昆山杜克大学
DUKE KUNSHAN UNIVERSITY

transformer

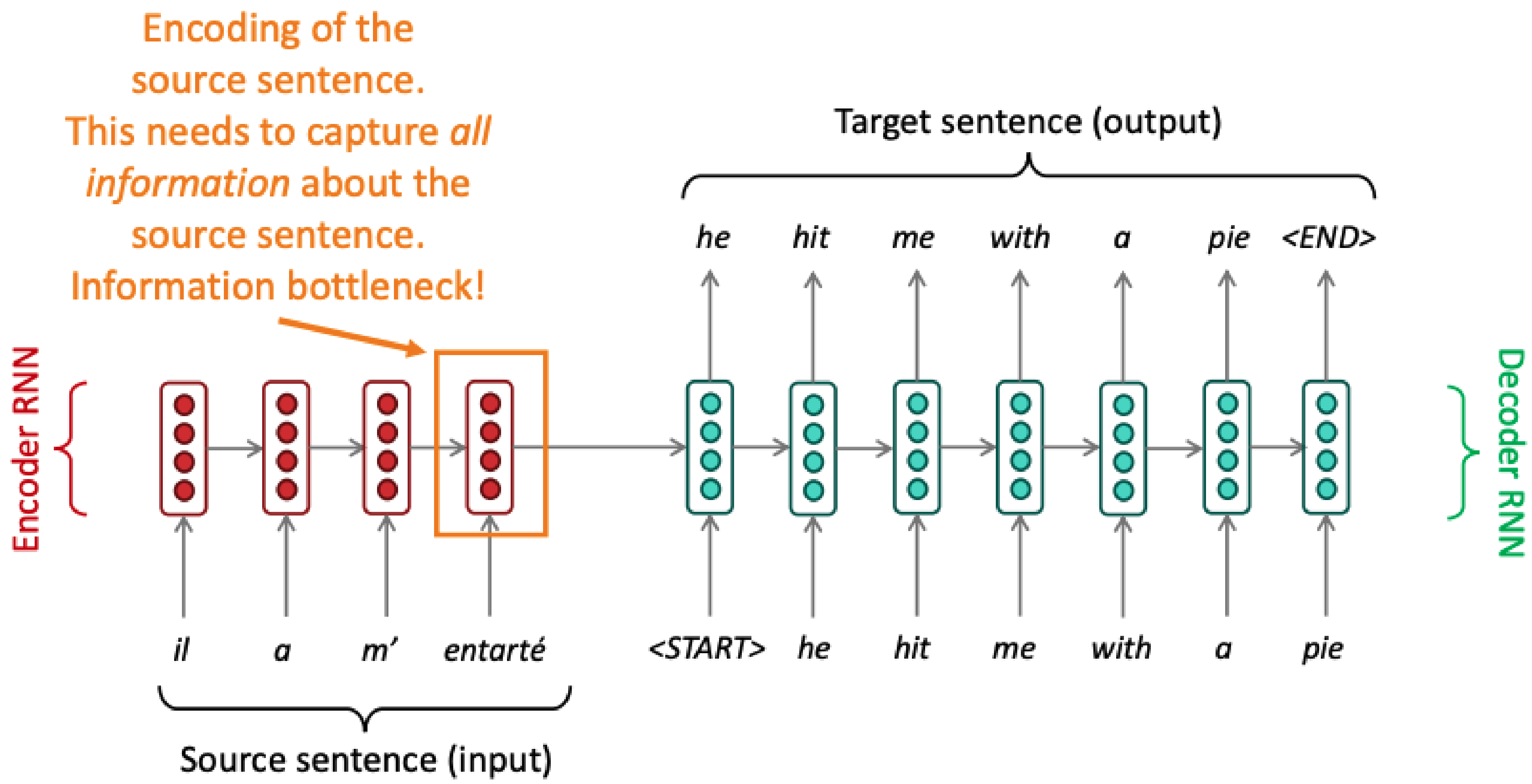
stats403_deep_learning

spring_2025

lecture_7

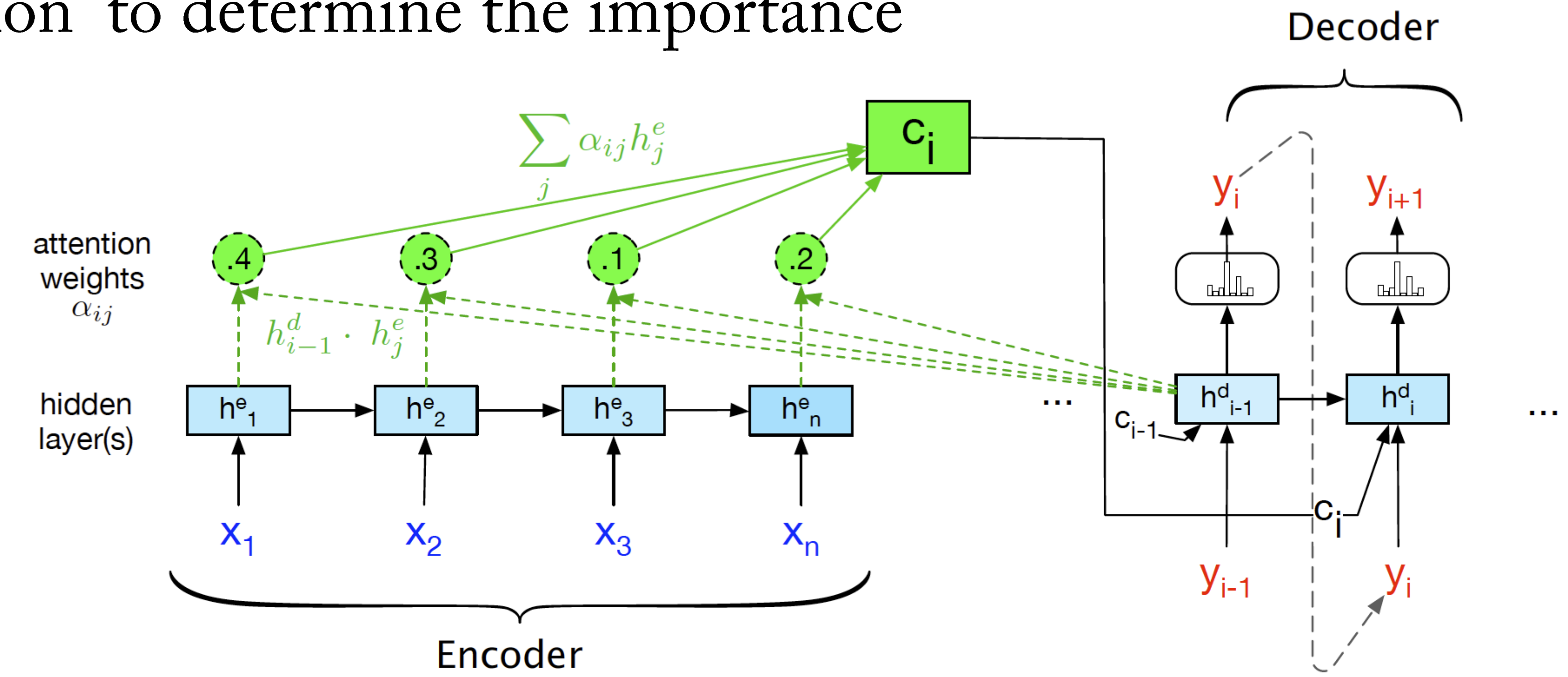
7.1 attention and self-attention

drawback of RNN encoder-decoder



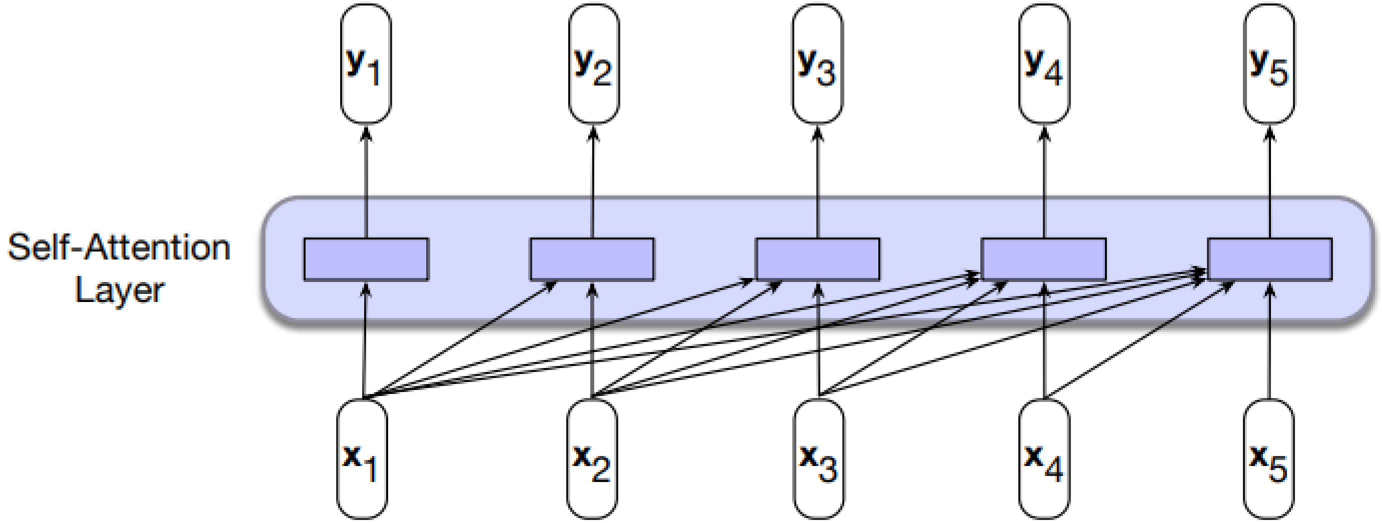
machine translation: bottleneck

- idea: map all hidden states to all output state by learning an alignment with weights
- use "attention" to determine the importance



transformer

- a transformer maps a sequence of input vectors $(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ to a sequence of output vectors $(\boldsymbol{y}_1, \dots, \boldsymbol{y}_n)$
- the most important layer in a transformer is the self-attention layer



