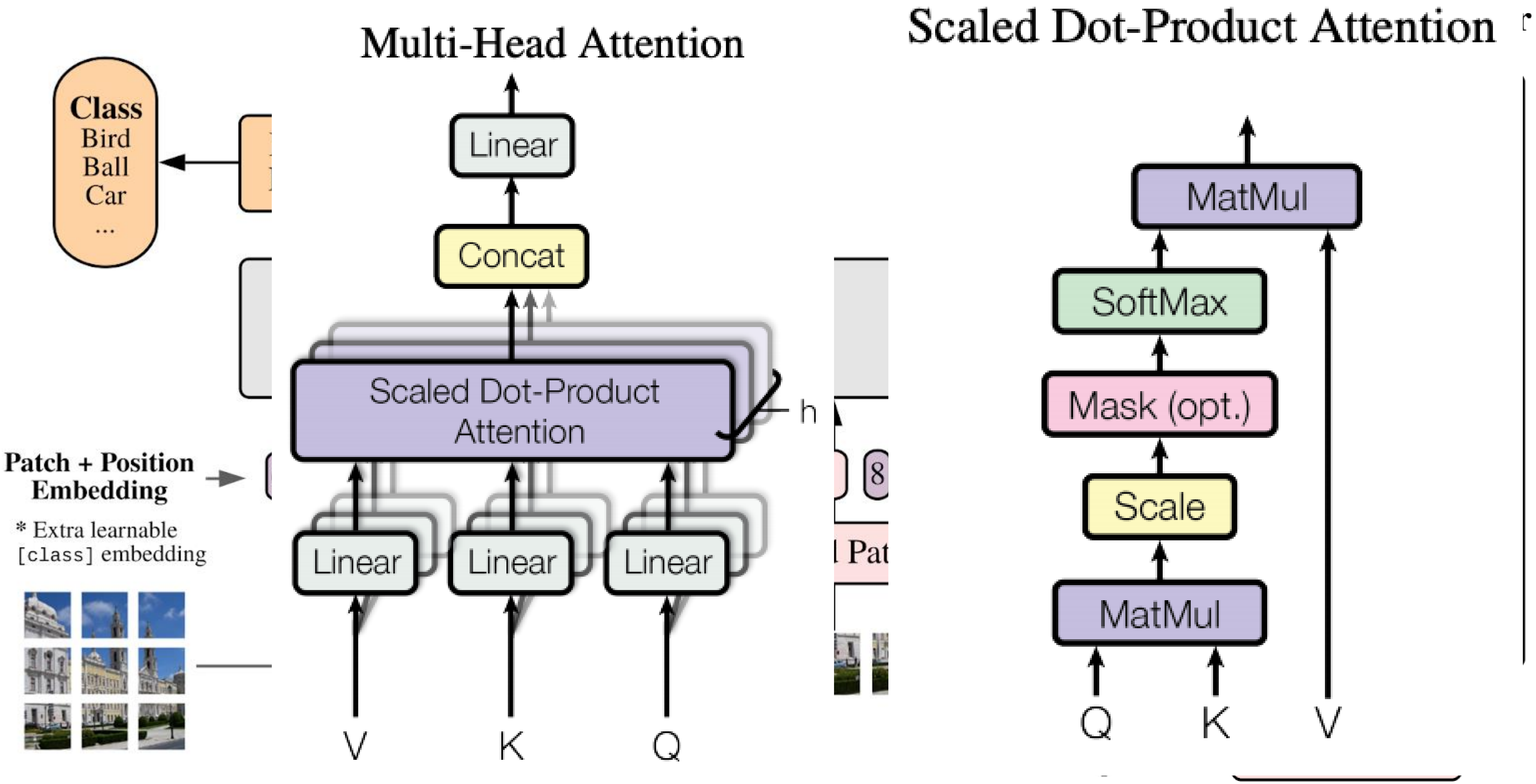
Vision Transformer (ViT)

Transformer Encoder



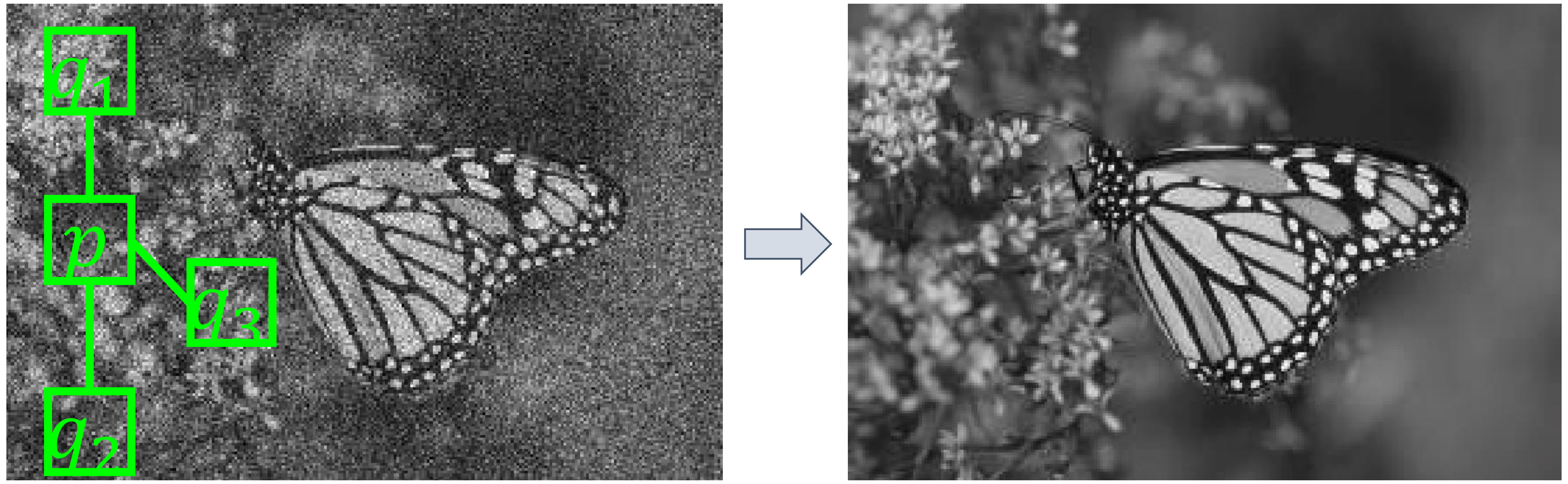
Dosovitskiy, Beyer, Kolesnikov, Weissenborn and Zhai et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale
. 2020

