دانشکده برق و کامپیوتر

روش‌های تعبیه‌سازی کلمات و نقش مکانیزم توجه در شبکه‌های عمیق

دکتر محمدجواد فدائی‌اسلام

۲۴ آذر ۱۴۰۰

فهرست مطالب

- مبدل‌ها و مکانیزم خود توجه
  - تاریخچه
  - مدل
- BERT و کاربردهای آن
- مکانیزم خود توجه در پردازش تصویر
  - ViT
  - TTR
  - DETR
  - TNT

مقدمه – نوآوری‌های شاخص در زمینه پردازش متن

- مبدل‌ها توسط **واسوانی** و همکارانش در سال ۲۰۱۷ ابداع شدند و اولین بار در زمینه پردازش زبان طبیعی مورد استفاده قرار گرفتند.

A. Vaswani et el. “**Attention is all you need**”, Annual Conference on Neural Information Processing Systems (NeurIPS), 30:5998-6008, 2017.

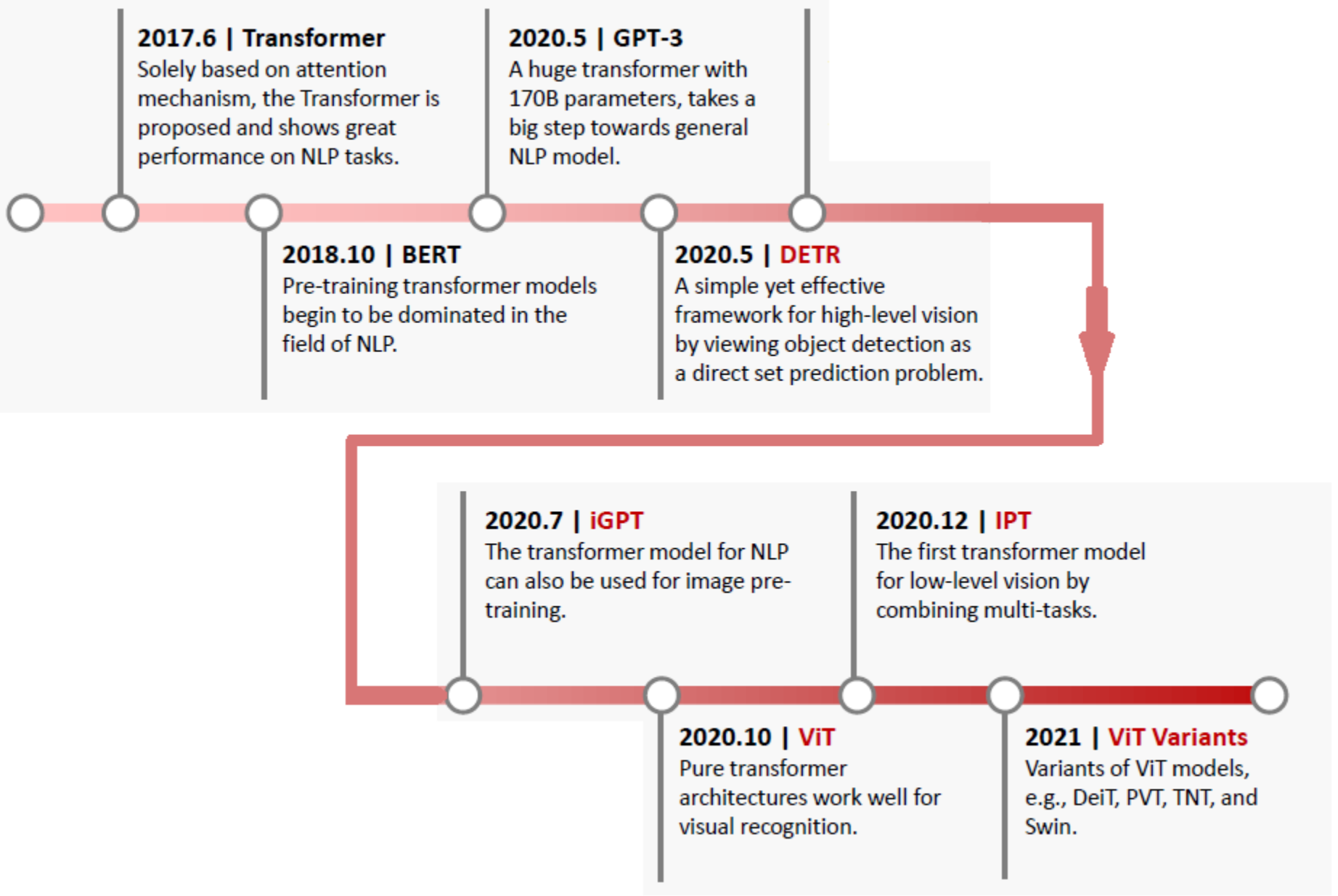
- دِولین و همکاران در سال ۲۰۱۹ مدل بازنمایی زبانی جدیدی به نام BERT ابداع نمودند. این مدل در ۱۱ زمینه پردازش زبان طبیعی بهترین نتایج را ارایه نمود.
- بروان در سال ۲۰۲۰ مدل GPT-3 را بر روی ۴۵ ترابایت متن با ۱۷۵ میلیارد پارامتر پیش‌آموزش داد و کارایی بالایی در کارهای سطح پایین زبان طبیعی بدون نیاز به تنظیم دقیق (fine-tune) به وجود آورد.

مقدمه – نوآوری‌های شاخص در زمینه پردازش تصویر

- شبکه‌های کانولوشنی در کاربردهای پردازش تصویر به عنوان یک مولفه اساسی در نظر گرفته می‌شوند، اما امروزه مبدل‌ها دارند نشان می‌دهند که جایگزینی بالقوه برای شبکه‌های کانولوشنی هستند.
- چن و همکاران یک مبدل دنباله‌ای آموزش دادند و به نتایجی قابل مقایسه با شبکه‌های کانولوشنی در کار کلاس‌بندی تصویر رسیدند.
- مدل ViT، اولین کدگذار مبدلی آموزش داده شده بر روی مجموعه ImageNet است که نتایج بسیار خوبی در مقایسه با مدل‌های کانولوشنی به دست آورده است.
- کاربردهای دیگر مبدل‌ها در پردازش تصویر

- Object detection
- Semantic Segmentation
- Image Processing
- Video Understanding

KEY MILESTONES IN THE DEVELOPMENT OF TRANSFORMER



Representative works of vision transformers

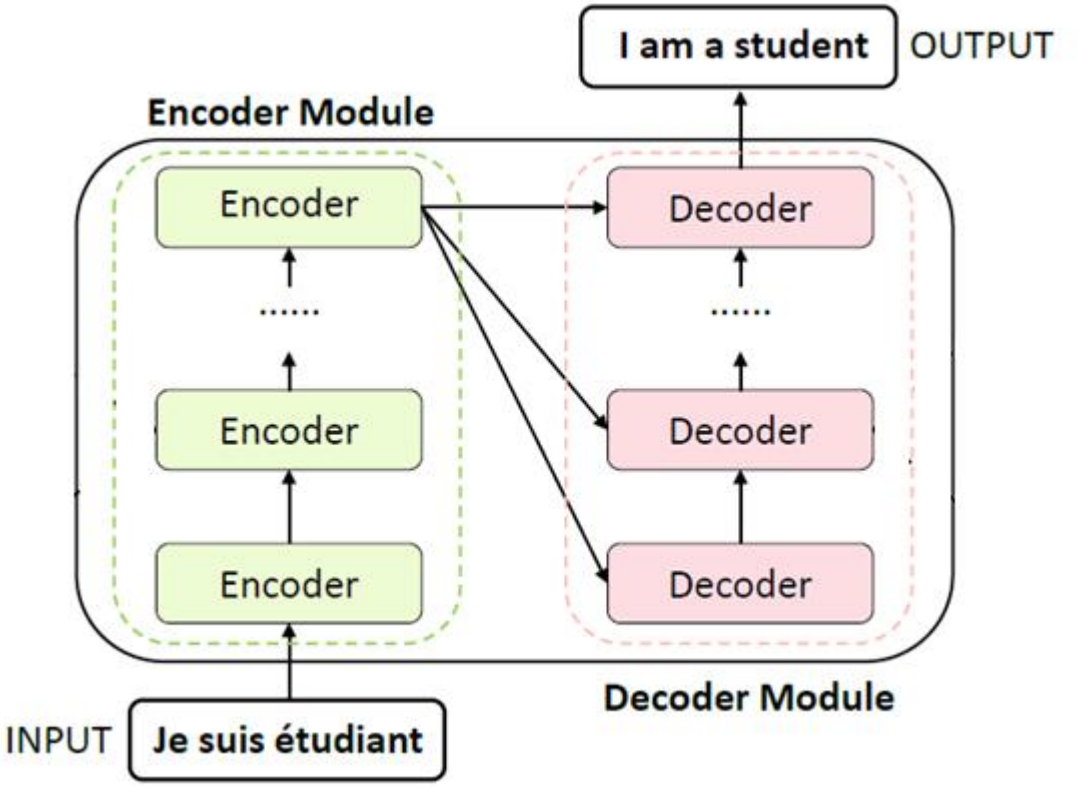
Backbone	Supervised pretraining	ViT [55]	ICLR 2021
		DeiT [219]	ICML 2021
		Swin [17]	ICCV 2021
	Self-supervised pretraining	iGPT [29]	ICML 2020
		MoCo v3 [32]	ICCV 2021
High/Mid-level vision	Object detection	DETR [19]	ECCV 2020
		Deformable DETR [291]	ICLR 2021
		ACT [284]	arXiv 2020
		UP-DETR [49]	CVPR 2021
		TSP [210]	arXiv 2020
	Segmentation	Max-DeepLab [228]	CVPR 2021
		VisTR [235]	CVPR 2021
		SETR [285]	CVPR 2021
	Pose Estimation	Hand-Transformer [102]	ECCV 2020
		HOT-Net [103]	MM 2020
		METRO [138]	CVPR 2021
Low-level vision	Image generation	Image Transformer [171]	ICML 2018
		Taming transformer [58]	CVPR 2021
		TransGAN [111]	arXiv 2021
	Image enhancement	IPT [27]	CVPR 2021
		TTSR [251]	CVPR 2020

Representative works of vision transformers (2)

Video processing	Video inpainting	STTN [268]	ECCV 2020
	Video captioning	Masked Transformer [288]	CVPR 2018
Multimodality	Classification	CLIP [180]	arXiv 2021
	Image generation	DALL-E [185] Cogview [51]	ICML 2021 arXiv 2021
	Multi-task	UniT [100]	arXiv 2021
Efficient transformer	Decomposition	ASH [159]	NeurIPS 2019
	Distillation	TinyBert [113]	EMNLP Findings 2020
	Quantization	FullyQT [176]	EMNLP Findings 2020
	Architecture design	ConvBert [112]	NeurIPS 2020

ساختار مبدل پایه

(نگاه سطح بالا)



- از نگاه سطح بالا و در کاربرد ترجمه، یک جمله را از یک زبان دریافت می‌کند و به یک زبان دیگر ترجمه می‌نماید.
- مبدل دارای یک ماژول کدگذار و یک ماژول گدگشا است.
- ماژول کدگذار از ۶ کدگذار تشکیل یافته است و ماژول کدگشا هم از همان تعداد کدگشا مانند شکل استفاده می‌کند.
- تعداد کدگذارها (عدد ۶) در کاربردهای دیگر می‌تواند متفاوت باشد.