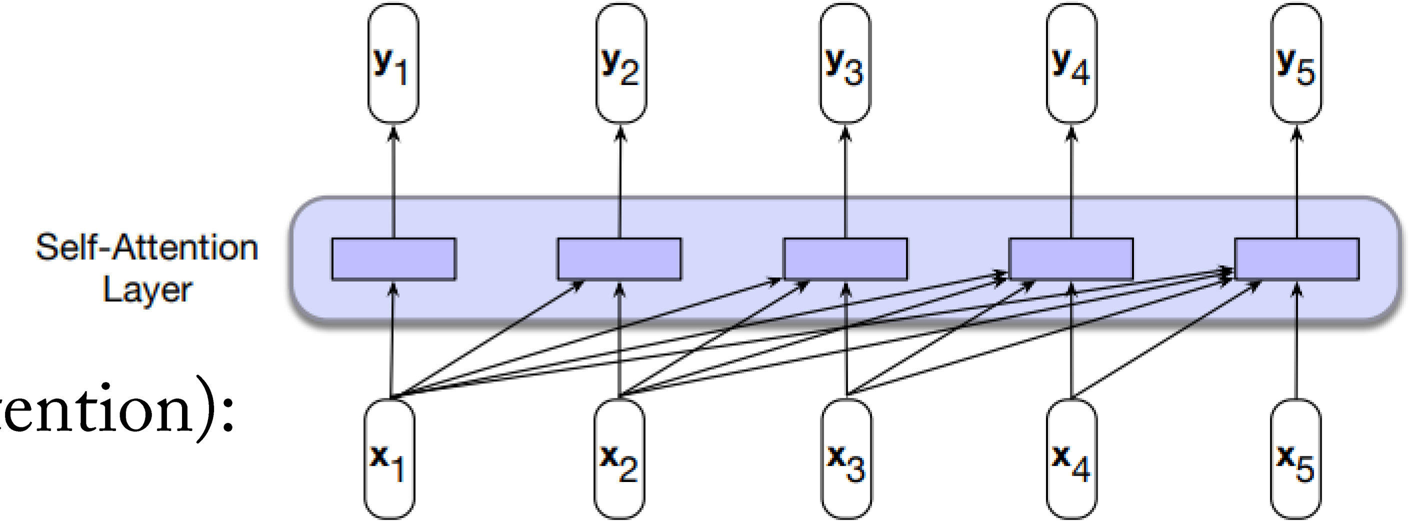
an example of (causal masked) self-attention which attends to all the inputs up to the current one

self-attention

- attention: compare an item of interest to a collection of other items in a way that reveals their relevance in the current context
- self-attention: the comparison is done within a given sequence

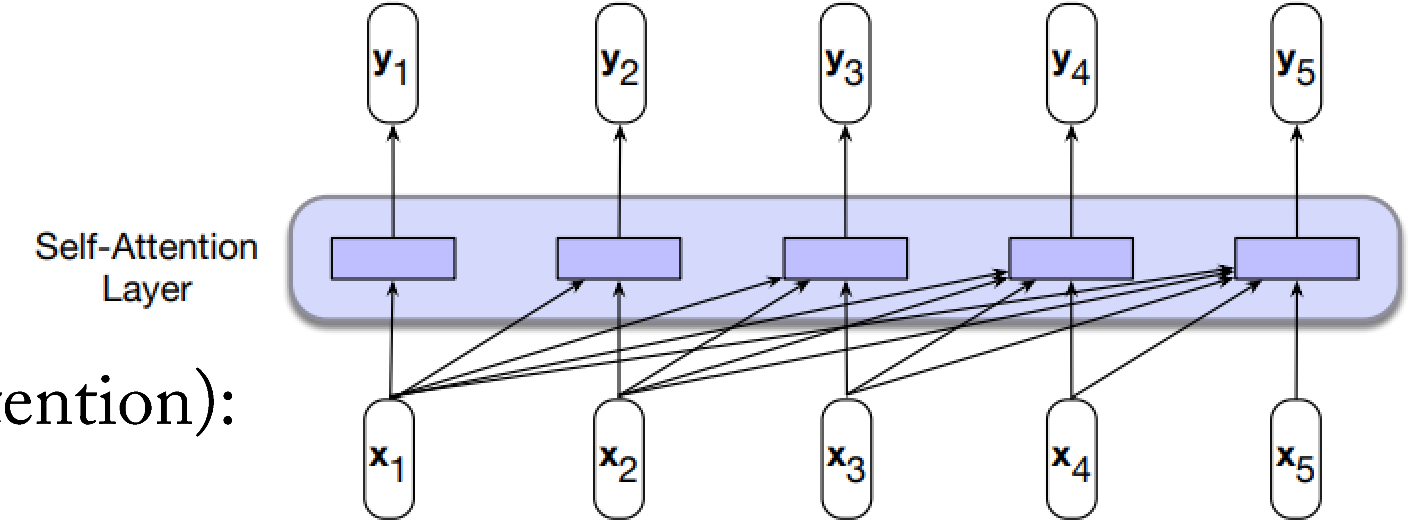
- the result of comparison is used to compute the output

self-attention



- example (a simple self-attention):
  - the simplest form of comparison is a dot product, e.g.,
    $$\text{score}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i\ , \boldsymbol{x}_j \rangle = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$
  - for example, to compute the output $\boldsymbol{y}_3$ in the above example, we need to compute $\text{score}(\boldsymbol{x}_3, \boldsymbol{x}_1)$, $\text{score}(\boldsymbol{x}_3, \boldsymbol{x}_2)$, $\text{score}(\boldsymbol{x}_3, \boldsymbol{x}_3)$

self-attention

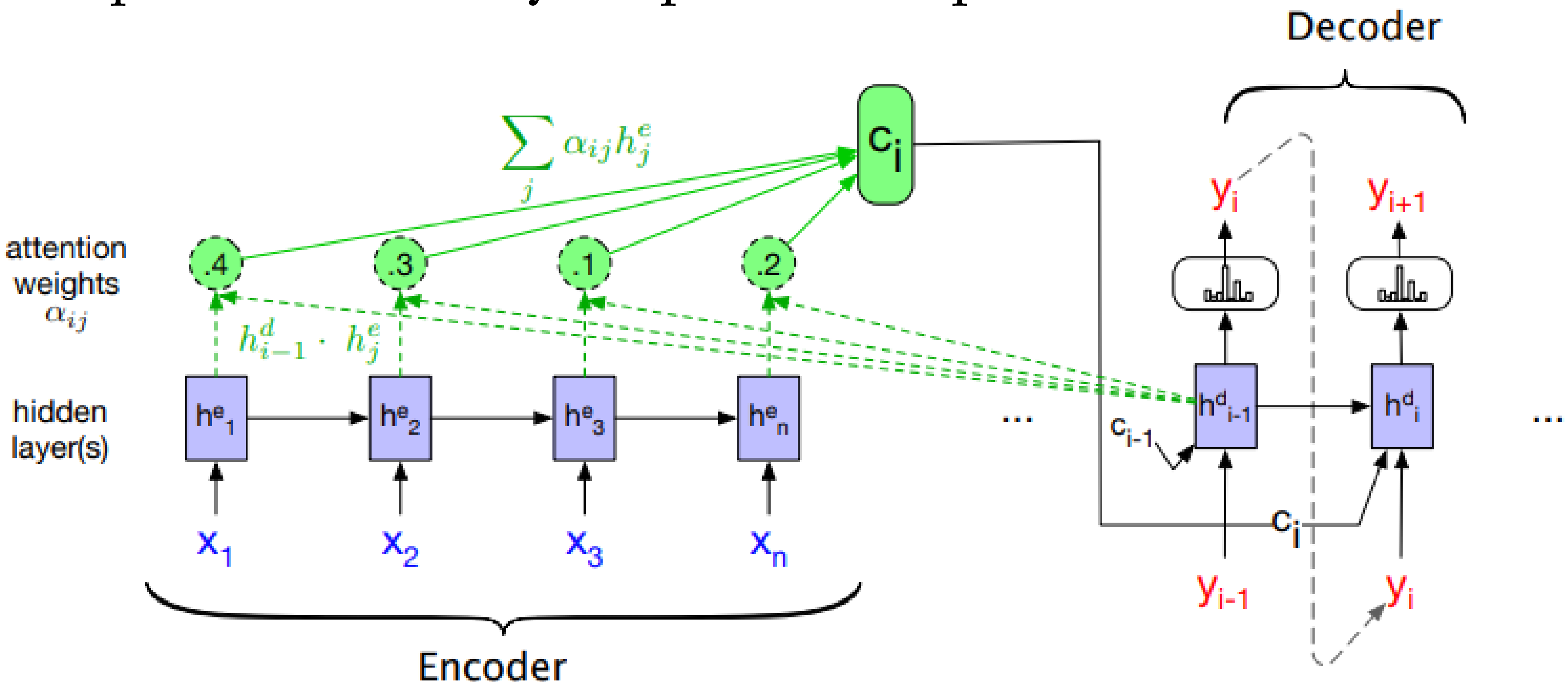


- example (a simple self-attention):
  - the output $\boldsymbol{y}_3$ then depends on a weighted average of $\boldsymbol{x}_1, \boldsymbol{x}_2$ and $\boldsymbol{x}_3$,
    $$\boldsymbol{y}_3 = \alpha_{31}\boldsymbol{x}_1 + \alpha_{32}\boldsymbol{x}_2 + \alpha_{33}\boldsymbol{x}_3$$
  - the weights α_{ij} are determined by these scores
  - it is naturally to normalize the scores using softmax, so that the weights are