Vanilla ViT

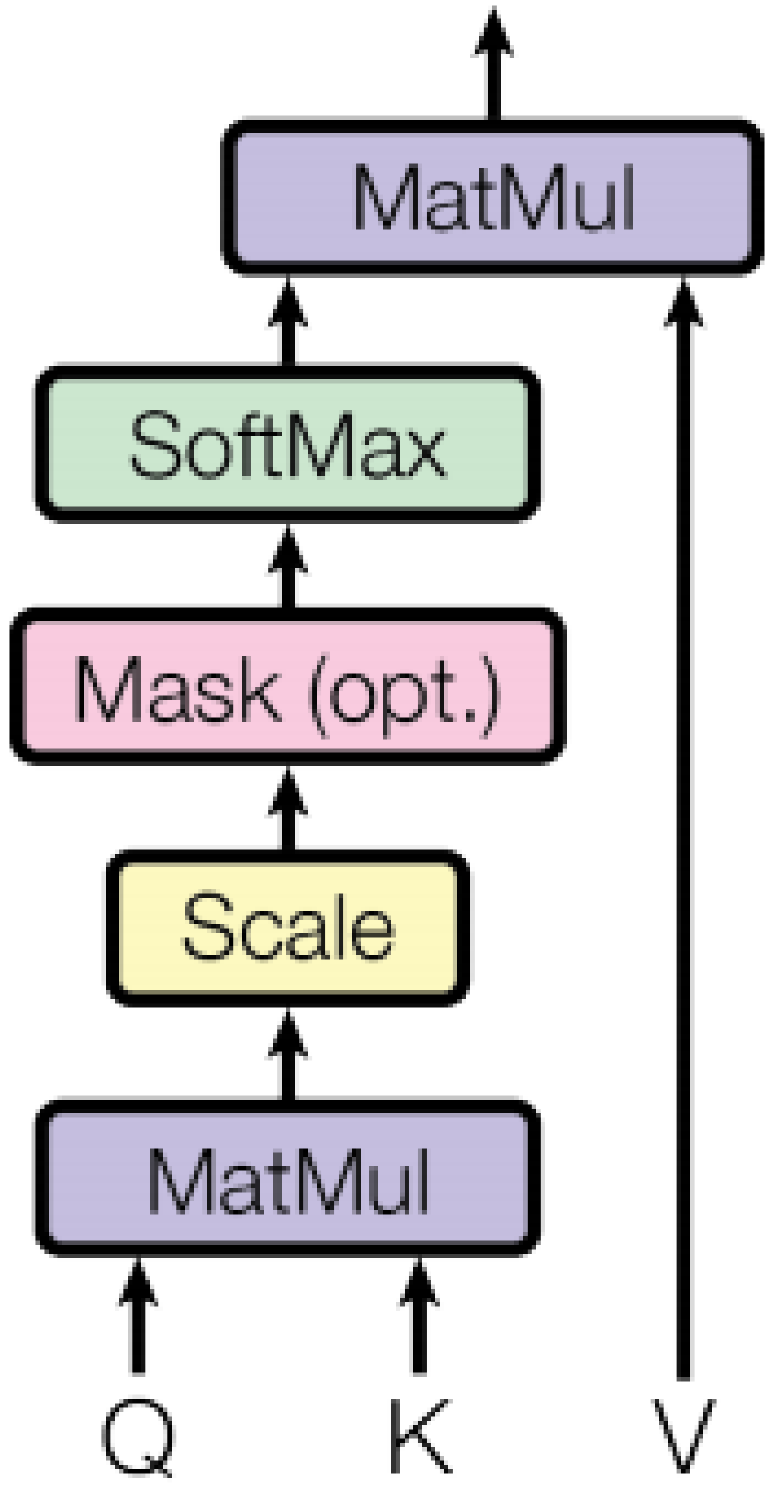


Multi-Head Attention
Scaled Dot-Product Attention
Class
Bird
Ball
Car
...
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q
Patch + Position Embedding
* Extra learnable [class] embedding
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V

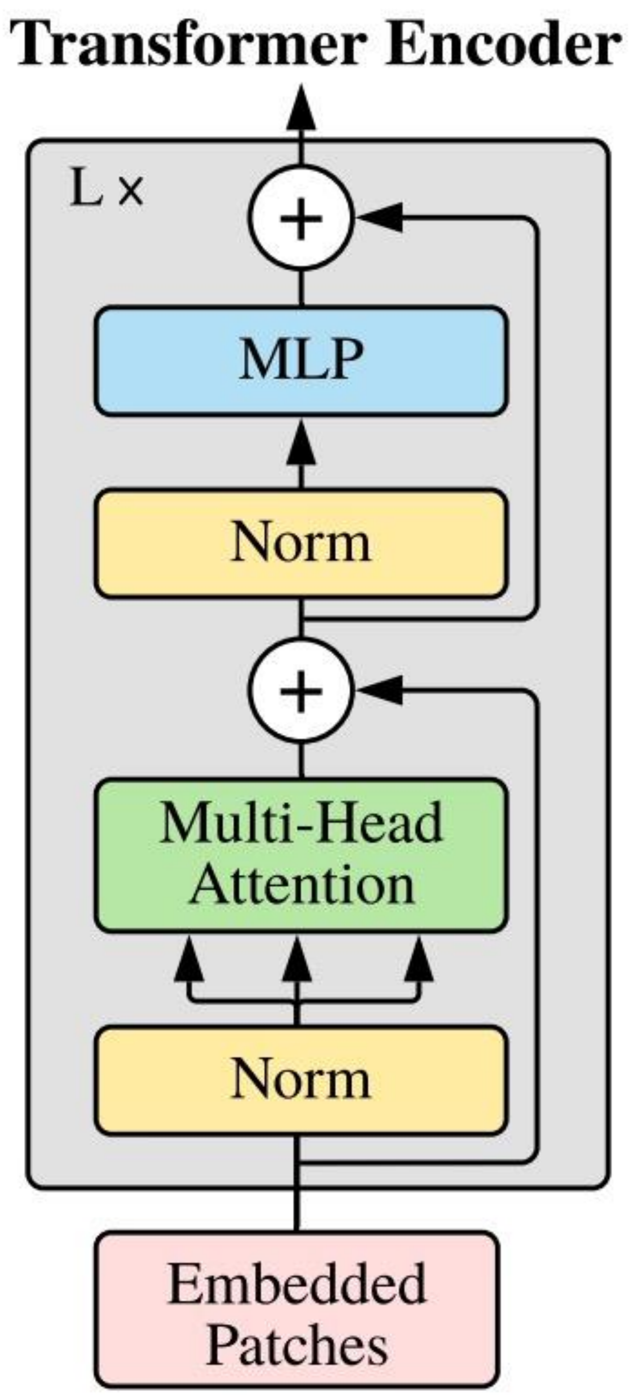
Vanilla ViT



Scaled Dot-Product Attention



Vanilla ViT



$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\, \cdots;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D},\, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D} \tag{1}$$