ساختار مبدل پایه

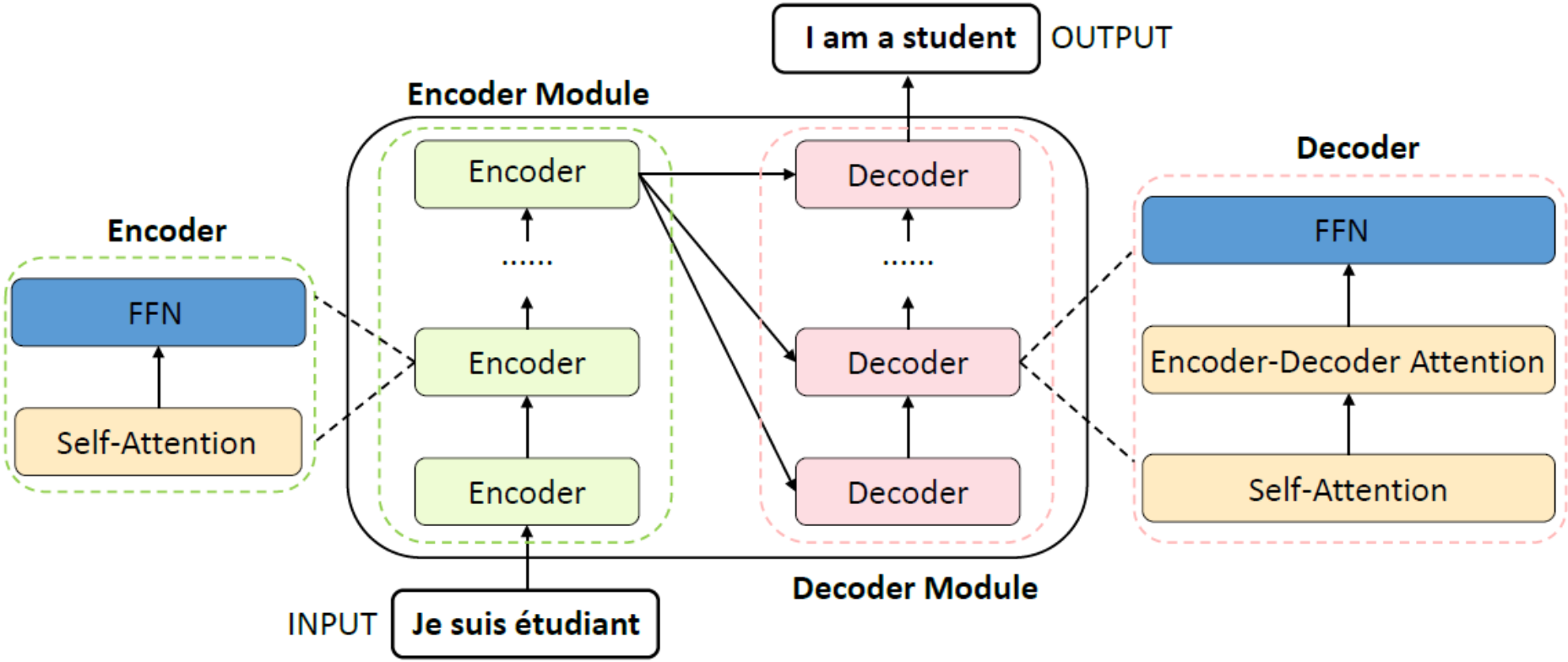
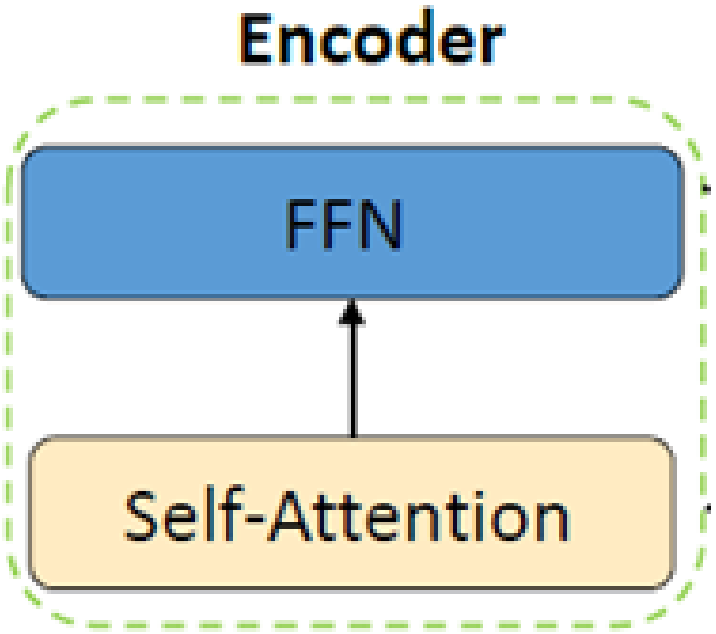


Fig. 2: Pipeline of vanilla transformer.

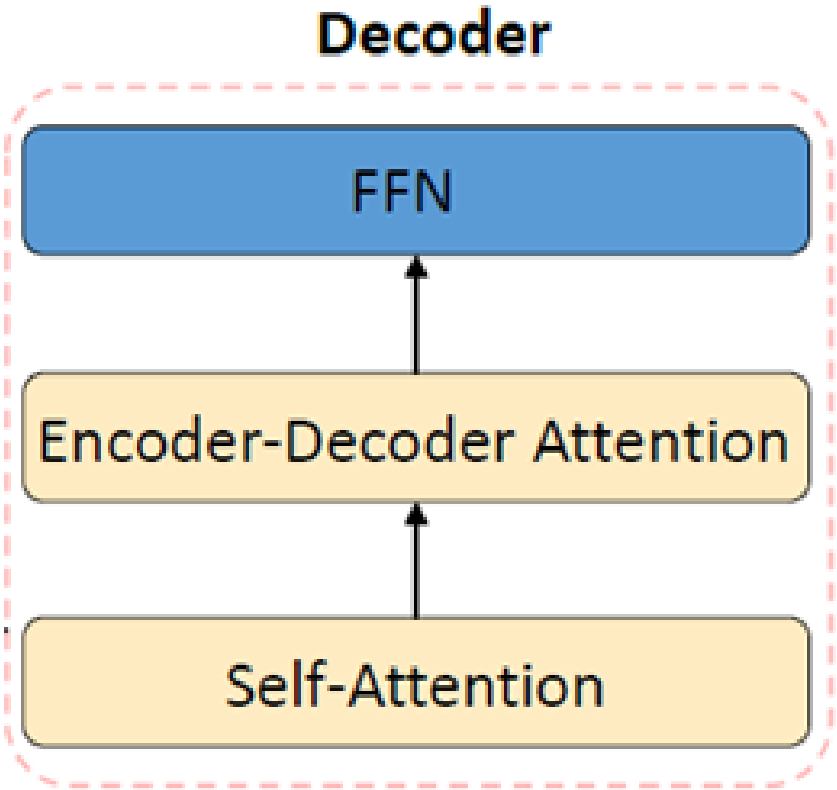
ساختار کدگذار



- کدگذارها مشابه هستند و از دو زیرلایه تشکیل شده‌اند:

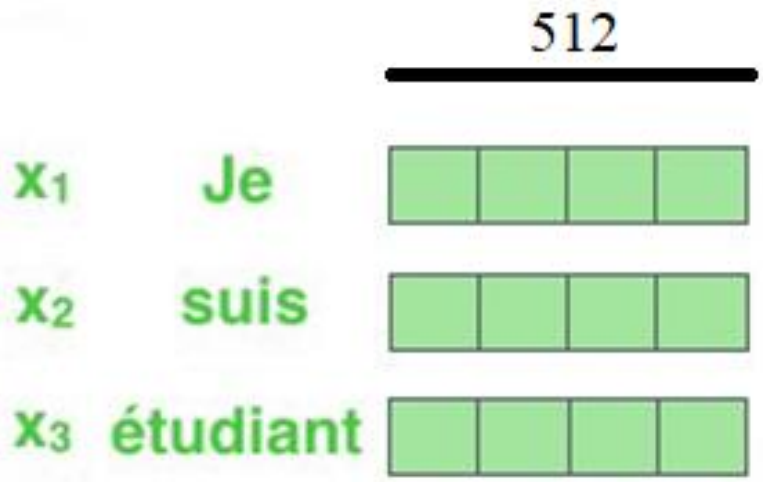
- Self-Attention
- Feed-forward Neural Network

ساختار کدگشا



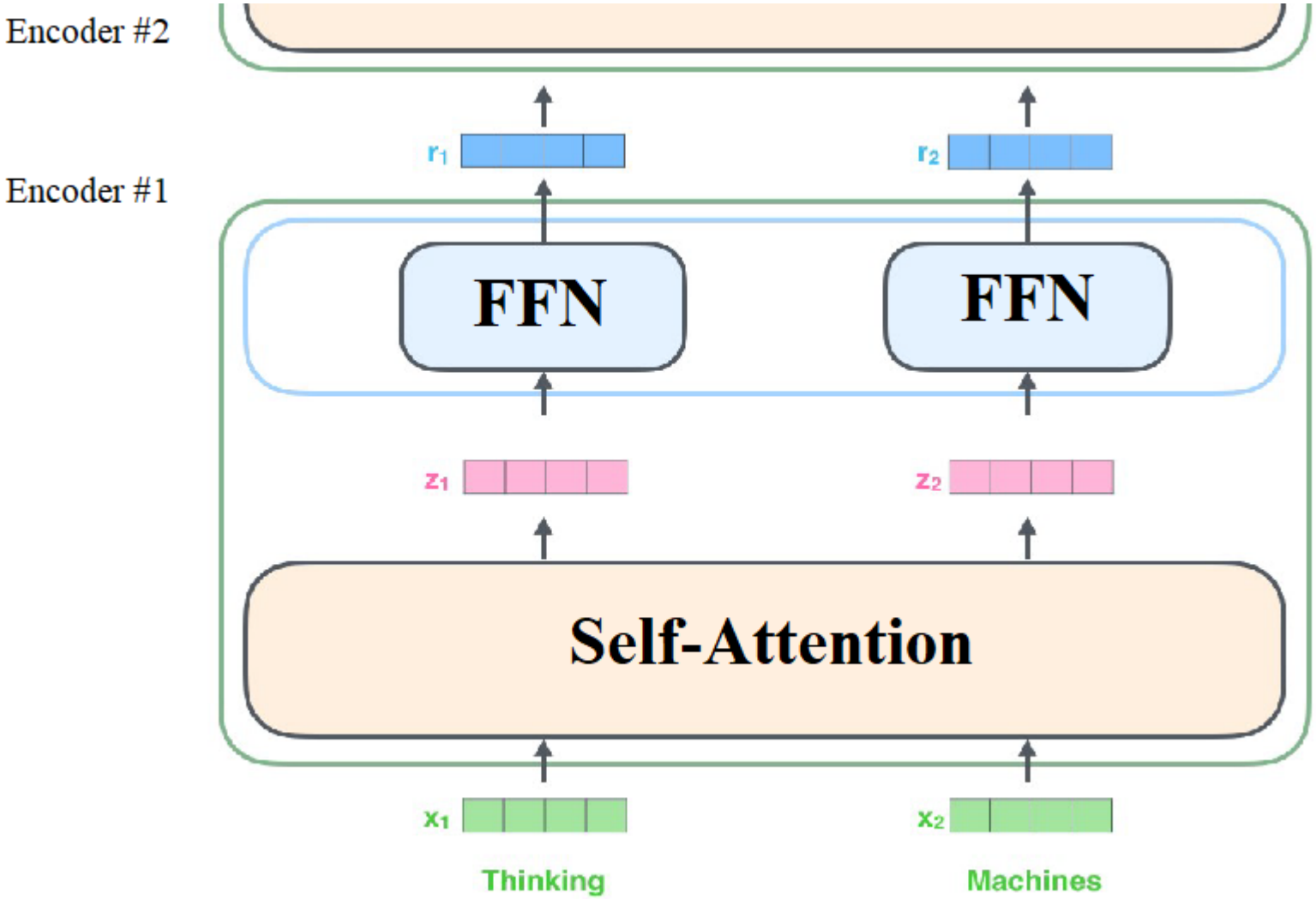
- کدگشاها مشابه هم هستند و یک لایه توجه از کدگذار بیشتر دارند:
  - Self-Attention
  - Encoder-Decoder Attention
  - Feed-forward Neural Network

تنسورها و نحوه انتقال اطلاعات



- در فرآیند ترجمه هر کلمه به صورت بردار ۵۱۲ تایی تعبیه می‌شود.
- ورودی هر کدگذار یک لیست است. کدگذار اول (پایین‌ترین کدگذار) لیستی از کلمات می‌گیرد. ورودی کدگذارهای بعدی، خروجی کدگذار قبلی است و به همان اندازه اولیه است.
- طول لیست یک ابرپارامتر است و معمولا برابر اندازه طولانی‌ترین جمله است.
- در ماژول self-attention کلمات به هم مرتبط هستند، اما در ماژول FFN به صورت جدا و موازی بردارها ادامه پیدا می‌کنند.