$$\alpha_{ij} = \mathrm{softmax}\left(\mathrm{score}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) = \frac{\exp(\mathrm{score}(\boldsymbol{x}_i, \boldsymbol{x}_j))}{\sum_{k=1}^{i} \exp(\mathrm{score}(\boldsymbol{x}_i, \boldsymbol{x}_k))}, \qquad \text{for all } j \le i$$

self-attention

- an example of using the simple self-attention
- each output $\boldsymbol{y}_i$ (more precisely, the hidden $\boldsymbol{h}_i$) in the decoder has a context $\boldsymbol{c}_i$ which depends differently on previous input vectors



self-attention in transformer

- the self-attention regime in transformers is more complicated

- each input vector $\boldsymbol{x}_i$ is transformed into three vectors of different roles:
  - $\boldsymbol{q}_i$ (query): current input used in comparison
  - $\boldsymbol{k}_i$ (key): previous input used in comparison
  - $\boldsymbol{v}_i$ (value): input for taking weighted average

- they can be simply calculated by applying linear transformations:
  - $\boldsymbol{q}_i = \boldsymbol{W}_Q \boldsymbol{x}_i,\ \boldsymbol{k}_i = \boldsymbol{W}_K \boldsymbol{x}_i, \boldsymbol{v}_i = \boldsymbol{W}_V \boldsymbol{x}_i$

Freud's Structure of the Human Psyche

Id:

Instincts

Ego:

Reality

Superego:

Morality

self-attention in transformer

- if we make a simple analogy to the previous example, the scores are given by

$$\text{score}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{q}_i , \boldsymbol{k}_j \rangle = \boldsymbol{q}_i \cdot \boldsymbol{k}_j, \qquad i \leq j$$

- the output is then

$$\boldsymbol{y}_i = \sum_{j \leq i} \alpha_{ij} \boldsymbol{v}_j = \sum_{j \leq i} \text{softmax}(\boldsymbol{q}_i \cdot \boldsymbol{k}_j) \boldsymbol{v}_j$$

self-attention in transformer

- nevertheless, numerically it is often beneficial to normalize the dot product, so that

$$\text{score}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\langle \boldsymbol{q}_i\ , \boldsymbol{k}_j \rangle}{\sqrt{d_k}} = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d_k}}, \qquad i \leq j$$

where d_k is the dimensionality of the key vector

self-attention in transformer

- numerically, it is more efficient to implement the above in matrix forms by combining the input vectors into a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

- then we transform it using three matrices $\boldsymbol{W_Q}, \boldsymbol{W_K}, \boldsymbol{W_V} \in \mathbb{R}^{d \times d}$

- we get three output matrices of size $N \times d$:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W_Q}, \qquad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W_K}, \qquad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W_V}$$

self-attention in transformer

- in the matrix form, the weighted average of "values" can be written as

$$\text{selfAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\boldsymbol{V}$$

- however, we need to implement $\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}$ so that it is "causal" (that is, we calculate a score $\boldsymbol{q}_i \cdot \boldsymbol{k}_j$ only when $i \leq j$)
- we need to get rid of (mask out) the upper-triangular entries of $\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}$

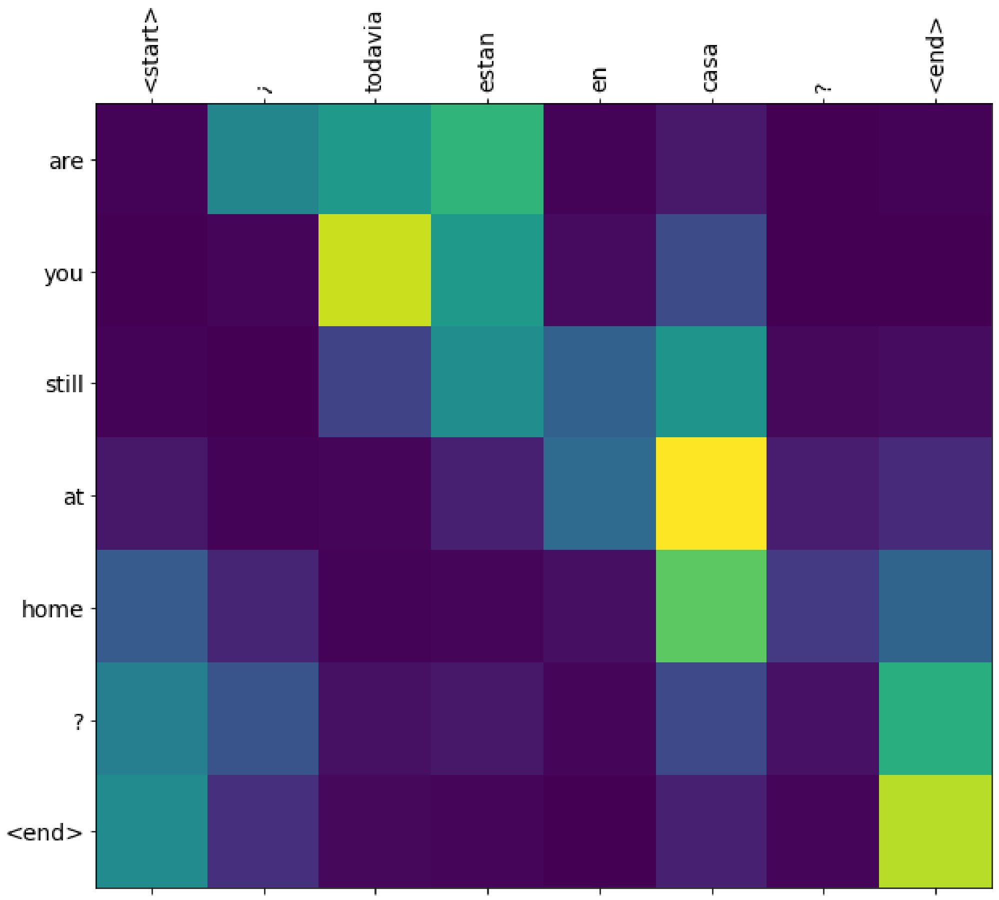
self-attention in transformer

- we need to get rid of (mask out) the upper-triangular entries of $\boldsymbol{QK}^{\mathrm{T}}$

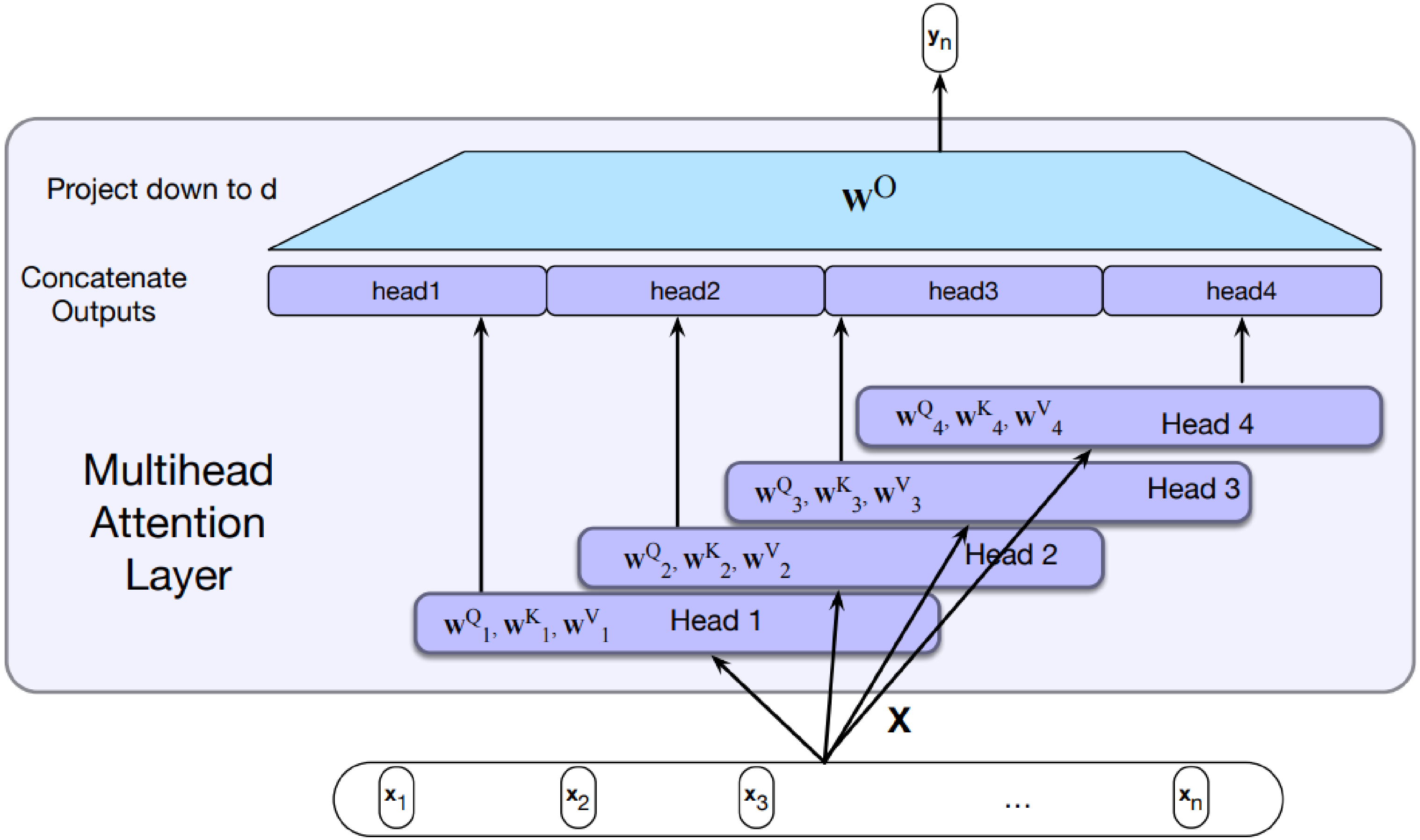
	q1•k1	$-\infty$	$-\infty$	$-\infty$	$-\infty$
	q2•k1	q2•k2	$-\infty$	$-\infty$	$-\infty$
N	q3•k1	q3•k2	q3•k3	$-\infty$	$-\infty$
	q4•k1	q4•k2	q4•k3	q4•k4	$-\infty$
	q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N