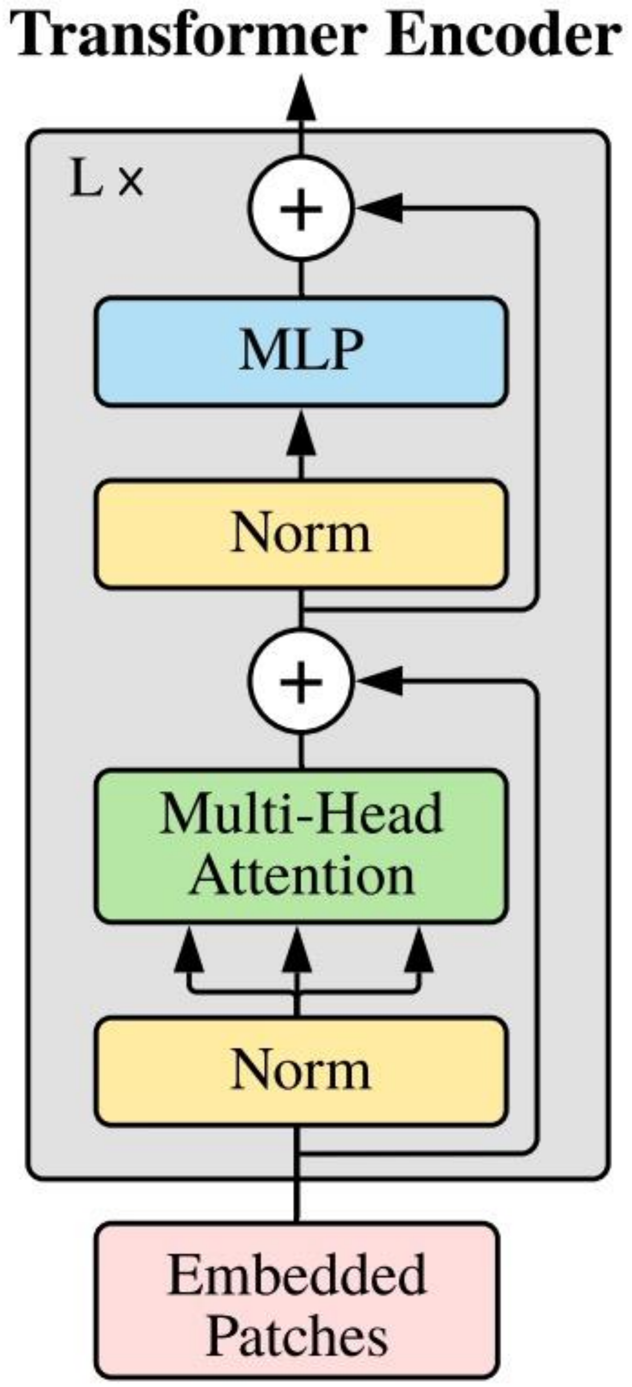
$$\mathbf{z'}_\ell = \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1\ldots L \tag{2}$$

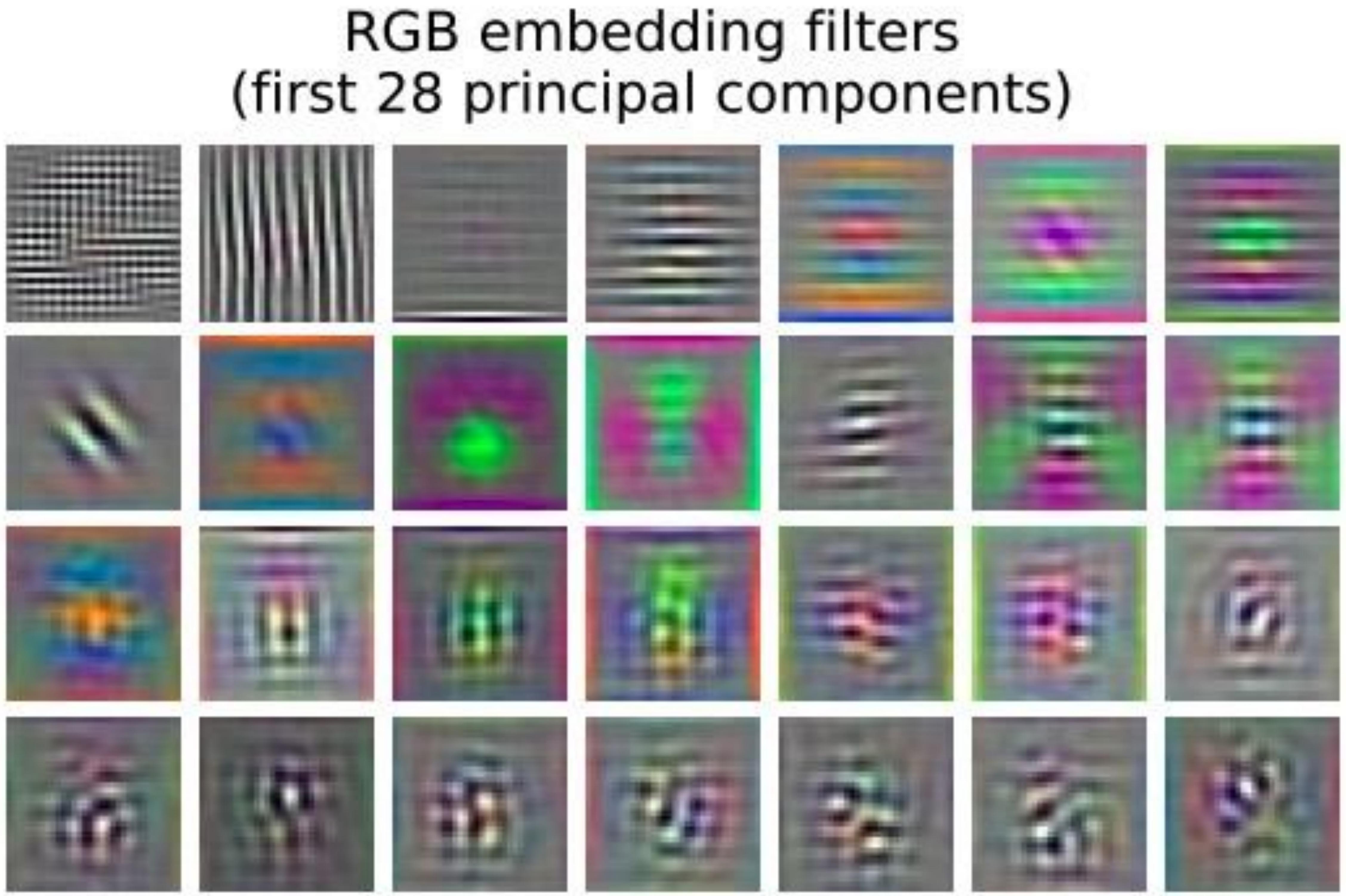
$$\mathbf{z}_\ell = \mathrm{MLP}(\mathrm{LN}(\mathbf{z'}_\ell)) + \mathbf{z'}_\ell, \qquad \ell = 1\ldots L \tag{3}$$