تنسورها و نحوه انتقال اطلاعات – ادامه

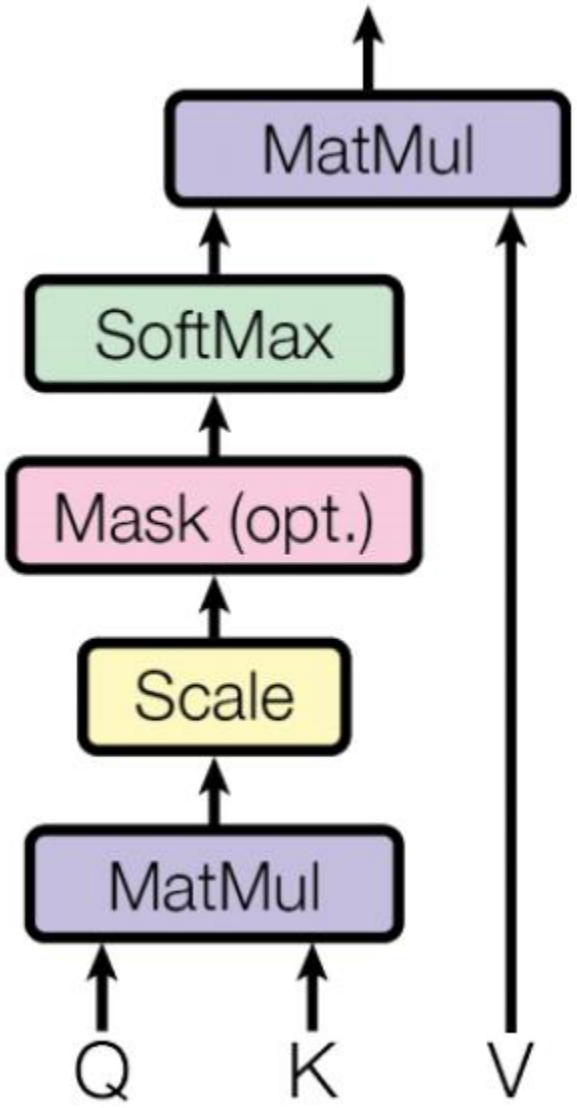


ماژول خودتوجه از نگاه سطح بالا

”The animal didn't cross the street because it was too tired”

- در جمله بالا مرجع ضمیر it چیست؟ animal یا street
- این موضوع برای انسان ساده است. اما درکش برای کامپیوتر آسان نیست.
- در فرایند پردازش این جمله، ماژول self-attention این امکان را می‌دهد که مرجع ضمیر it کلمه animal تداعی شود.
- وقتی مدل هر کلمه را پردازش می‌کند، ماژول self-attention این امکان را ایجاد می‌کند تا مدل نگاهی به کلمات دیگر داشته باشد.

Self-Attention layer



لایه خودتوجه

MatMul: ضرب دو ماتریس

Q: Query

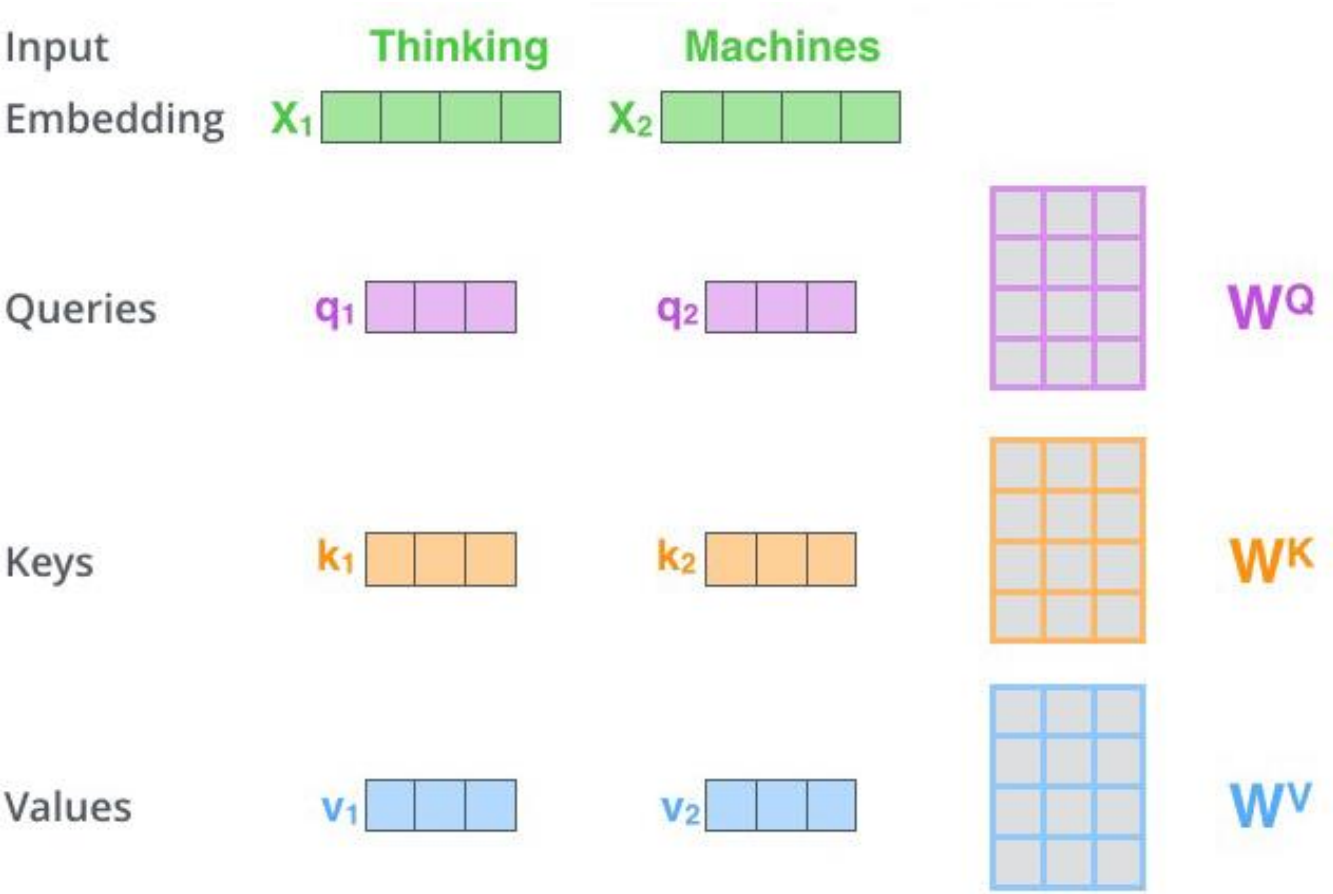
K: Key

V: Value

این لایه کمک می‌کند کدگذار در حین کدگذاری یک کلمه به **کلمات دیگر** توجه داشته باشد.

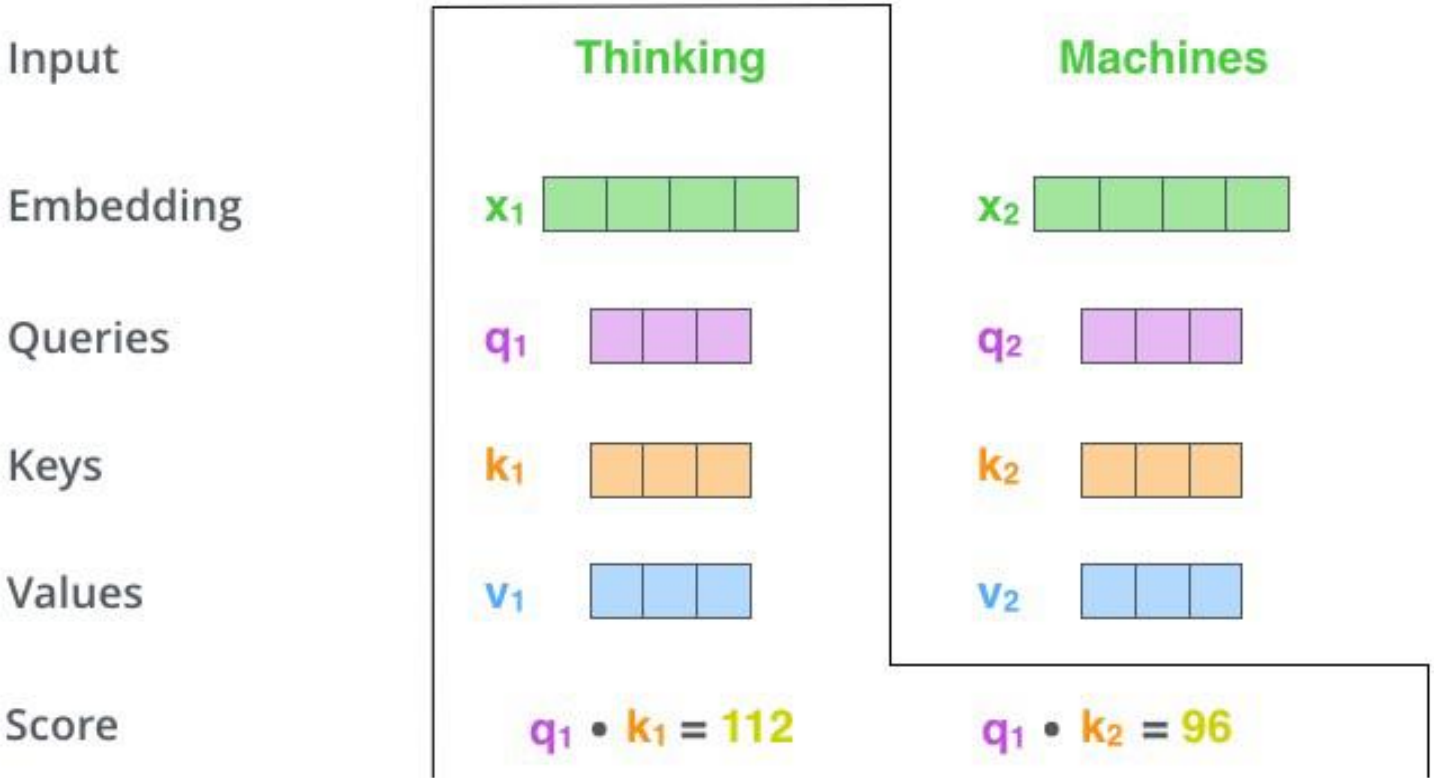
SELF-ATTENTION – گام صفر

برای هر جمله ورودی، ماتریس‌های Q، K و V از روی ماتریس تعبیه شده ورودی X ساخته می‌شود. این ماتریس‌ها حاصل ضرب ماتریسی X با W^Q، W^K و W^V است که در فرآیند آموزش به دست آمده‌اند.

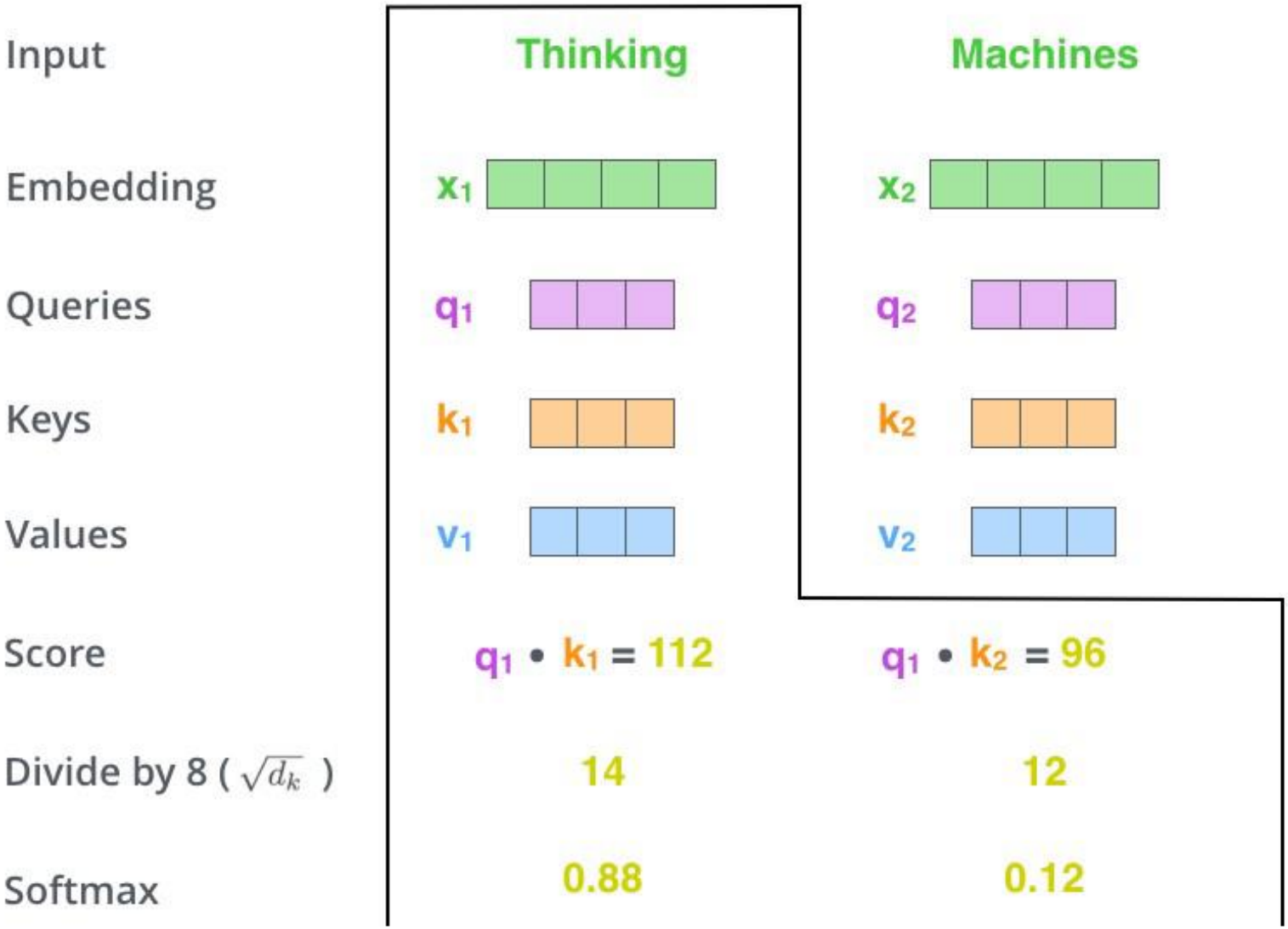


SELF-ATTENTION – گام اول: محاسبه امتیاز (SCORE)