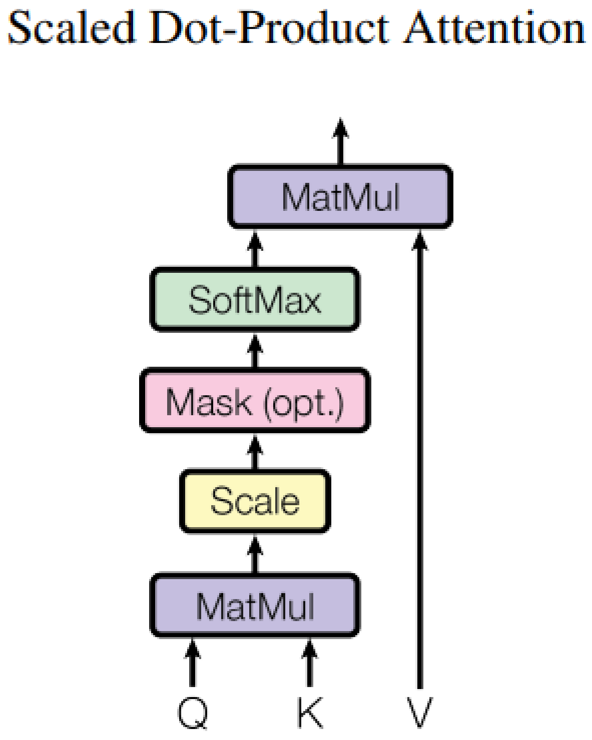
an example of attention weights



multi-head attention



multi-head attention



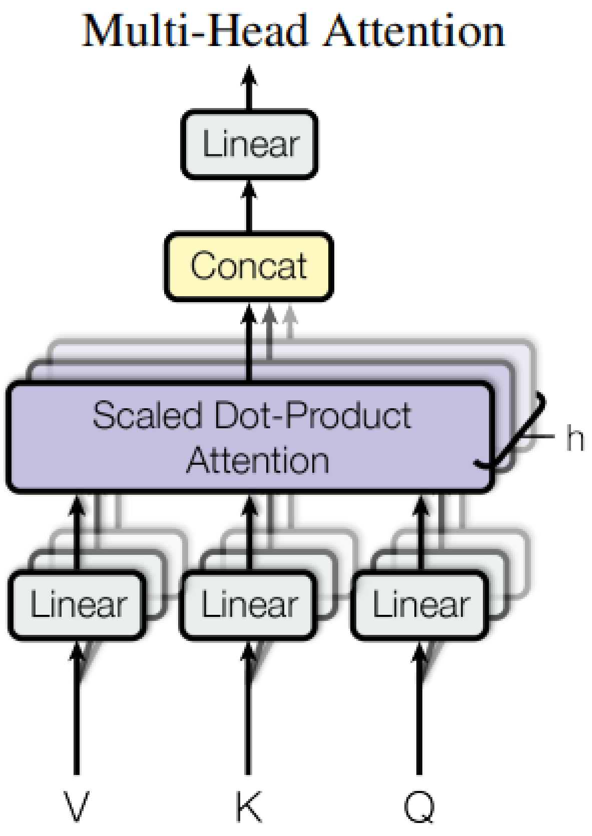
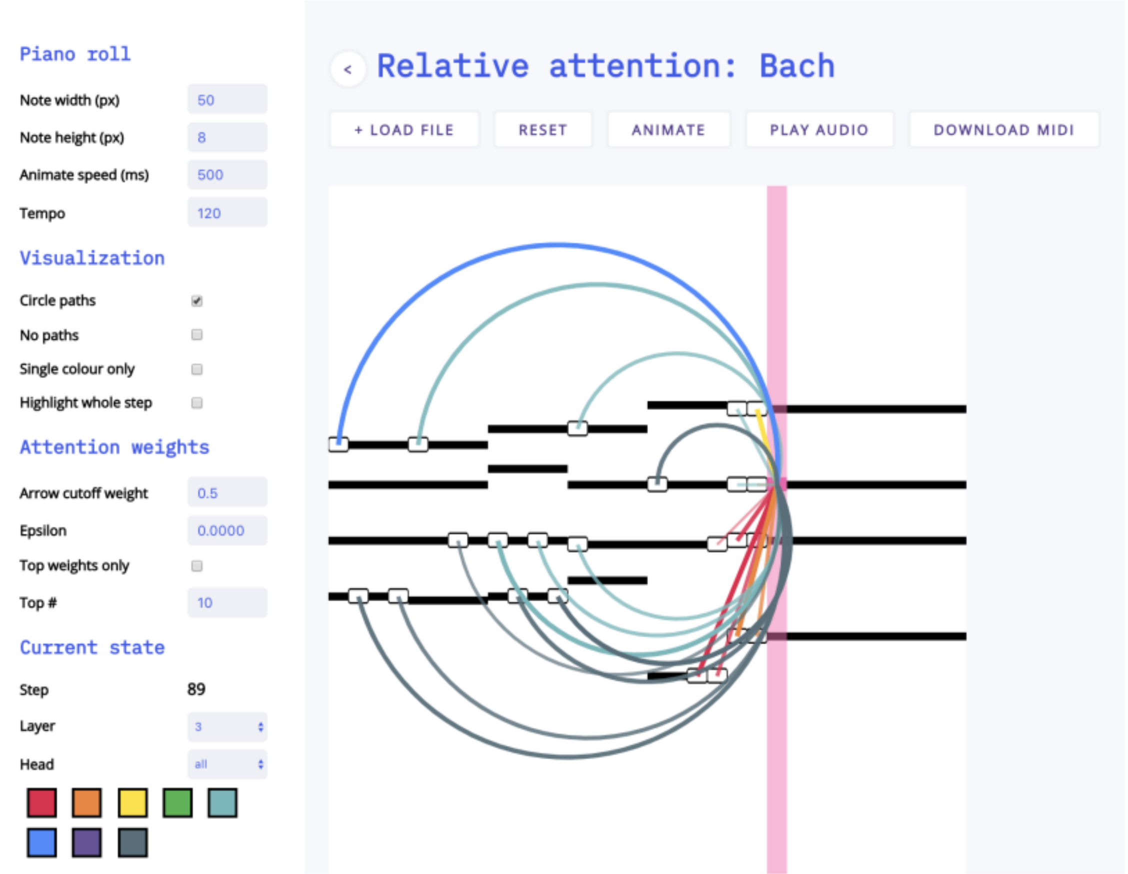


image credit: Vaswani, A. et al (2017). Attention is all you need.