$$\mathbf{y} = \mathrm{LN}(\mathbf{z}_L^0) \tag{4}$$

Vanilla ViT

- ViT-L / 32 model first layer, it means the patch has 32 x 32 pixels





Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Vanilla ViT

- Previous CNN is tiny compared with ViT models
- CNN is much faster

Network	#param.	image size	throughput (image/s)	ImNet top-1
Convnets				
ResNet-18 [21]	12M	224^2	4458.4	69.8
ResNet-50 [21]	25M	224^2	1226.1	76.2
ResNet-101 [21]	45M	224^2	753.6	77.4
ResNet-152 [21]	60M	224^2	526.4	78.3
RegNetY-4GF [40]⋆	21M	224^2	1156.7	80.0
RegNetY-8GF [40]⋆	39M	224^2	591.6	81.7
RegNetY-16GF [40]⋆	84M	224^2	334.7	82.9
ViT-B/16 [15]	86M	384^2	85.9	77.9
ViT-L/16 [15]	307M	384^2	27.3	76.5
DeiT-Ti	5M	224^2	2536.5	72.2
DeiT-S	22M	224^2	940.4	79.8
DeiT-B	86M	224^2	292.3	81.8
DeiT-B↑384	86M	384^2	85.9	83.1

DeiT: Touvron and Cord et al. 2020

Vanilla ViT

- Small and Base size model could be improved with strong data augmentation and regularization (DeiT)
- More data helps ViT (ImageNet-22K, JFT-300M)

1.3M Images

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [60]	224^2	22M	4.6G	940.4	79.8
DeiT-B [60]	224^2	86M	17.5G	292.3	81.8
DeiT-B [60]	384^2	86M	55.4G	85.9	83.1

(b) ImageNet-22K pre-trained models 14M Images

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [19]	384^2	307M	190.7G	27.3	85.2

(c) JFT-300M pre-trained models 300M Images

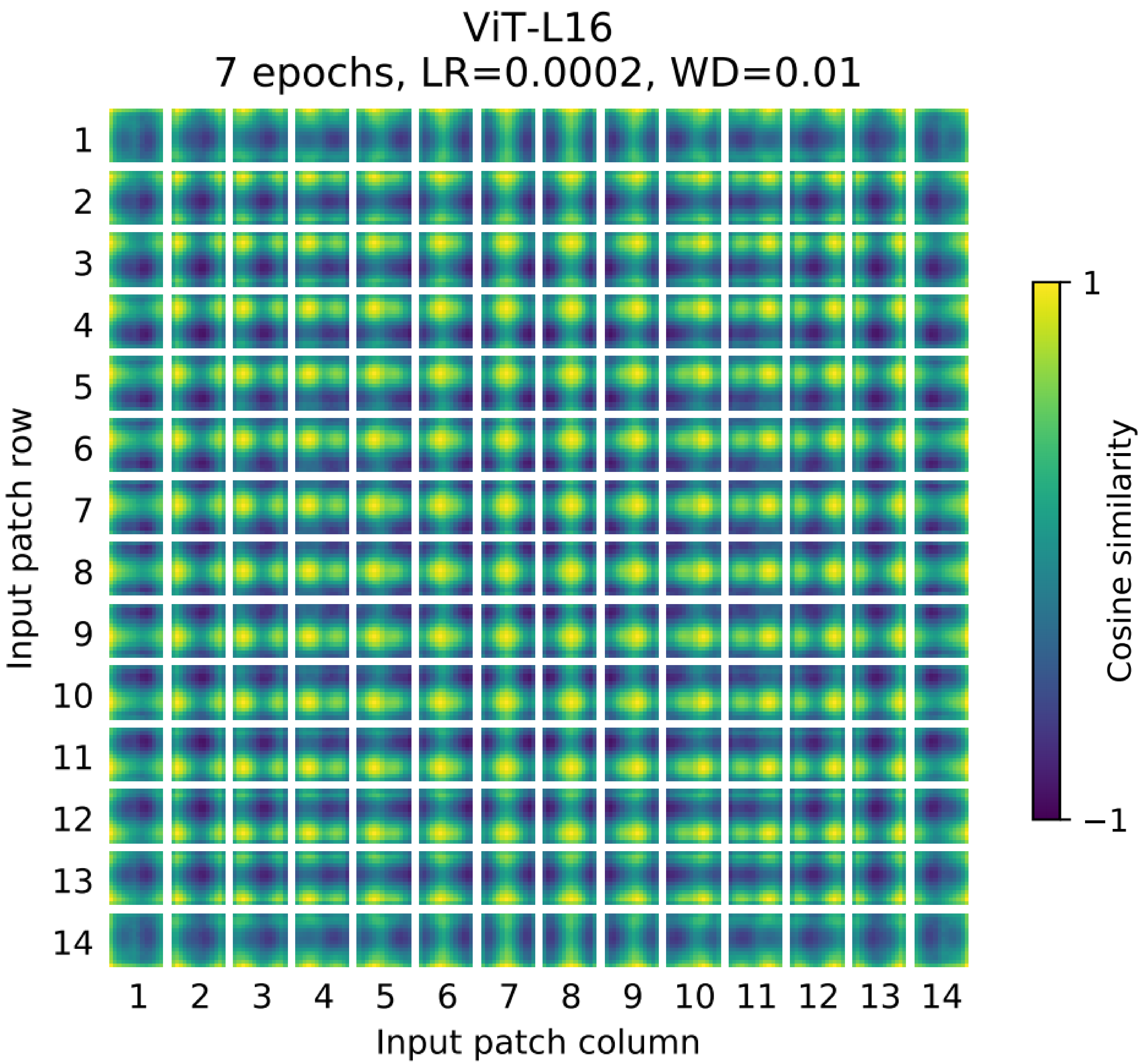
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.2
ViT-L/16 [19]	384^2	307M	190.7G	27.3	87.1
ViT-H/14 [19]	384^2	307M	190.7G	27.3	88.1