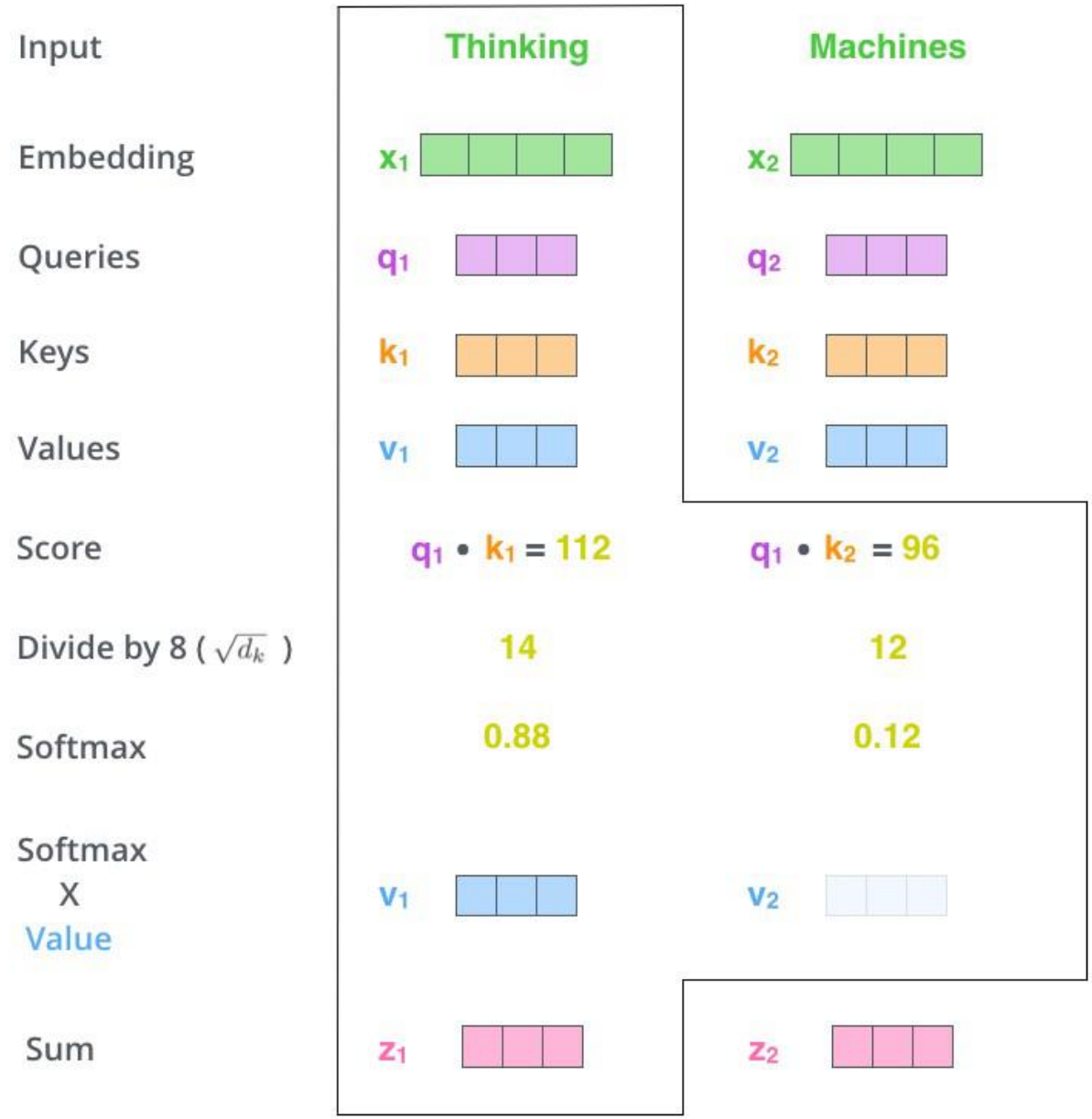
امتیاز تعیین می کند که وقتی کلمه ای را در یک موقعیت خاص رمزگذاری می کنیم، چه مقدار تمرکز روی سایر قسمت های جمله ورودی قرار دهیم. امتیاز از حاصل ضرب نقطه‌ای بردار پرس و جو کلمه مورد نظر با بردار کلید کلمات دیگر محاسبه می شود.



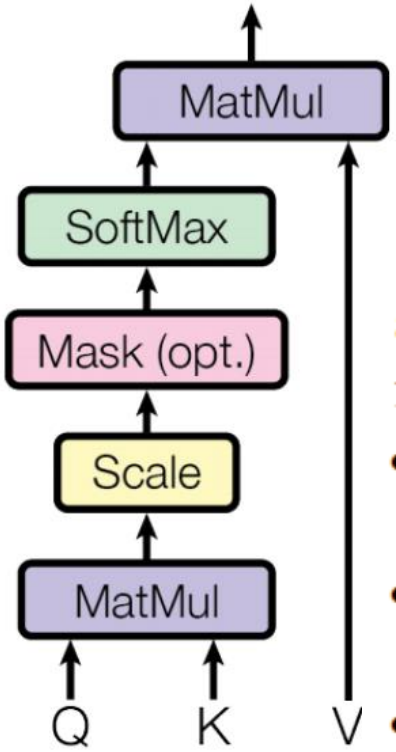
SELF-ATTENTION – گام دوم: نرمال‌سازی و محاسبه SOFTMAX



SELF-ATTENTION – گام سوم: محاسبه VALUE و جمع آن



معماری خودتوجه



attention function between different input vectors is calculated as follows

- **Step 1:** Compute scores between different input vectors with $\mathbf{S} = \mathbf{Q} \cdot \mathbf{K}^\top$;
- **Step 2:** Normalize the scores for the stability of gradient with $\mathbf{S}_n = \mathbf{S}/\sqrt{d_k}$;
- **Step 3:** Translate the scores into probabilities with softmax function $\mathbf{P} = \text{softmax}(\mathbf{S}_n)$;
- **Step 4:** Obtain the weighted value matrix with $\mathbf{Z} = \mathbf{V} \cdot \mathbf{P}$.

The process can be unified into a single function:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q} \cdot \mathbf{K}^\top}{\sqrt{d_k}}) \cdot \mathbf{V}$$

$$\text{SoftMax}\left(\vec{z}\right)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

توضیح مراحل

Step 1 computes scores between each pair of different vectors, and these scores determine the degree of attention that we give other words when encoding the word at the current position.

Step 2 normalizes the scores to enhance gradient stability for improved training.

Step 3 translates the scores into probabilities.

Finally, each value vector is multiplied by the sum of the probabilities. Vectors with larger probabilities receive additional focus from the following layers.

The attention layer in the decoder module is similar to the encoder module with the following exceptions: The key matrix K and value matrix V are derived from the encoder module, and the query matrix Q is derived from the previous layer.

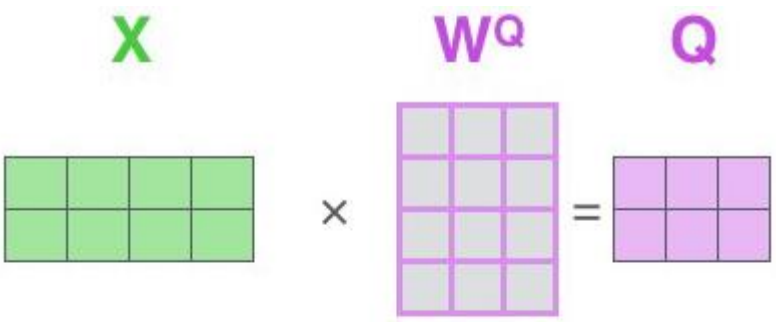
How should one understand the queries, keys, and values

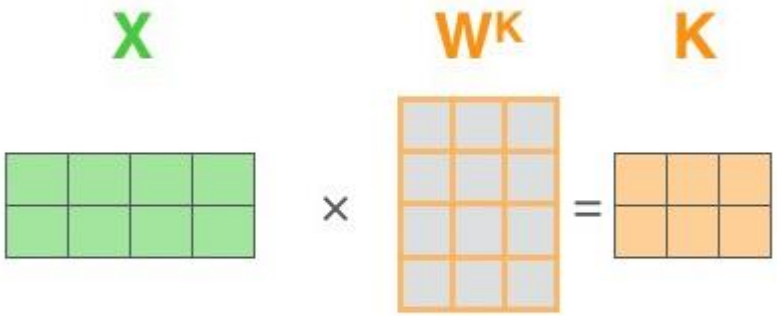
- The key/value/query concepts come from retrieval systems. For example, when you type a query to search for some video on YouTube, the search engine will map your **query** against a set of **keys** (video title, description, etc.) associated with candidate videos in the database, then present you the best matched videos (**values**).
- The attention operation can be thought of as a retrieval process as well, so the key/value/query concepts also apply here.

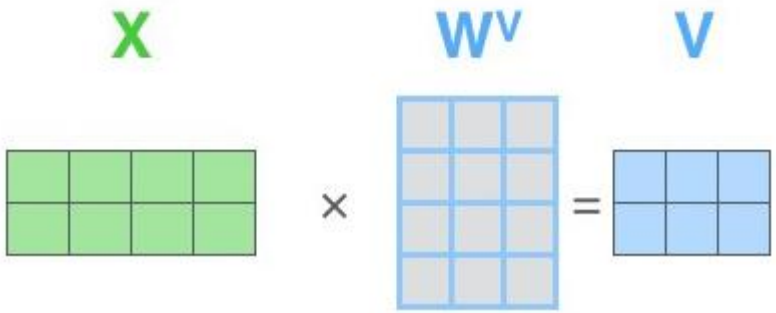
How are the queries, keys, and values obtained

- The proposed attention architecture alone doesn't say much about how the queries, keys, and values are obtained, they can come from different sources depending on the application scenario.
  - For unsupervised language model training like GPT, Q,K,V are usually from the same source, so such operation is also called self-attention.
  - For machine translation task, it first applies self-attention separately to source and target sequences, then on top of that it applies another attention where Q is from the target sequence and K,V are from the source sequence.
  - For recommendation systems, Q can be from the target items, K,V can be from the user profile and history.

محاسبه Self-Attention به صورت ماتریسی





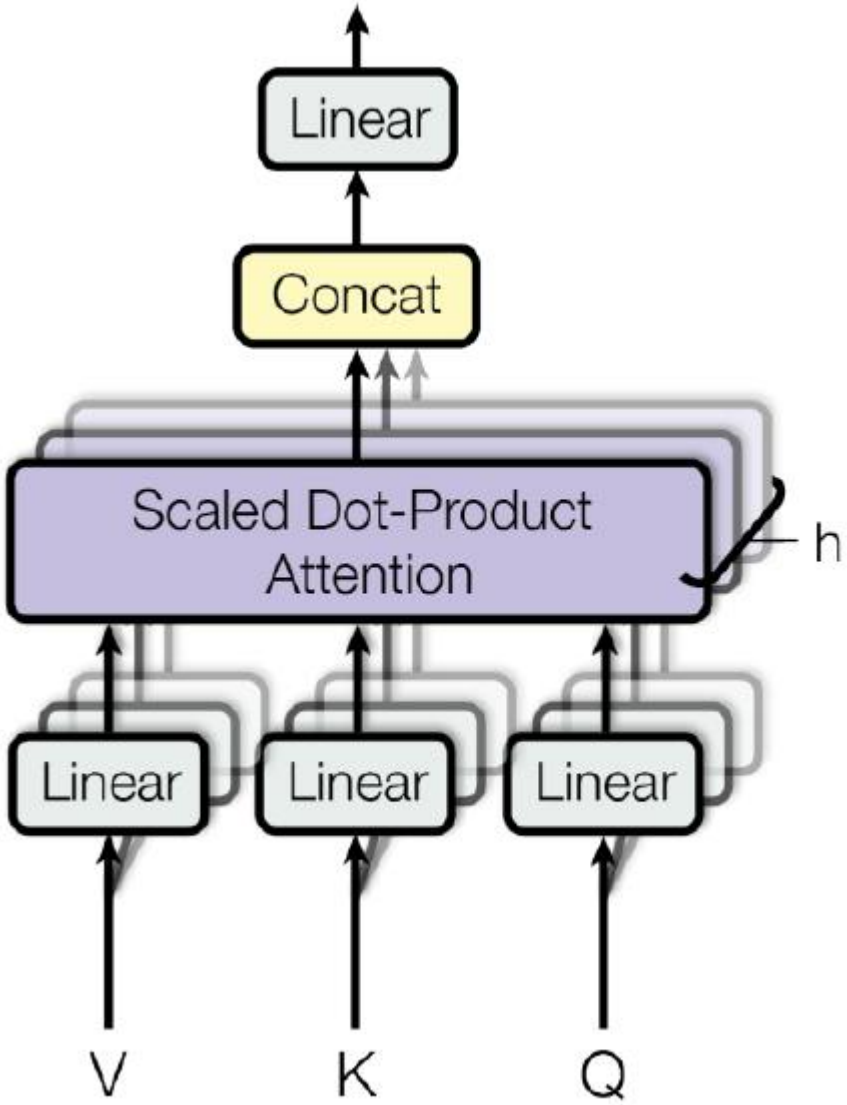


$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

مکانیزم توجه چندسر Multi-Head Attention

- روشی برای بهبود کارایی لایه خودتوجه پایه است.
- مکانیزم خودتوجه یکسر تنها بر روی یک یا تعداد محدودی نقطه متمرکز می‌شود، این در حالی است که در برخی از مواقع باید به مکان‌های مختلفی توجه نمود.
- برای رسیدن به این هدف سرهای متعددی طراحی می‌شود که هرکدام در زمان یادگیری با شرایط اولیه متفاوتی آموزش دیده‌اند. از این طریق نمایش‌های متنوعی حاصل می‌شود.