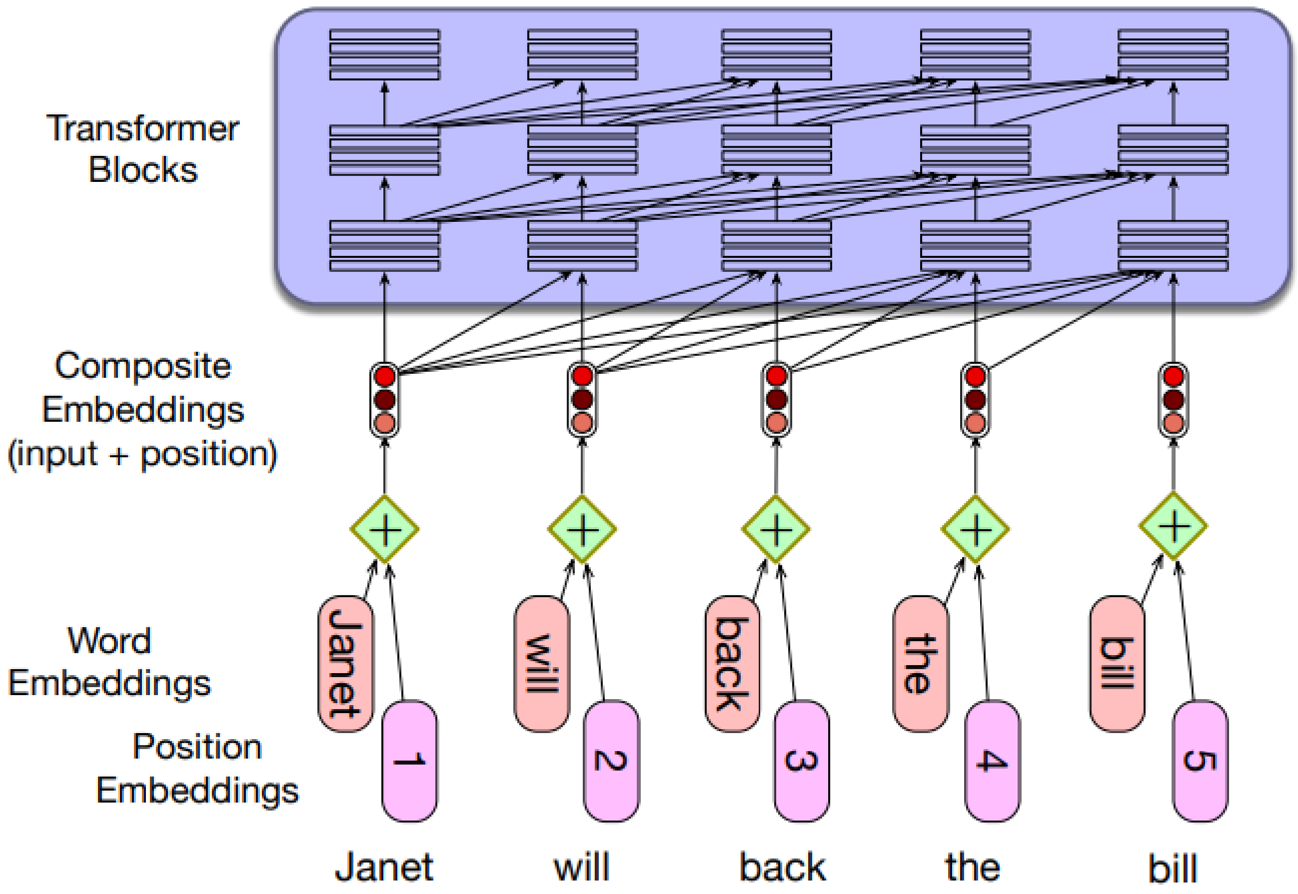
another example

- https://storage.googleapis.com/nips-workshop-visualization/index.html



7.2 transformer

positional embedding (encoding)



positional embedding (encoding)

- transformers process input data in parallel, which means they don't inherently understand the order or position of elements in a sequence
- positional embedding refers to the technique used to inject information about the position of words or tokens in a sequence into the model
- create a matrix where each row corresponds to a position in the sequence; these embeddings are then added element-wise to the input embeddings

positional embedding (encoding)

- one popular choice

$$\vec{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$

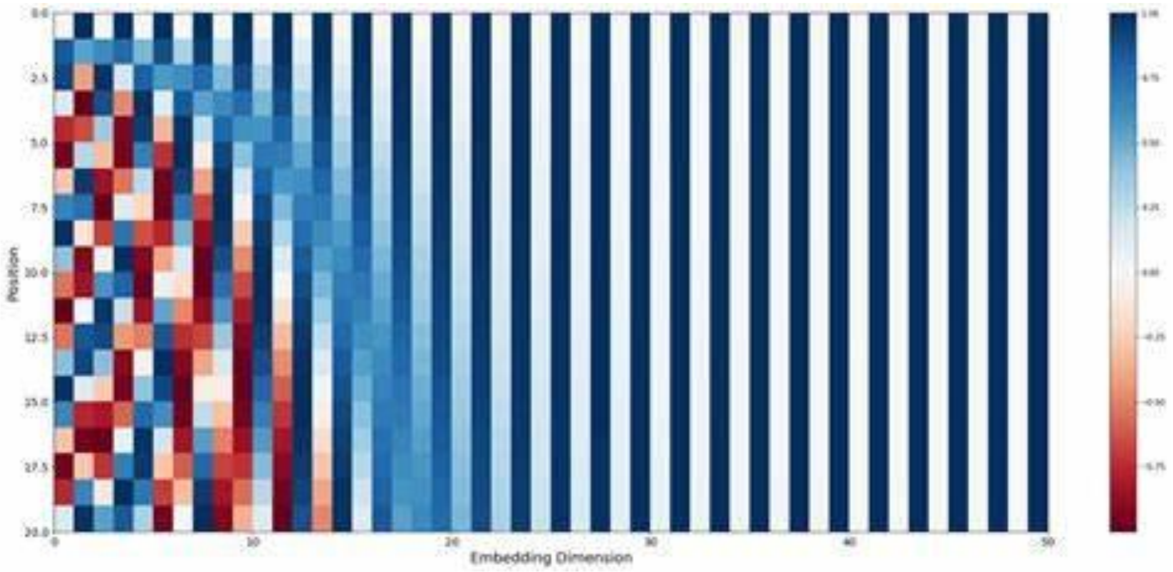
dimension

position

positional embedding (encoding)

- one popular choice

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$



positional embedding (encoding)

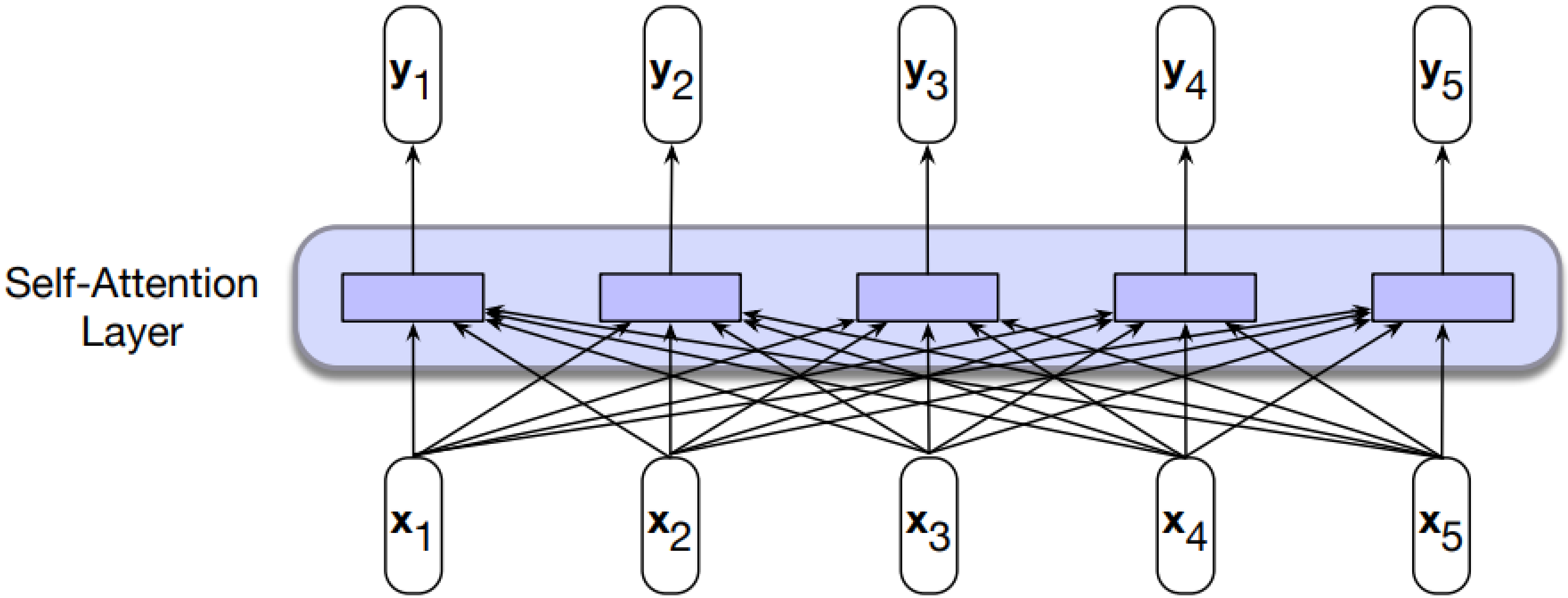
- it is also possible to encode relative positions of pairs of input
- it is also possible to encode positions for query and key

$$\mathrm{RoPE}(q)[2i] = q[2i] \cdot \cos(\theta_i) - q[2i+1] \cdot \sin(\theta_i)$$

$$\mathrm{RoPE}(q)[2i+1] = q[2i] \cdot \sin(\theta_i) + q[2i+1] \cdot \cos(\theta_i)$$