DeiT: Touvron and Cord et al. 2020

Vanilla ViT

- ViT learns intuitive local structures
- Closer patches tend to have more similarity of position embeddings

Each patch shows the position embedding cosine similarity between the patch at this location and all other patches

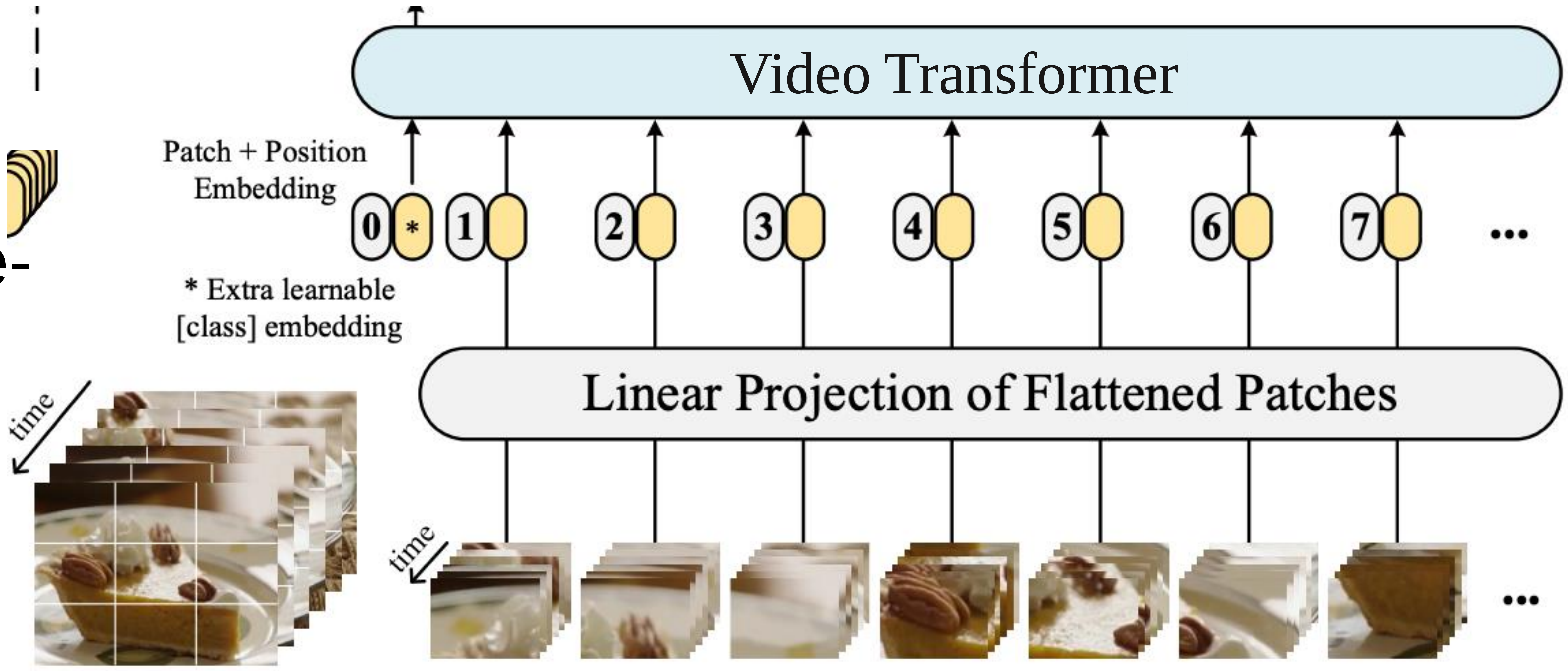


ViT: Dosovitskiy, Beyer, Kolesnikov, Weissenborn and Zhai et al. 2020

Transformer in Videos

TimeSformer

- Share similar structure as ViT
- Video clip as a sequence of frame-level patches (16x16)