Multi-Head Attention – (2)



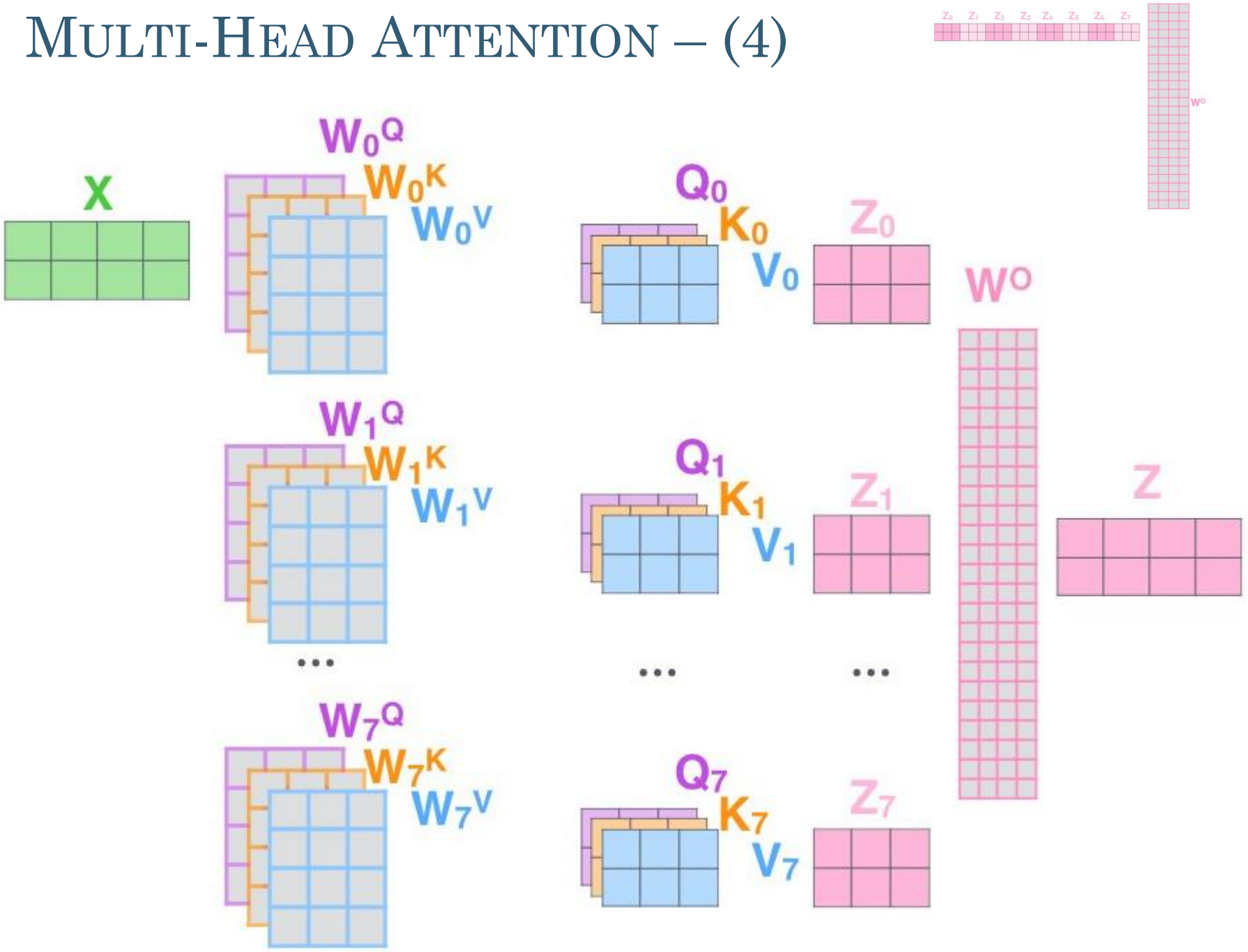
Multi-Head Attention – (3)

In each group, there are h vectors with dimension $d_{q'} = d_{k'} = d_{v'} = d_{model}/h = 64$. The vectors derived from different inputs are then packed together into three different groups of matrices: $\{\mathbf{Q}_i\}_{i=1}^{h}$, $\{\mathbf{K}_i\}_{i=1}^{h}$ and $\{\mathbf{V}_i\}_{i=1}^{h}$.

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}', \mathbf{K}', \mathbf{V}') &= \text{Concat}(\text{head}_1, \cdots, \text{head}_h)\mathbf{W}^o, \\ \text{where head}_i &= \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i). \end{aligned} \tag{10}$$

Here, $\mathbf{Q}'$ (and similarly $\mathbf{K}'$ and $\mathbf{V}'$) is the concatenation of $\{\mathbf{Q}_i\}_{i=1}^{h}$, and $\mathbf{W}^o \in \mathbb{R}^{d_{model} \times d_{model}}$ is the linear projection matrix.

Multi-Head Attention – (4)



Position of each word مکان هر کلمه

- معماری خودتوجه مکان یک کلمه را در نظر نمی‌گیرد.
- ویژگی ترتیبی بودن کلمات جمله ایجاب می‌کند این موضوع در نظر گرفته شود.
- برای در نظر گرفتن این اطلاعات کدگذاری مکانی با بعد d_{model} به داده‌های تعبیه شده اضافه می‌شود.

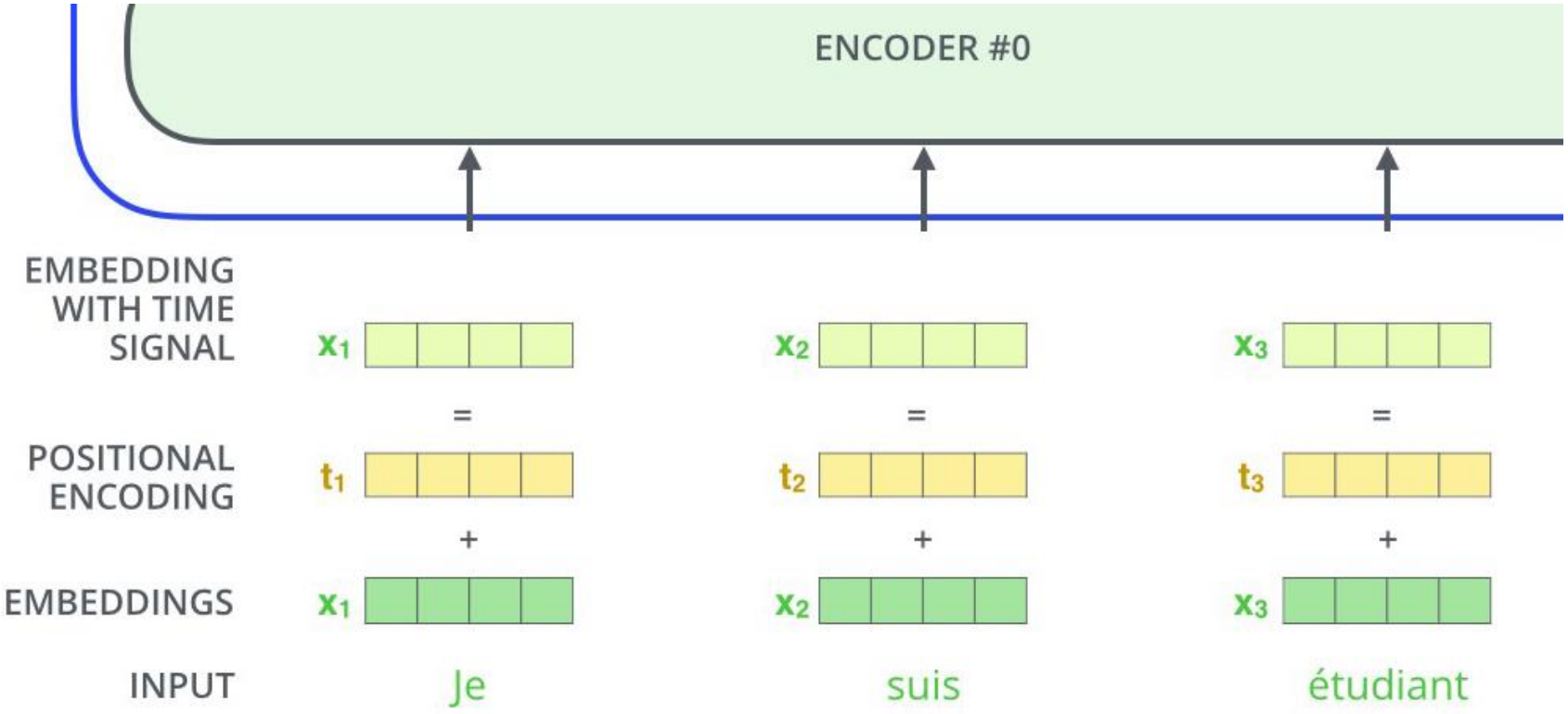
رابطه مکان هر کلمه

$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}});$$
$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}),$$

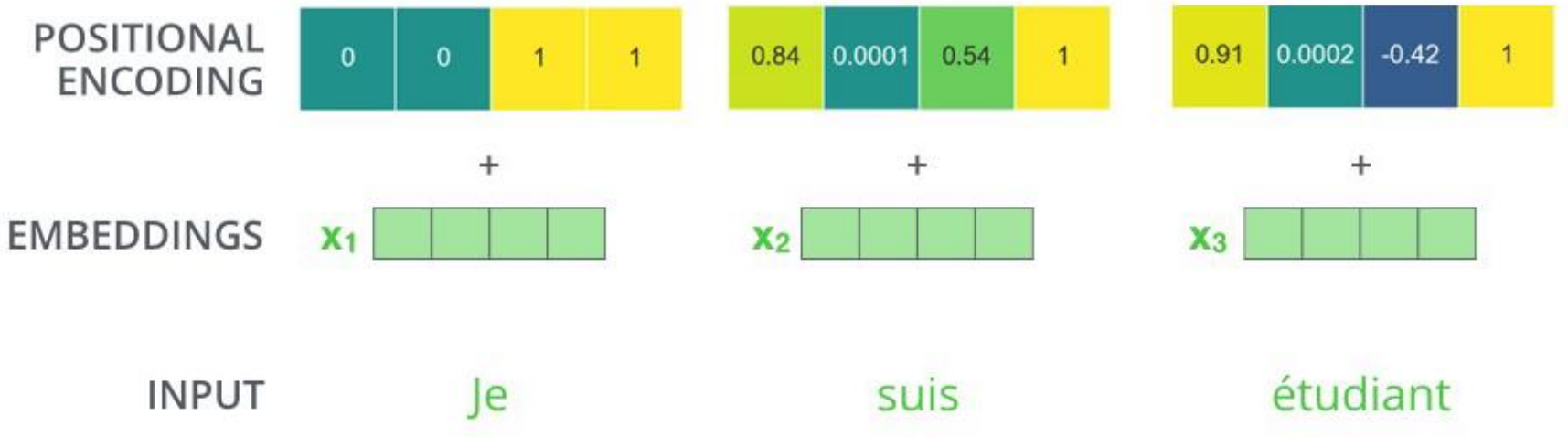
- pos = مکان یک کلمه در جمله
- i = بعد مورد نظر در ماتریس تعبیه

- each element of the positional encoding corresponds to a sinusoid, and it allows the transformer model to learn to attend by relative positions and extrapolate to longer sequence lengths during inference.