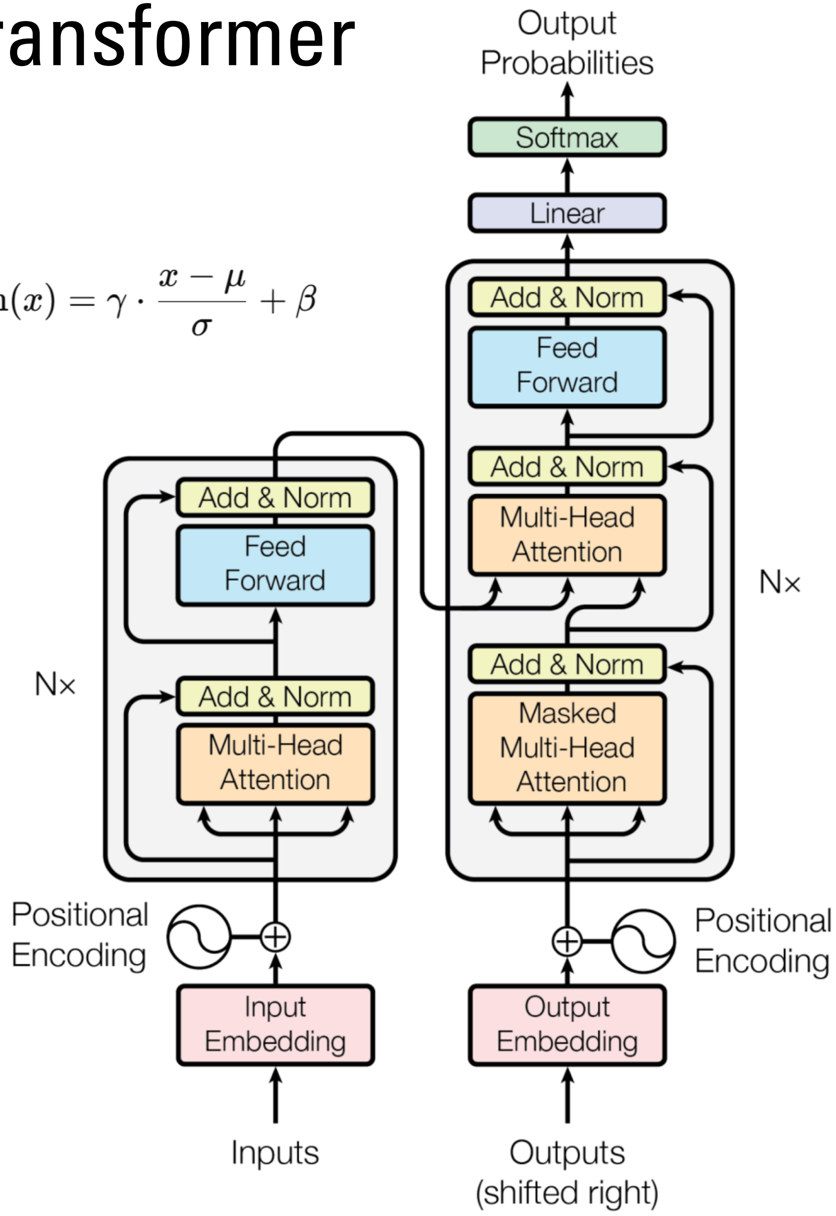
$$\theta_i = p/10000^{2i/d} \quad \text{(like classic sinusoidal freq)}$$