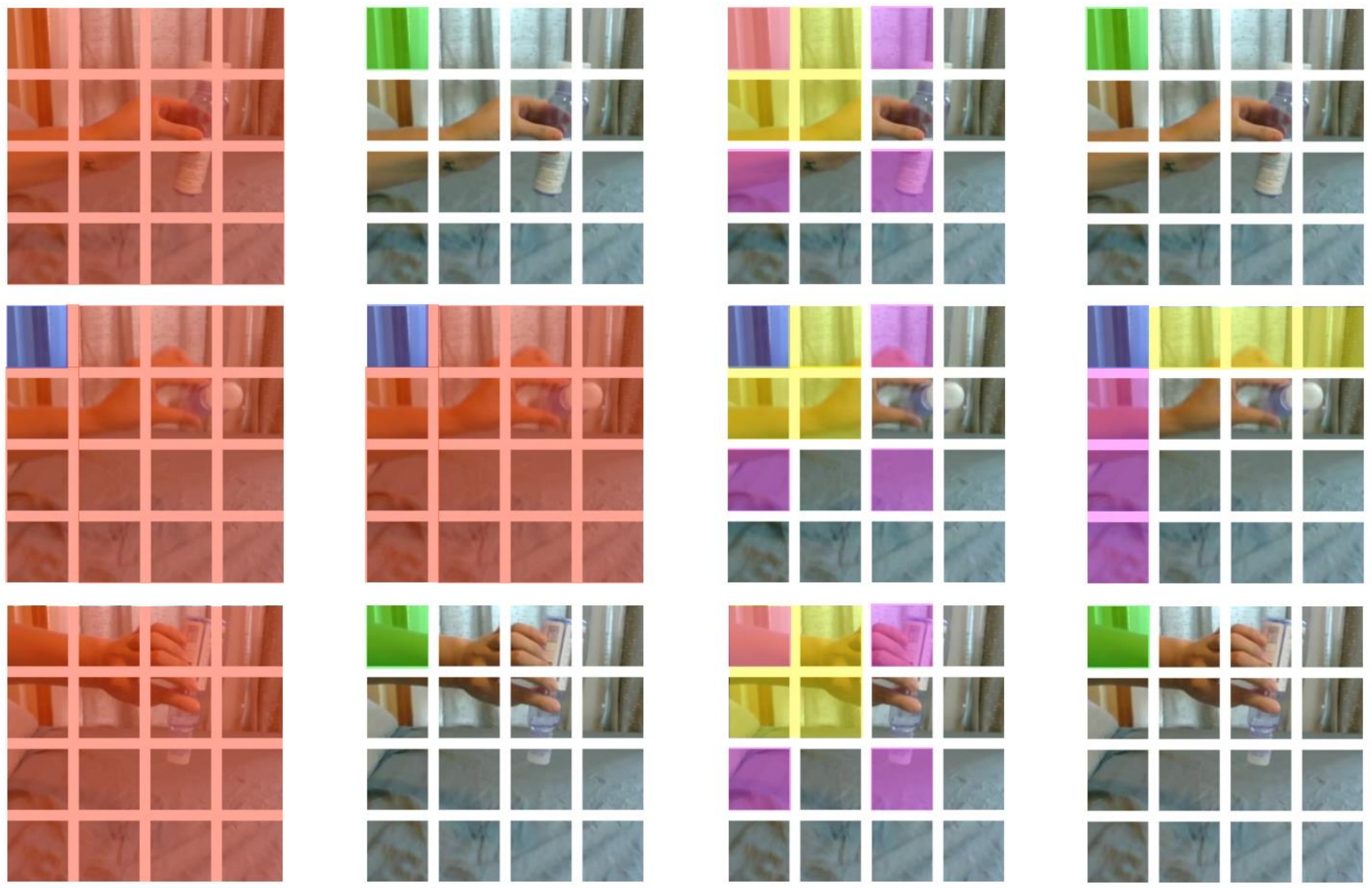
TimeSformer: Bertasius et al. 2021

Transformer in Videos

TimeSformer

- How to model temporal information?

Attention	Params	K400	SSv2
Space	85.9M	76.9	36.6
Joint Space-Time	85.9M	77.4	58.5
Divided Space-Time	121.4M	**78.0**	**59.5**
Sparse Local Global	121.4M	75.9	56.3
Axial	156.8M	73.5	56.2



Space Attention (S)

Joint Space-Time Attention (ST)

Divided Space-Time Attention (T+S)

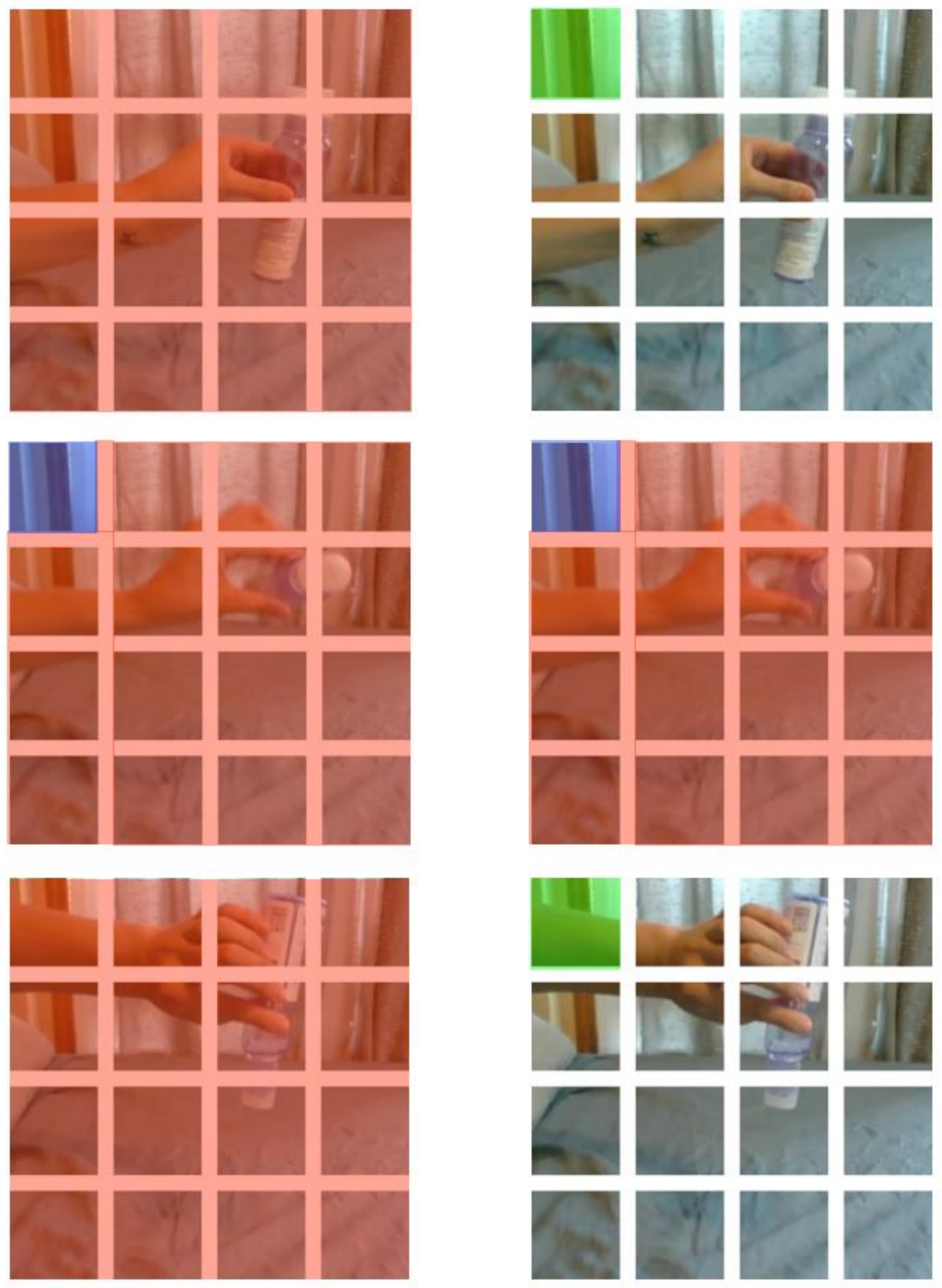
Sparse Local Global Attention (L+G)