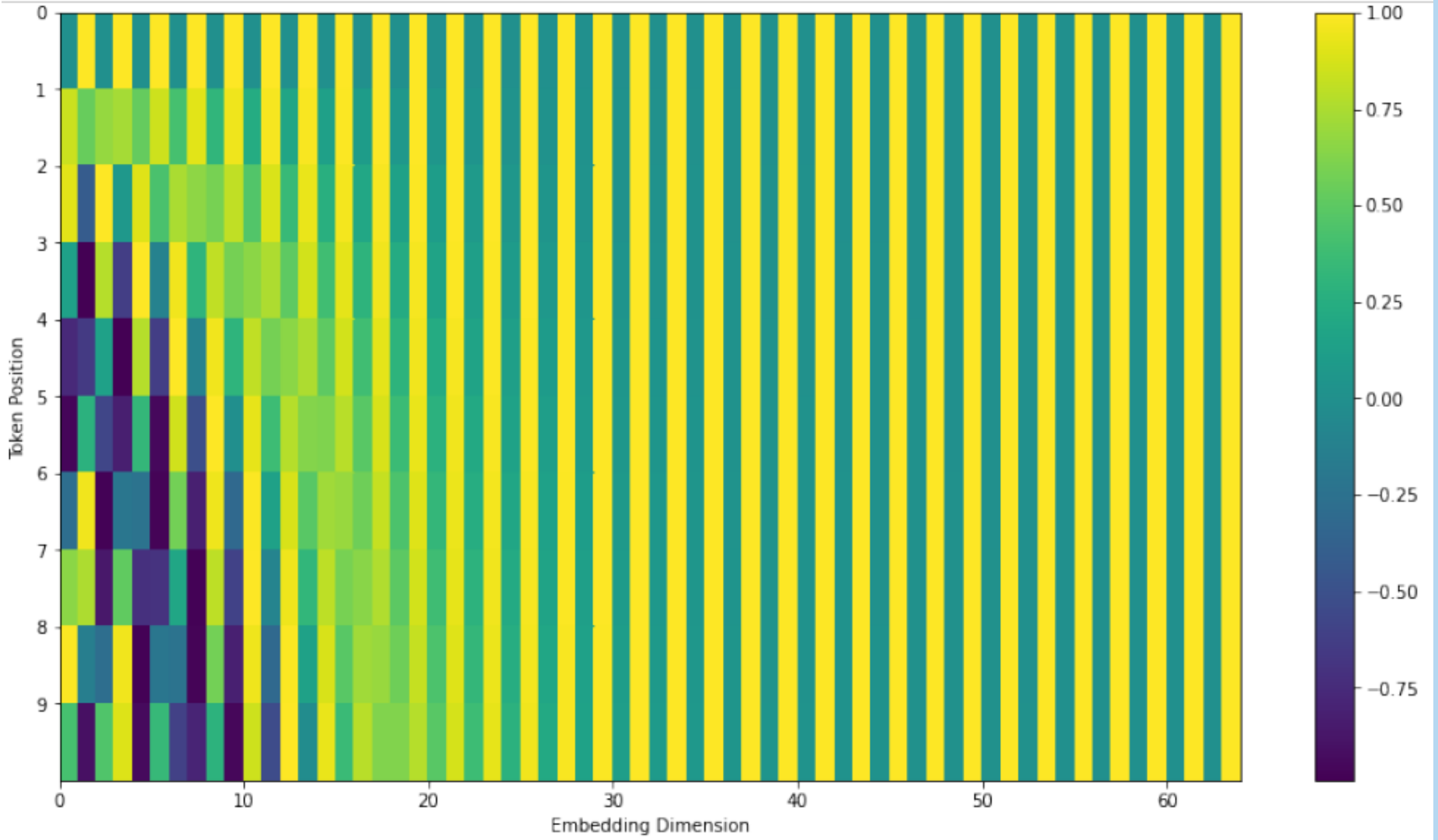
رابطه مکان هر کلمه – مثال



کدگذاری مکان هر کلمه – مثال

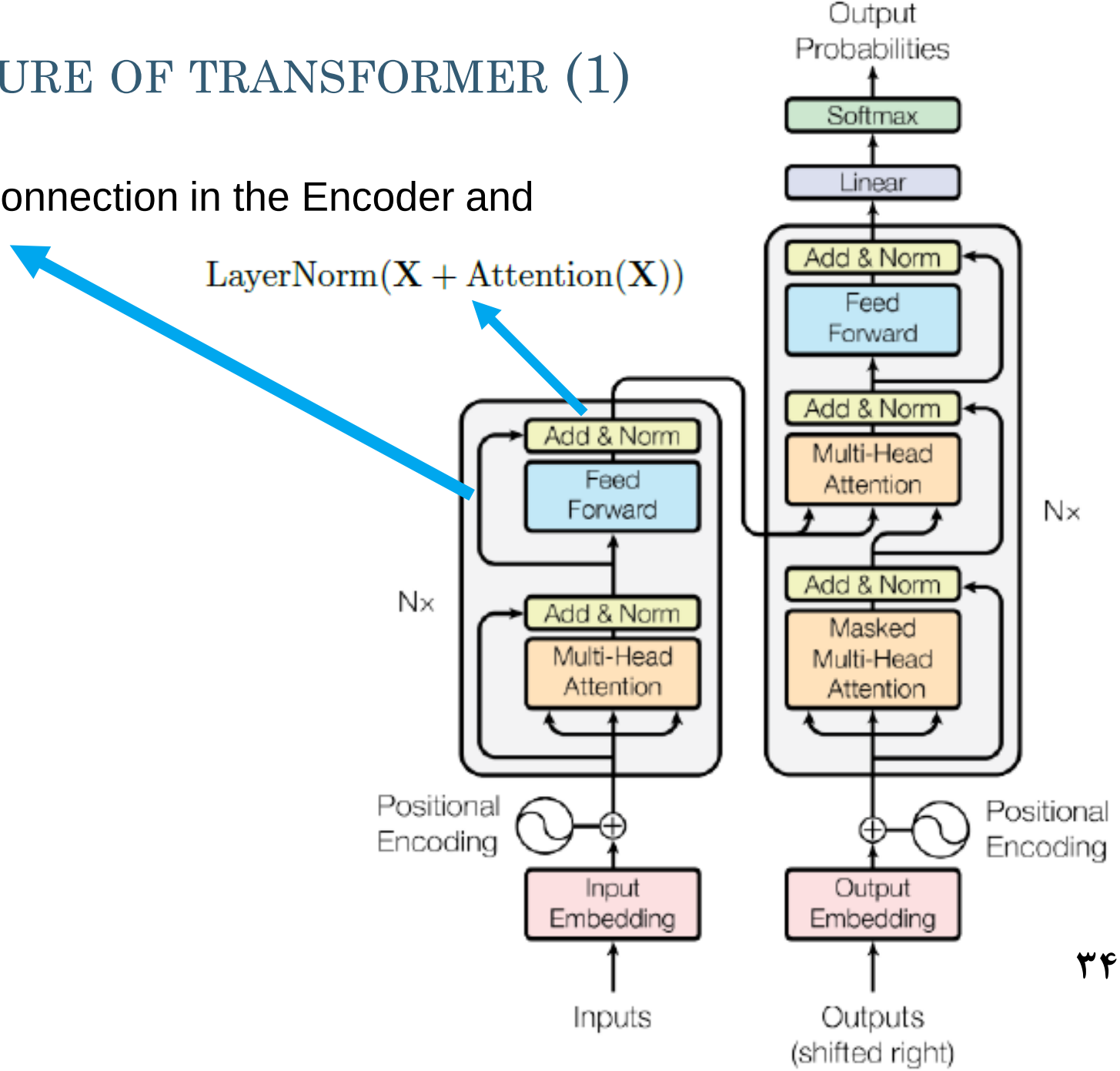


نمایشی تصویری از اعداد کدگذاری مکانی



Structure of transformer (1)

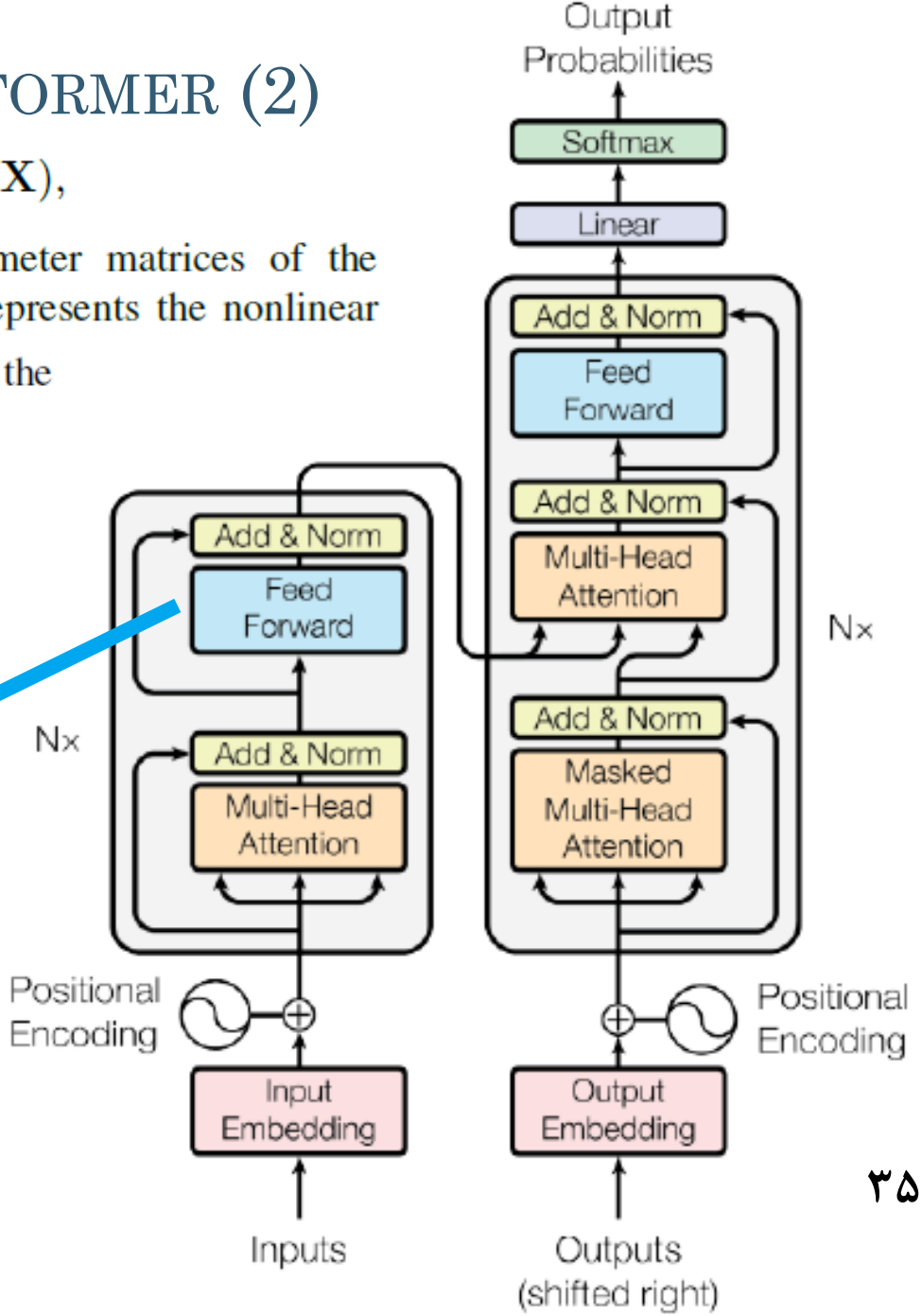
Residual Connection in the Encoder and Decoder



Structure of transformer (2)

$$\mathrm{FFN}(\mathbf{X}) = \mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{X}),$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are the two parameter matrices of the two linear transformation layers, and σ represents the nonlinear activation function . The dimensionality of the hidden layer is $d_h = 2048$.