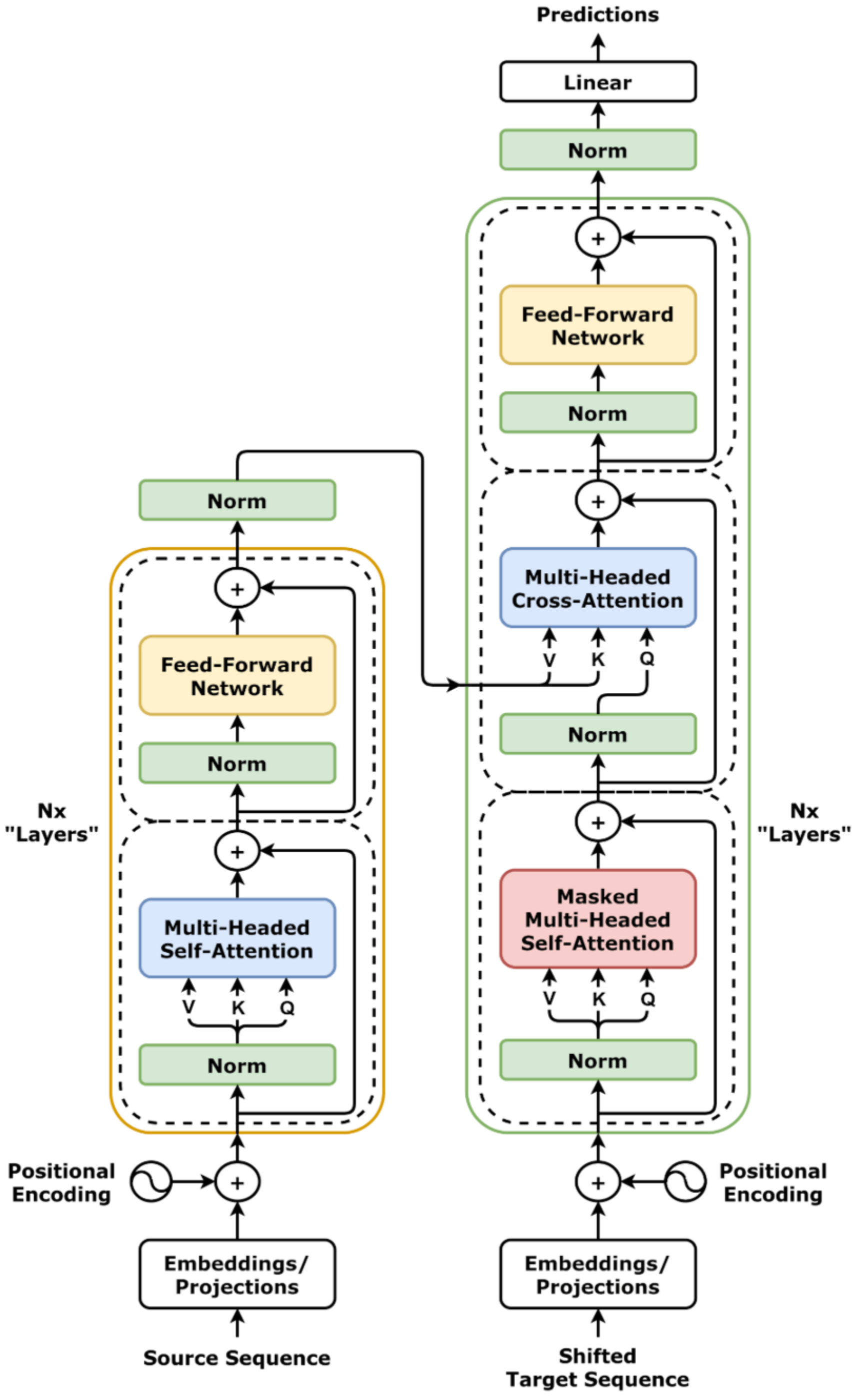
non-causal attention is also possible



transformer

$$\mathrm{LayerNorm}(x) = \gamma \cdot \frac{x - \mu}{\sigma} + \beta$$



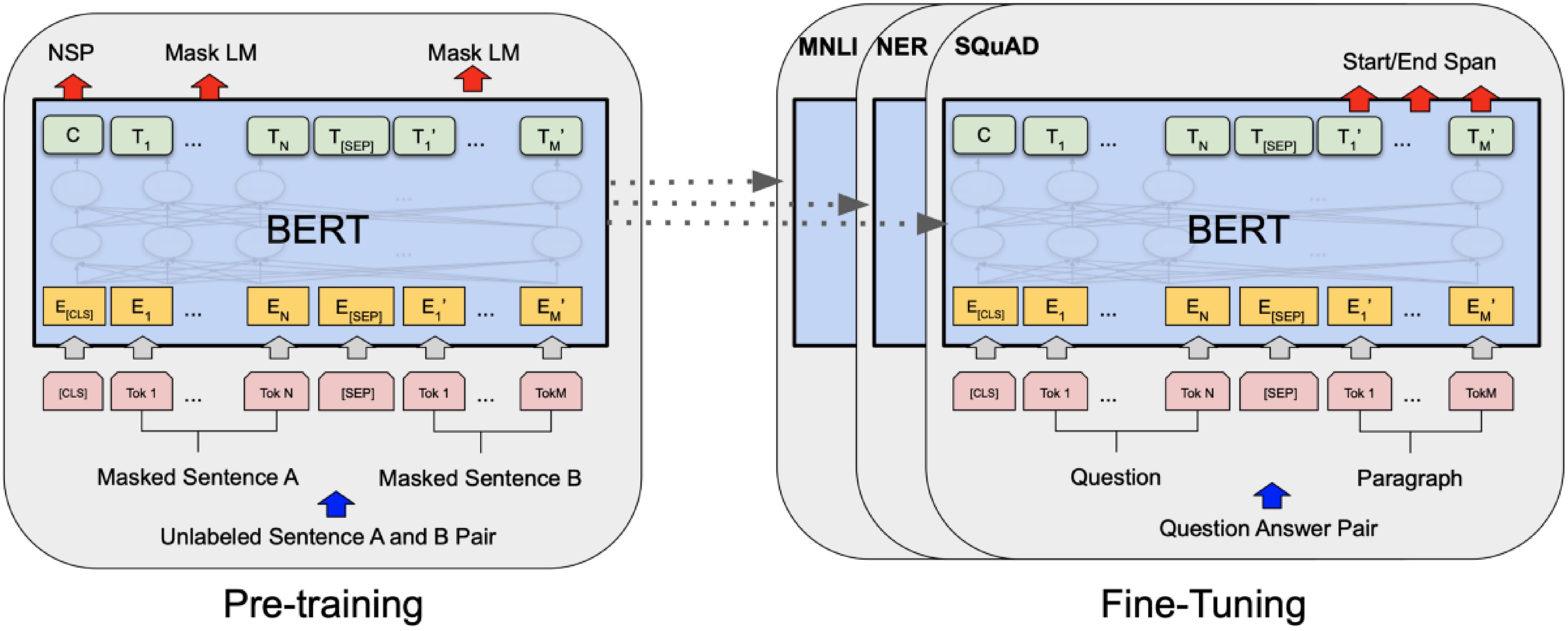


- AnnotatedTransformer.ipynb - Colab

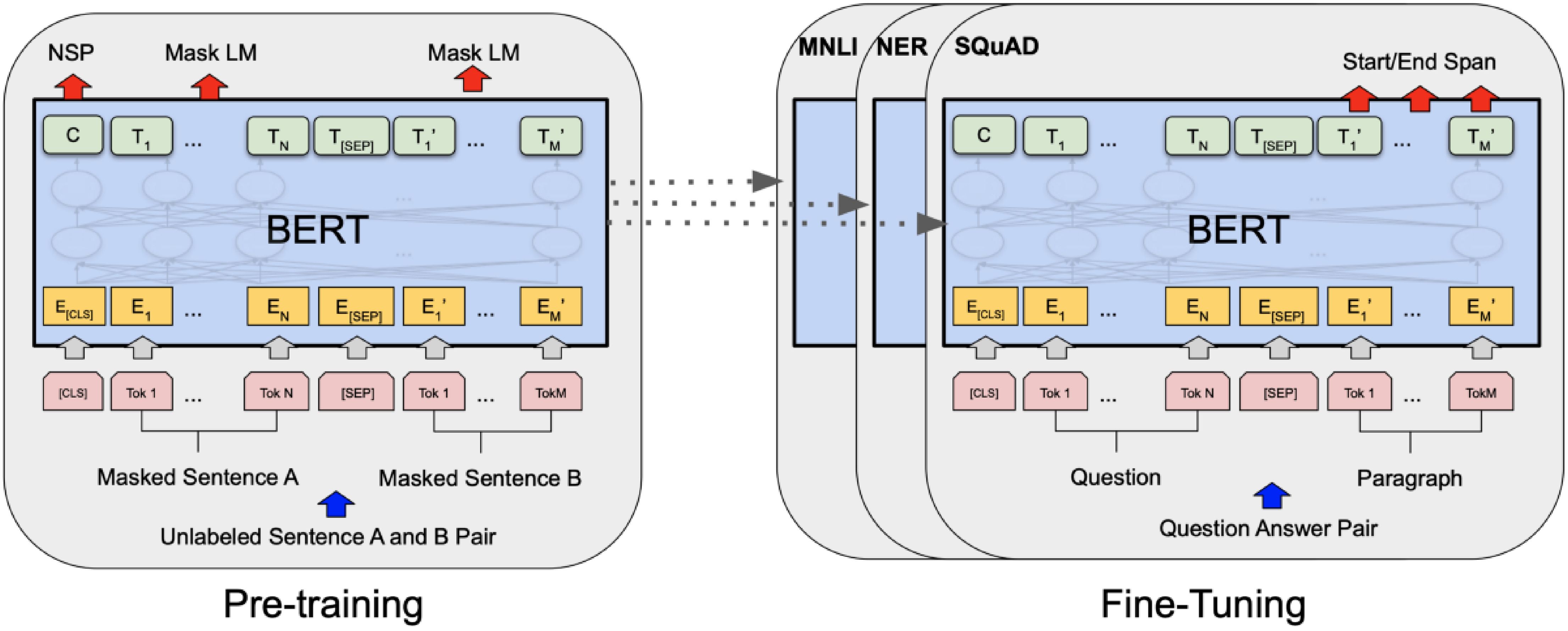
advantage of transformers

- parallelization and efficiency
- long-range dependencies
- reduced vanishing gradient problem
- interpretability
- …

example: BERT (Bidirectional Encoder Repr. fr. Transformers)



example: BERT (Bidirectional Encoder Repr. fr. Transformers)

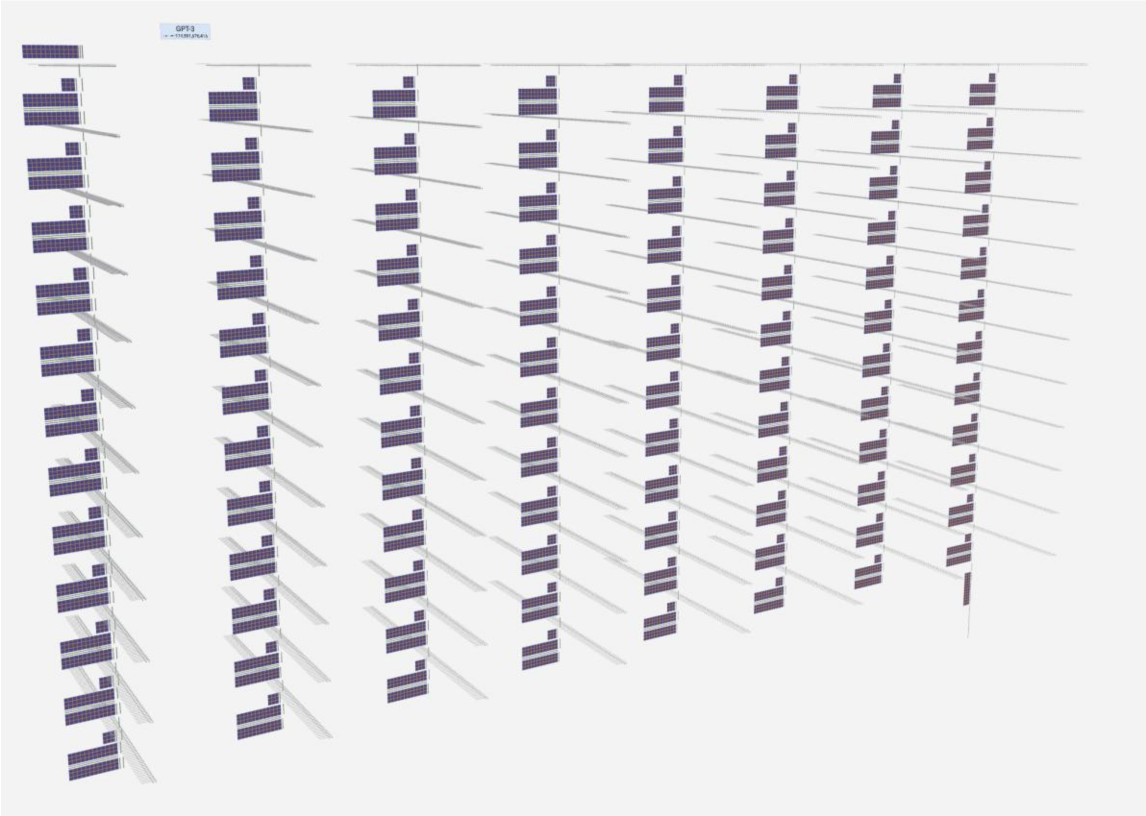


It uses only the encoder part of the transformer.

Not "causal": bi-directional.

example: GPT (Generative Pretrained Transformer)