Axial Attention (T+W+H)

TimeSformer: Bertasius et al. 2021

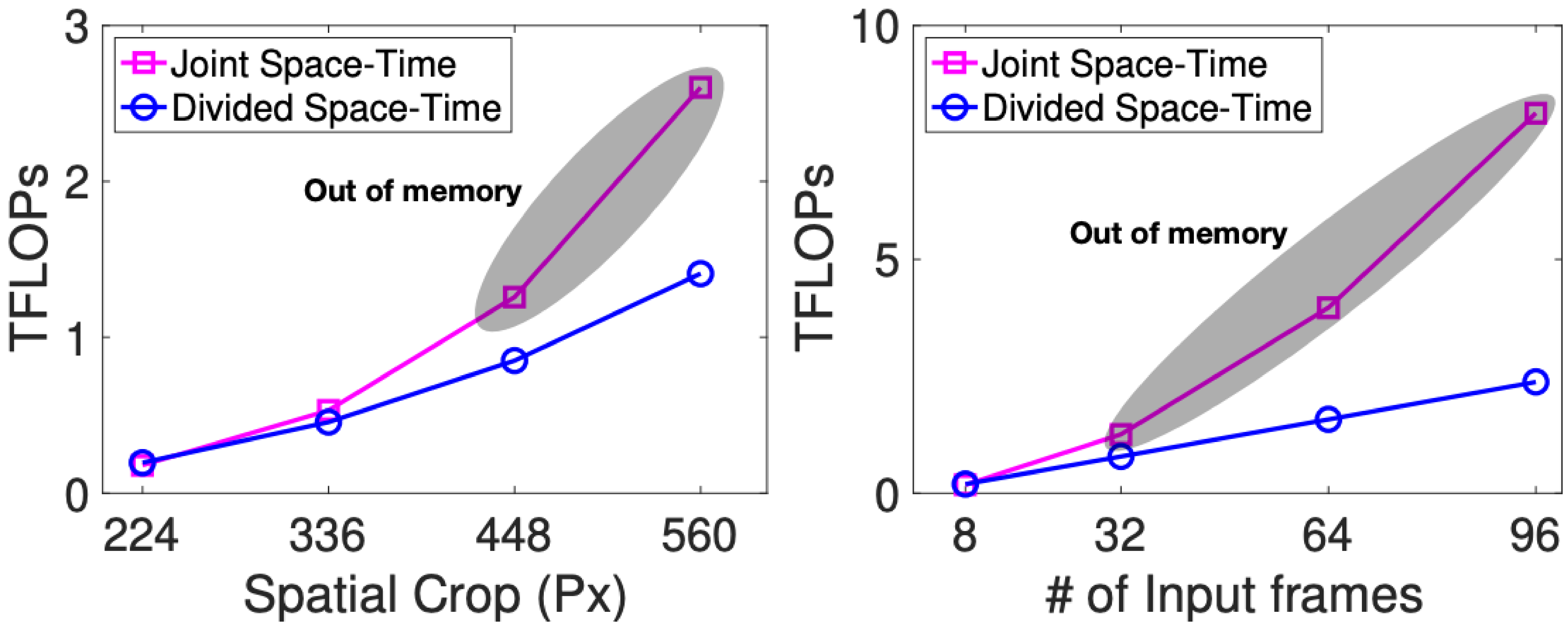
Transformer in Videos

TimeSformer



Joint Space-Time Attention (ST)

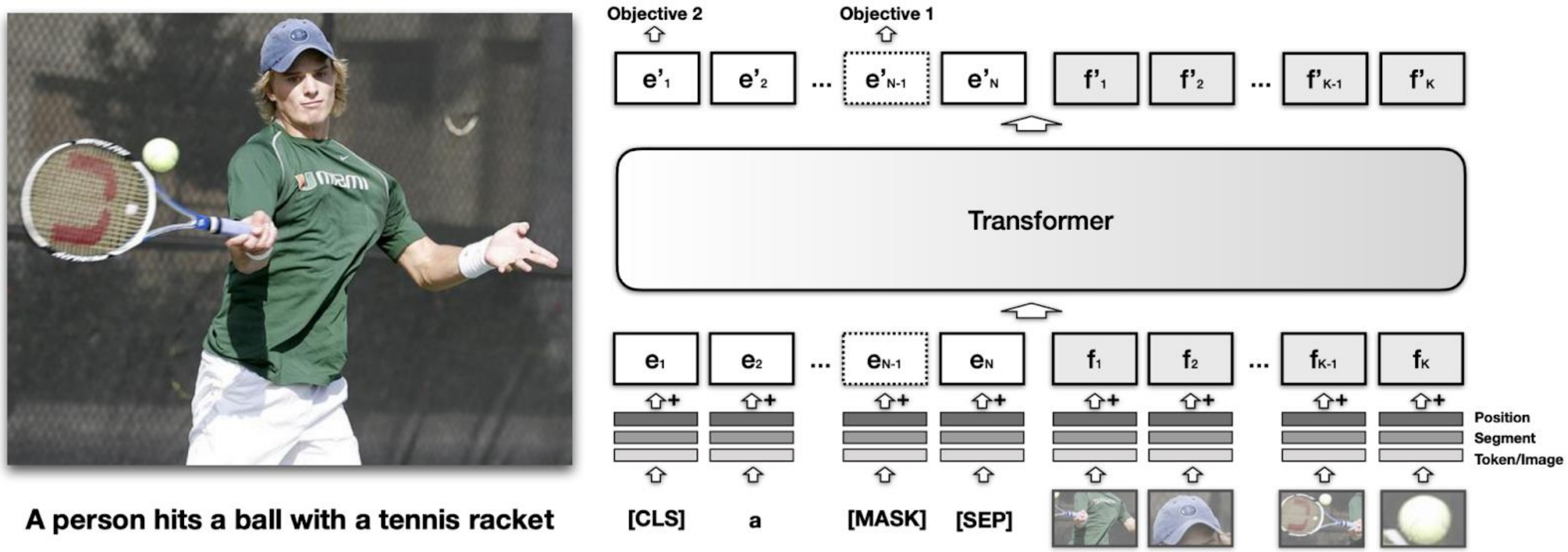
Divided Space-Time Attention (T+S)