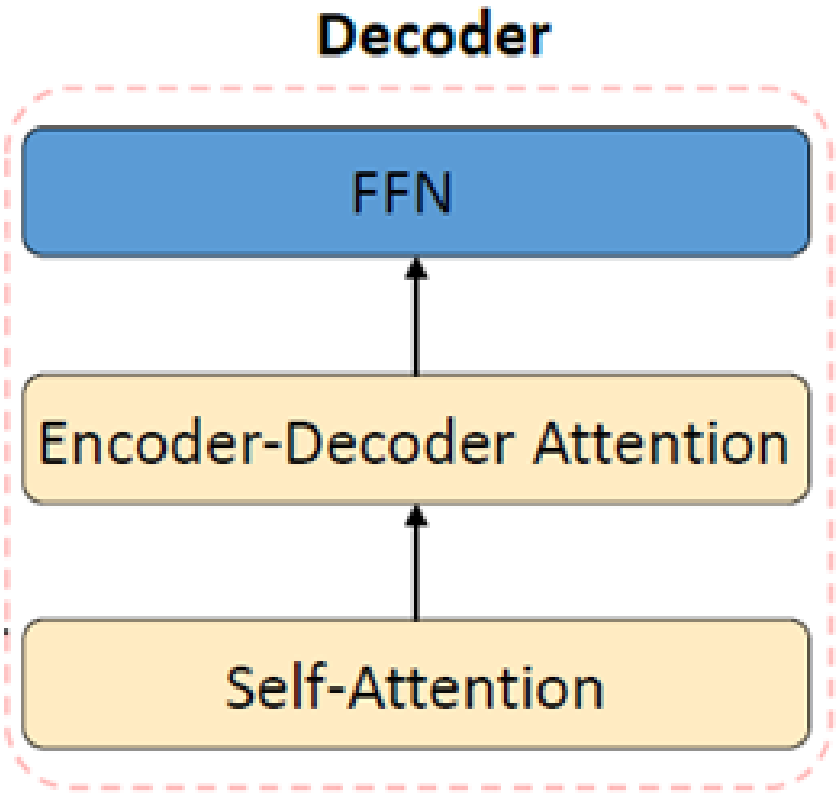


Feed Forward Network (FFN)

ساختار لایه توجه کدگذار – کدگشا در ماژول کدگشا
Encoder-Decoder Attention layer

لایه توجه کدگذار-کدگشا در ماژول کدگشا مشابه لایه خودتوجه در ماژول کدگذار است با یک تفاوت:

ماتریس کلیدی K و V از بالاترین لایه ماژول کدگذار به دست می‌آید و Q از لایه قبل به دست می‌آید.



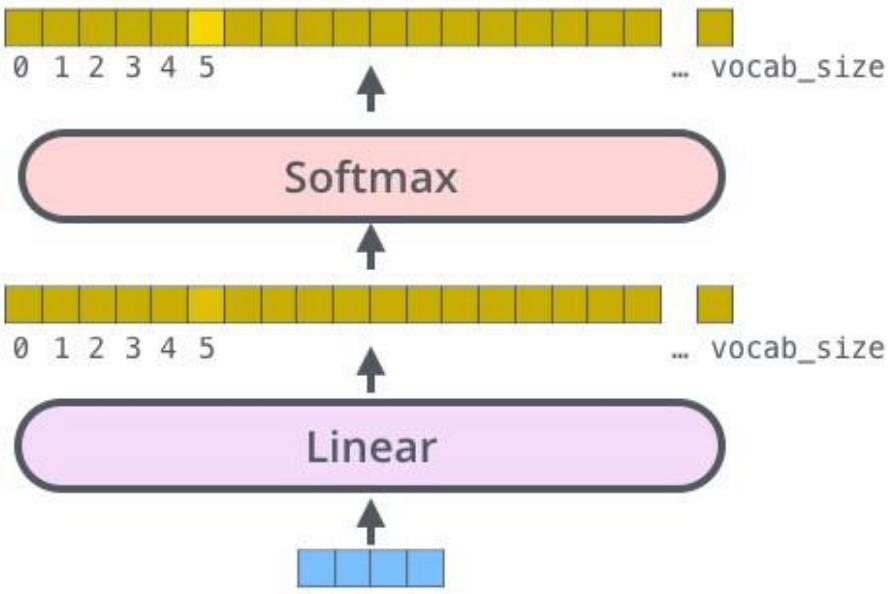
The Final Linear and Softmax Layer

Which word in our vocabulary is associated with this index?

am

Get the index of the cell with the highest value (**argmax**)

5



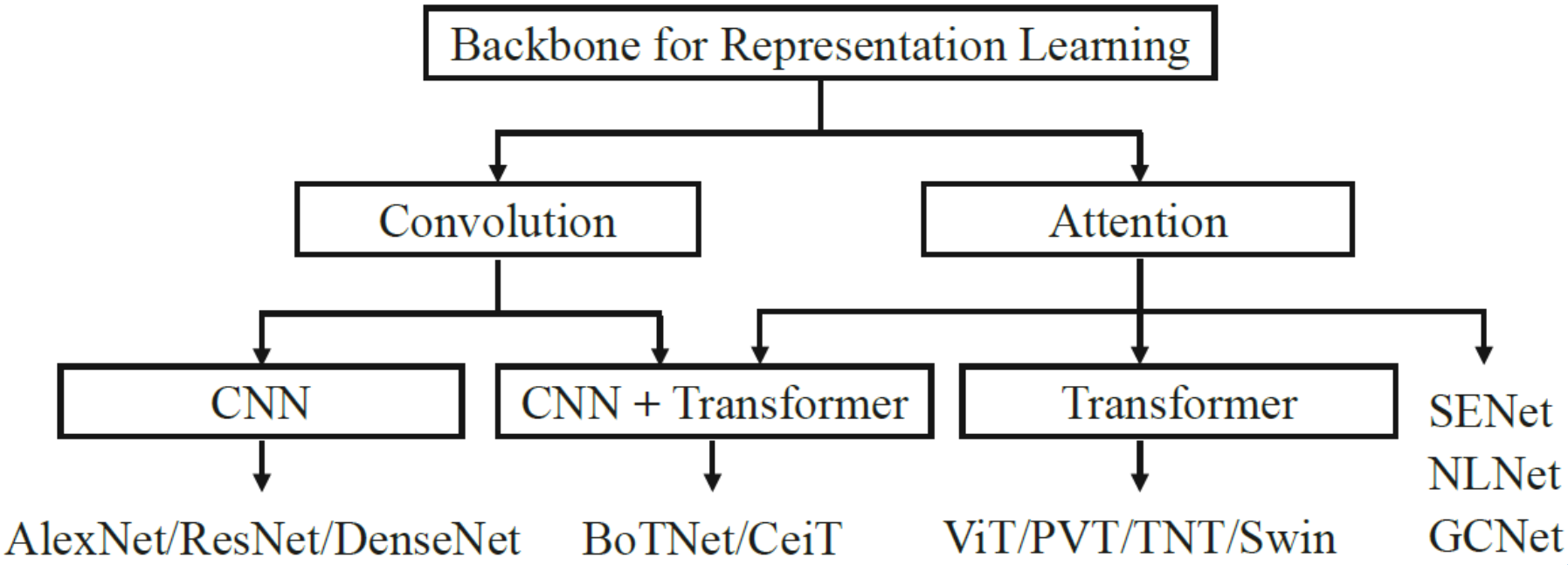
BERT: Bidirectional Encoder Representations from Transformers

- روشی برای بازنمایی زبانی است.
- این روش بر روی داده‌های بدون برچسب آموزش داده شده است یا به عبارتی Pre-train شده است.
- برای تنظیم دقیق (fine-tune) آن برای یک کاربرد خاص می‌توان یک لایه بر بالای آن اضافه نمود و آن را آموزش داد.
- این مبدل در حقیقت این قابلیت را دارد که برای تعبیه‌سازی جمله به کار رود.
- استفاده از آن بسیار آسان است.

Transformers in CV area

- When used for CV tasks, most transformers adopt the original transformer's encoder module. Such transformers can be treated as a new type of feature extractor.
- Compared with CNNs which focus only on local characteristics, transformer can capture long distance characteristics, meaning that it can easily derive global information.
- And in contrast to RNNs, whose hidden state must be computed sequentially, transformer is more efficient because the output of the self-attention layer and the fully connected layers can be computed in parallel and easily accelerated.

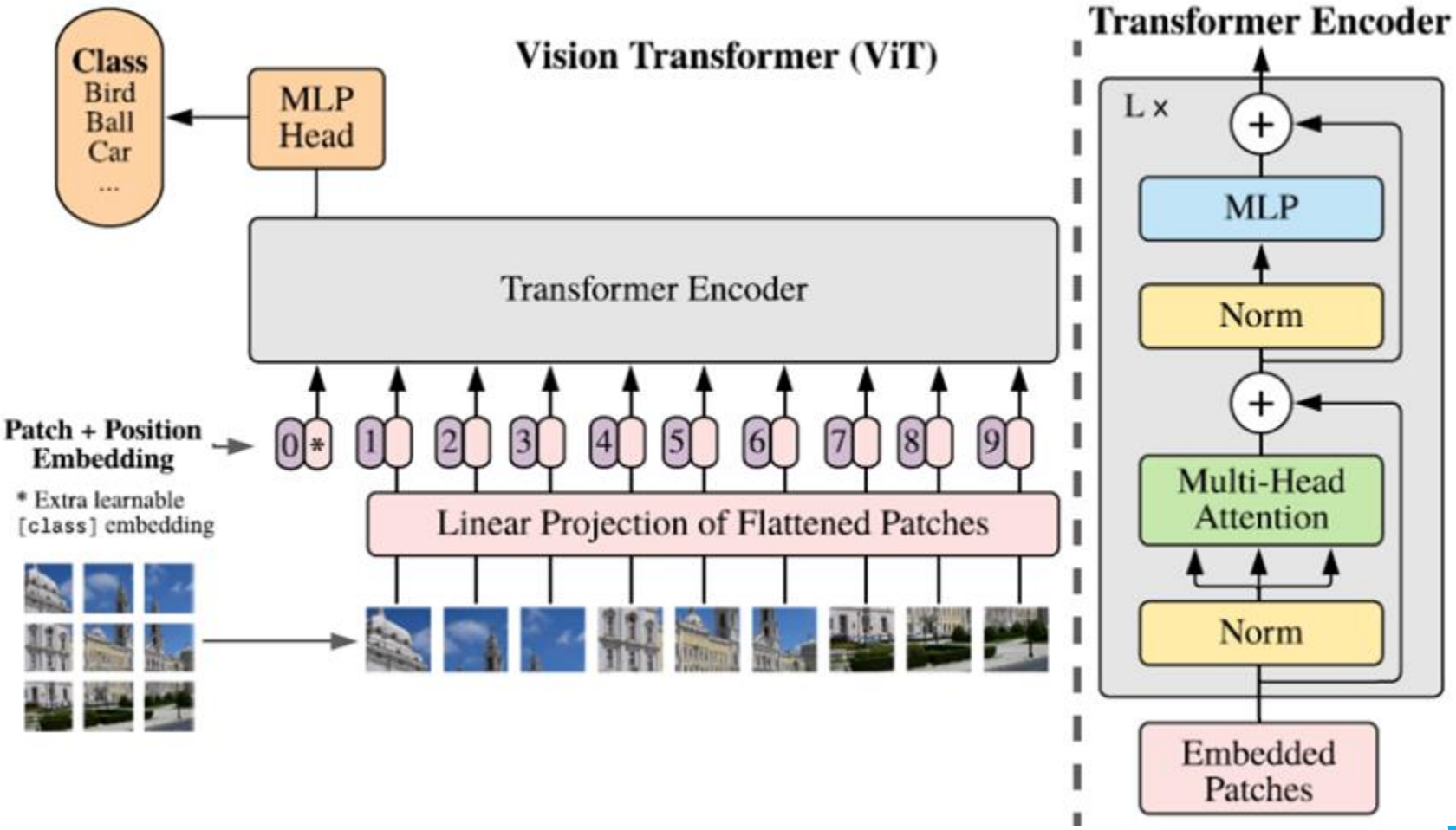
زیرساخت‌های استفاده شده در یادگیری بازنمایی



ViT. Vision Transformer

- با الهام از موفقیت های مبدل‌ها در پردازش زبان طبیعی، این روش مبدل استاندارد را به طور مستقیم روی تصاویر، اعمال نمود.
- برای انجام این کار، یک تصویر را به تکه‌هایی تقسیم کرد و دنباله تعبیه‌شده این تکه‌ها به عنوان ورودی به یک مبدل ارایه شد. تکه‌های تصویری مانند کلمات در یک کاربرد برنامه پردازش زبان طبیعی رفتار می شوند.
- مدل در کاربرد طبقه بندی تصویر به روش نظارت شده آموزش داده شد.