Transformer for Multi-modality

Image-Text

- Image patches are generated by pre-trained object detector
- Mask language modeling

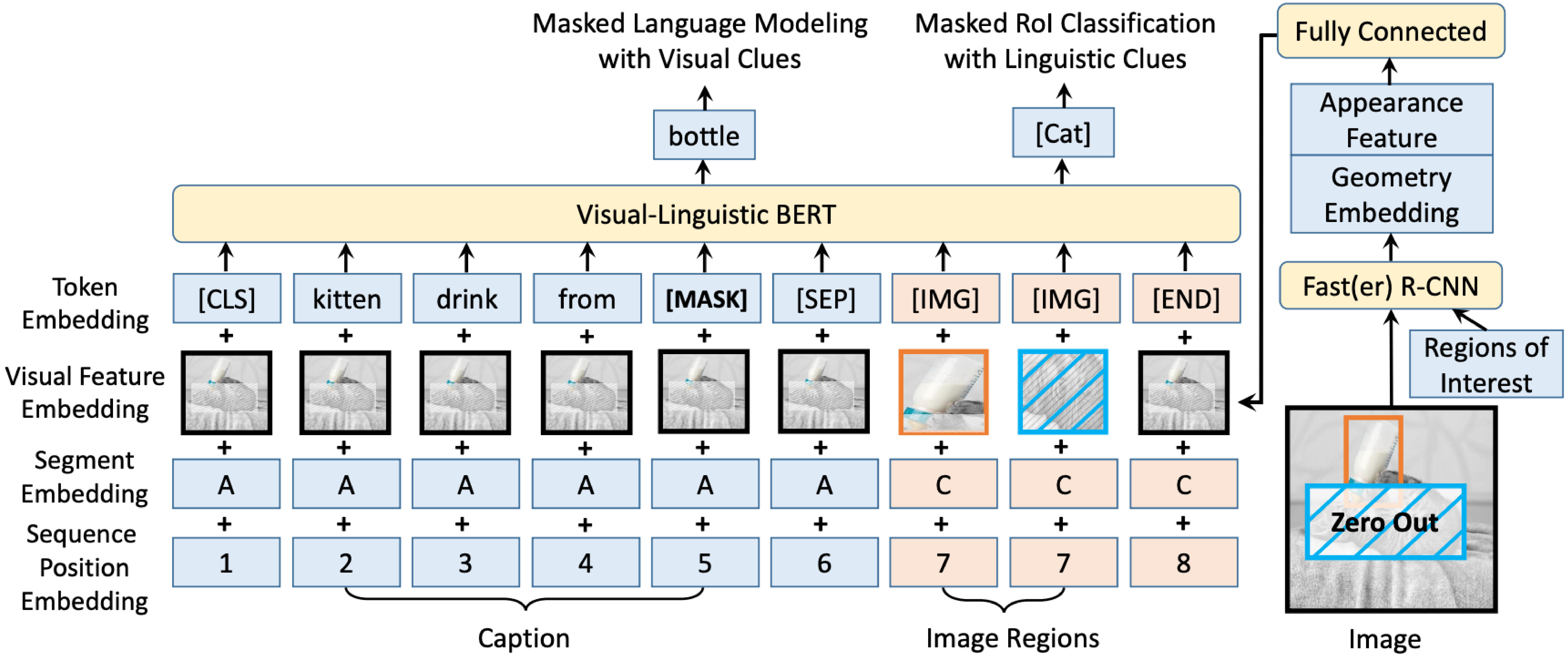


Visual-BERT: Li et al. 2019

Transformer for Multi-modality

Image-Text

- Extend with masked RoI classification

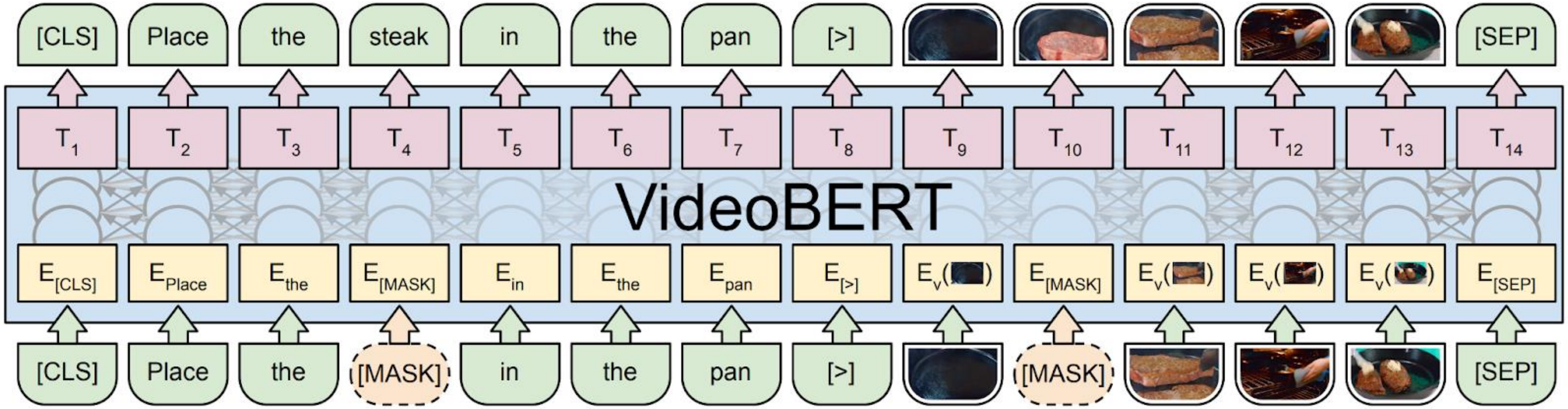


VL-BERT: Su et al. 2019

Transformer for Multi-modality

Video-Text

- Concatenate word sequence with video frame sequence
- Mask language modeling



VideoBERT: Sun et al. 2019