معماری ViT

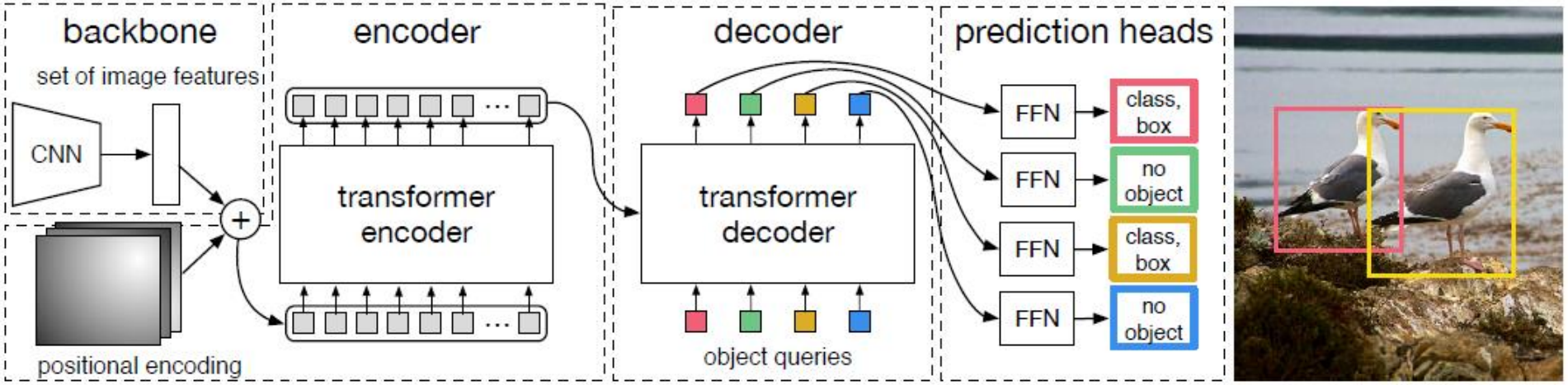

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 * 1 2 3 4 5 6 7 8 9
Linear Projection of Flattened Patches
Transformer Encoder
L x
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches

مقایسه ViT با ResNet

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

End-to-End Object Detection with Transformers
DETR - DEtection TRansformer

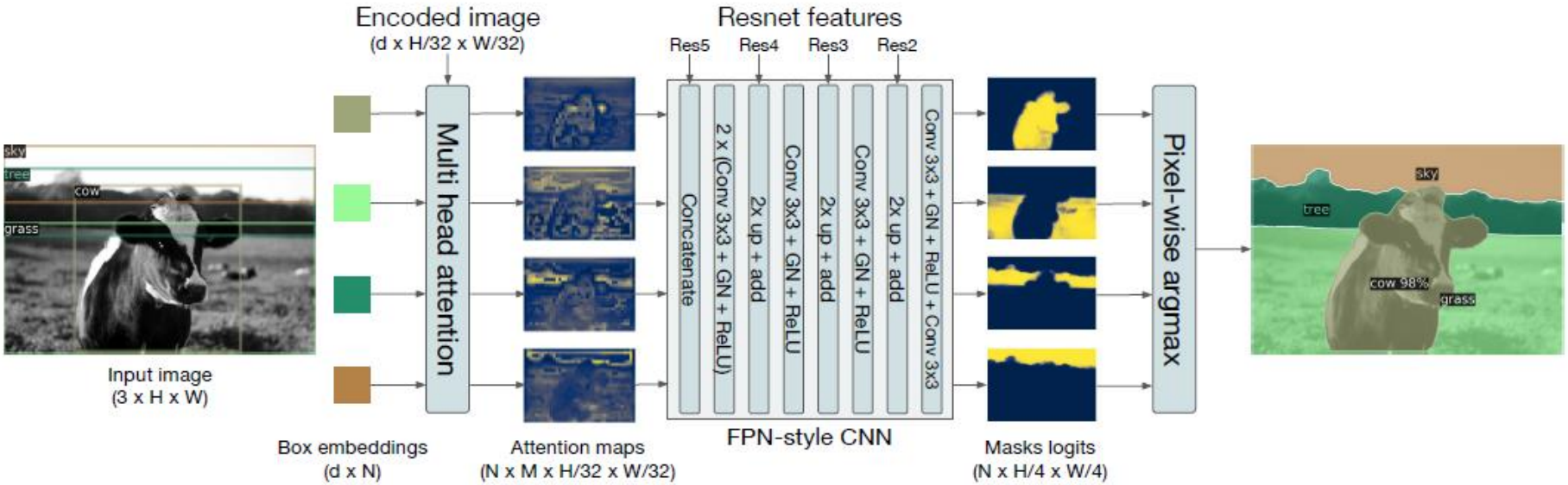
- معماری کلی DETR به طرز شگفت آوری ساده است. شامل سه جزء اصلی است که در زیر توضیح می دهیم: یک شبکه کانولوشنی پایه برای بازنمایی ویژگی فشرده، یک مبدل کدگذار-کدگشا، و یک شبکه پیشخور ساده (FFN) که پیش‌بینی تشخیص نهایی را انجام می‌دهد.



مقایسه DETR با Faster RCNN

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**

DETR FOR PANOPTIC SEGMENTATION



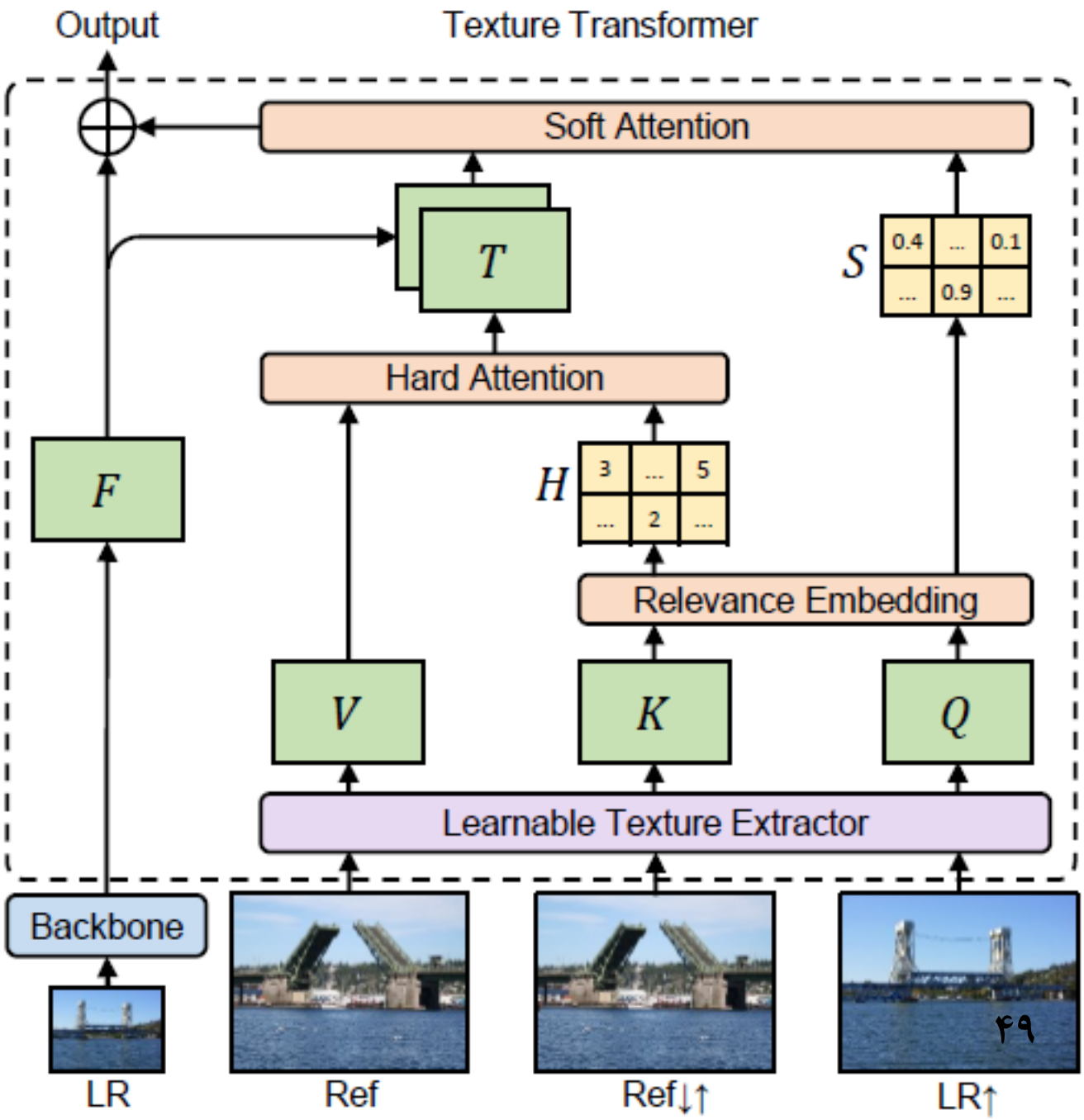
Training data-efficient image transformers & distillation through attention

- In this paper, competitive convolution-free transformers has been produced by training on Imagenet only.
- They have introduced a teacher-student strategy specific to transformers. It relies on a distillation token ensuring that the student learns from the teacher through attention.
- We show the interest of this token-based distillation, especially when using a convnet as a teacher.

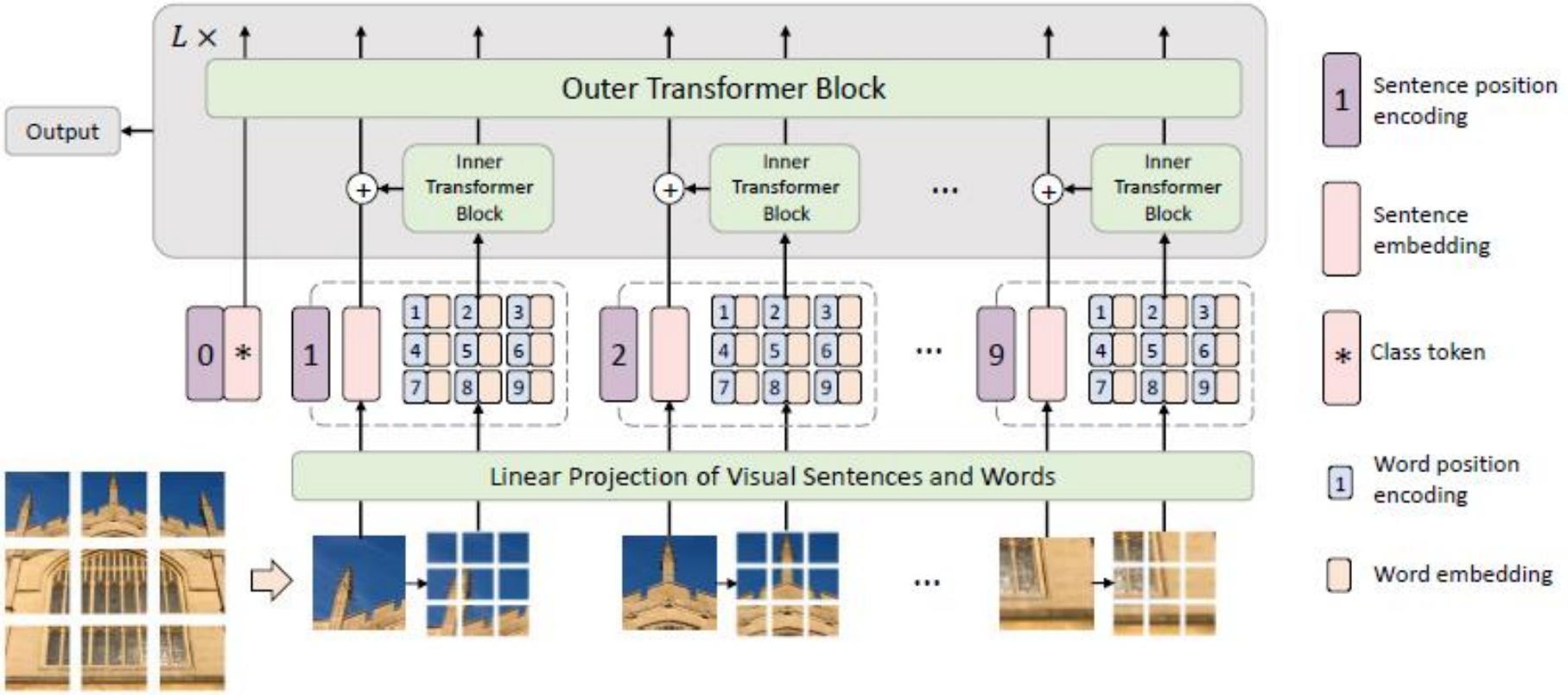
Texture Transformer Network for Image Super-Resolution (TTSR),

- TTSR consists of four closely-related modules optimized for image generation tasks:
  - a learnable texture extractor by DNN,
  - a relevance embedding module,
  - a hard-attention module for texture transfer,
  - a self-attention module for texture synthesis.

TTSR


Output
Texture Transformer
Soft Attention
T
S
0.4
...
0.1
...
0.9
...
Hard Attention
F
H
3
...
5
...
2
...
Relevance Embedding
V
K
Q
Learnable Texture Extractor
Backbone
LR
Ref
Ref↓↑
LR↑

TNT- Transformer-iN-Transformer



This paper shows that the attention inside these local patches are also essential for building visual transformers with high performance and explores a new architecture, namely, TNT.

Specifically, we regard the local patches (16×16) as "visual sentences" and present to further divide them into smaller patches (4×4) as "visual words".

References

- https://jalammar.github.io/illustrated-transformer/

- Kai Han, et al. “A Survey on Vision Transformer”, https://arxiv.org/abs/2012.12556.