- decoder-only



Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

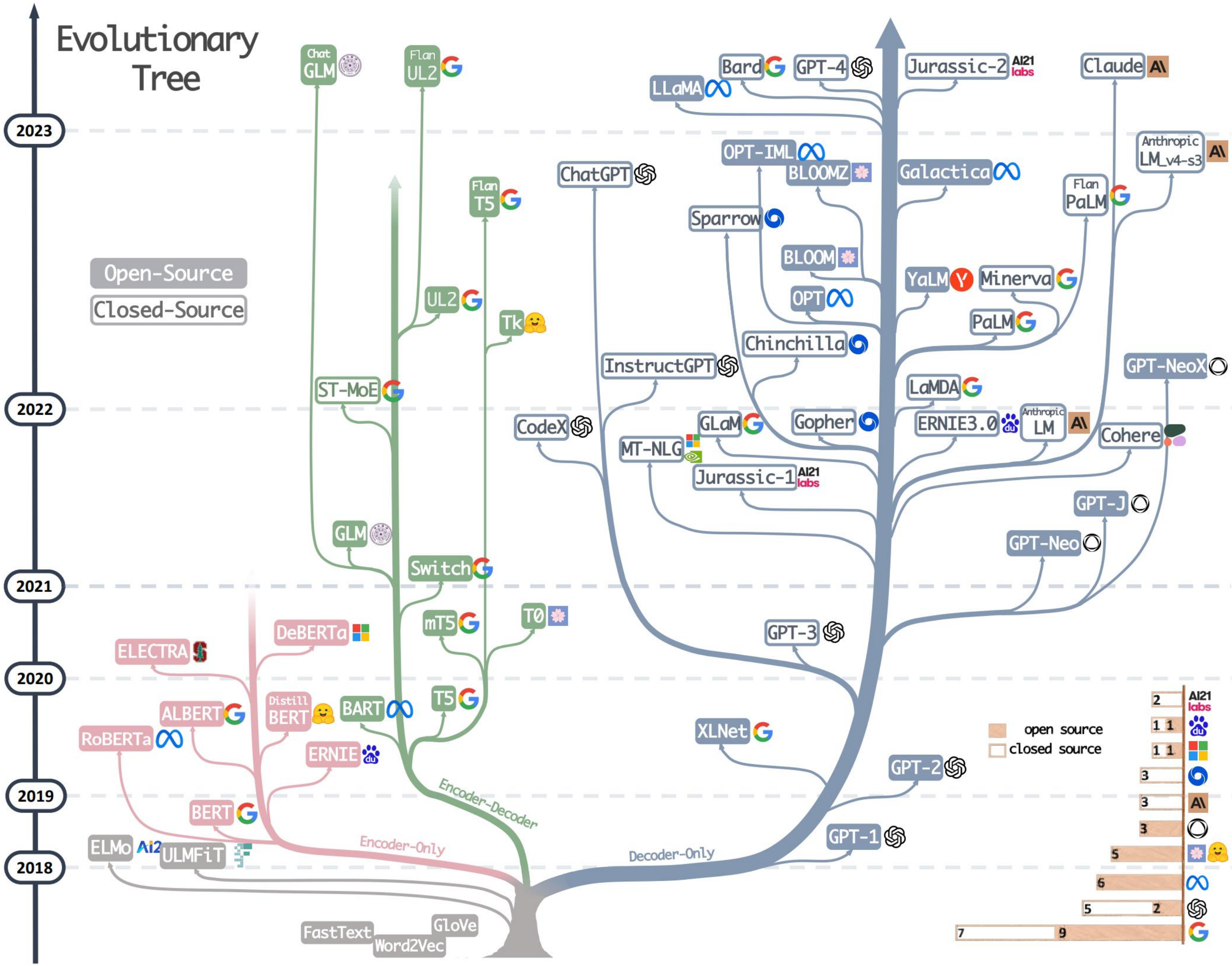
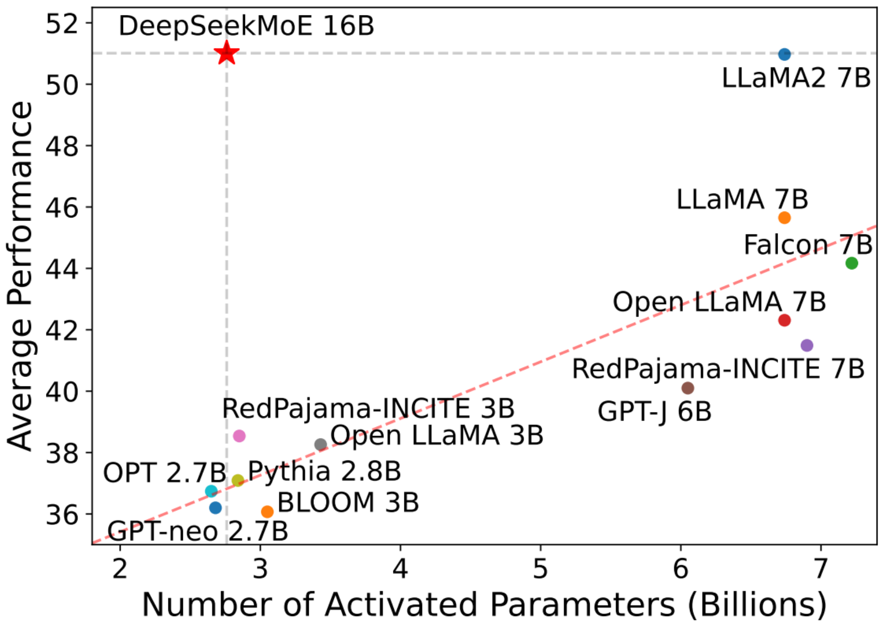
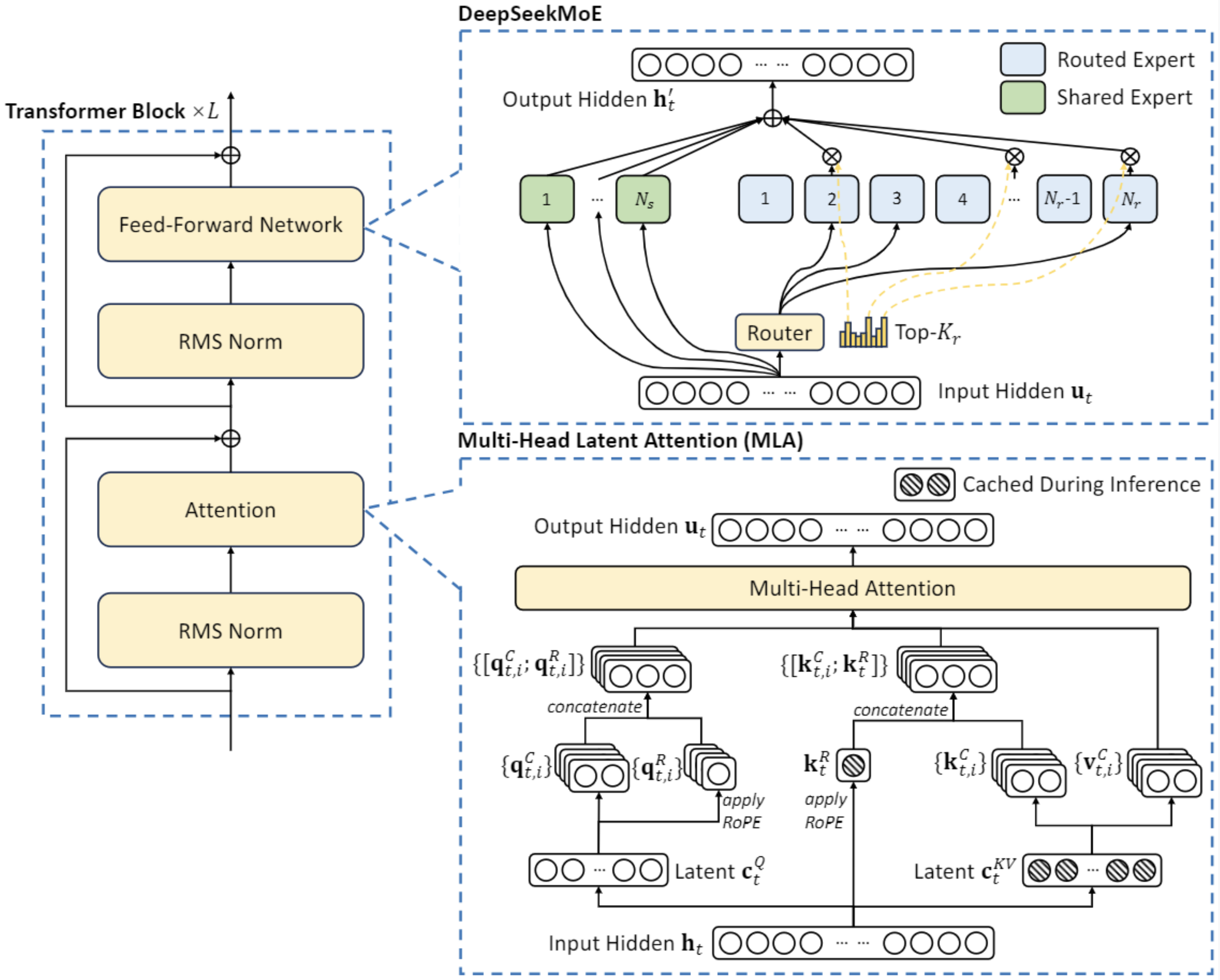


image credit: Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., ... & Hu, X. (2024)

example: DeepSeek

- decoder-only

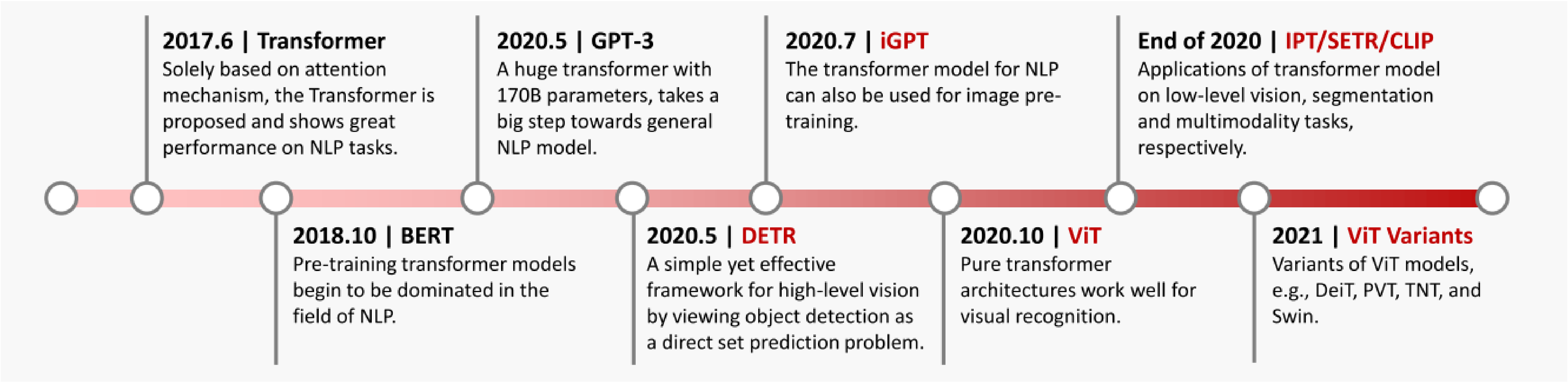




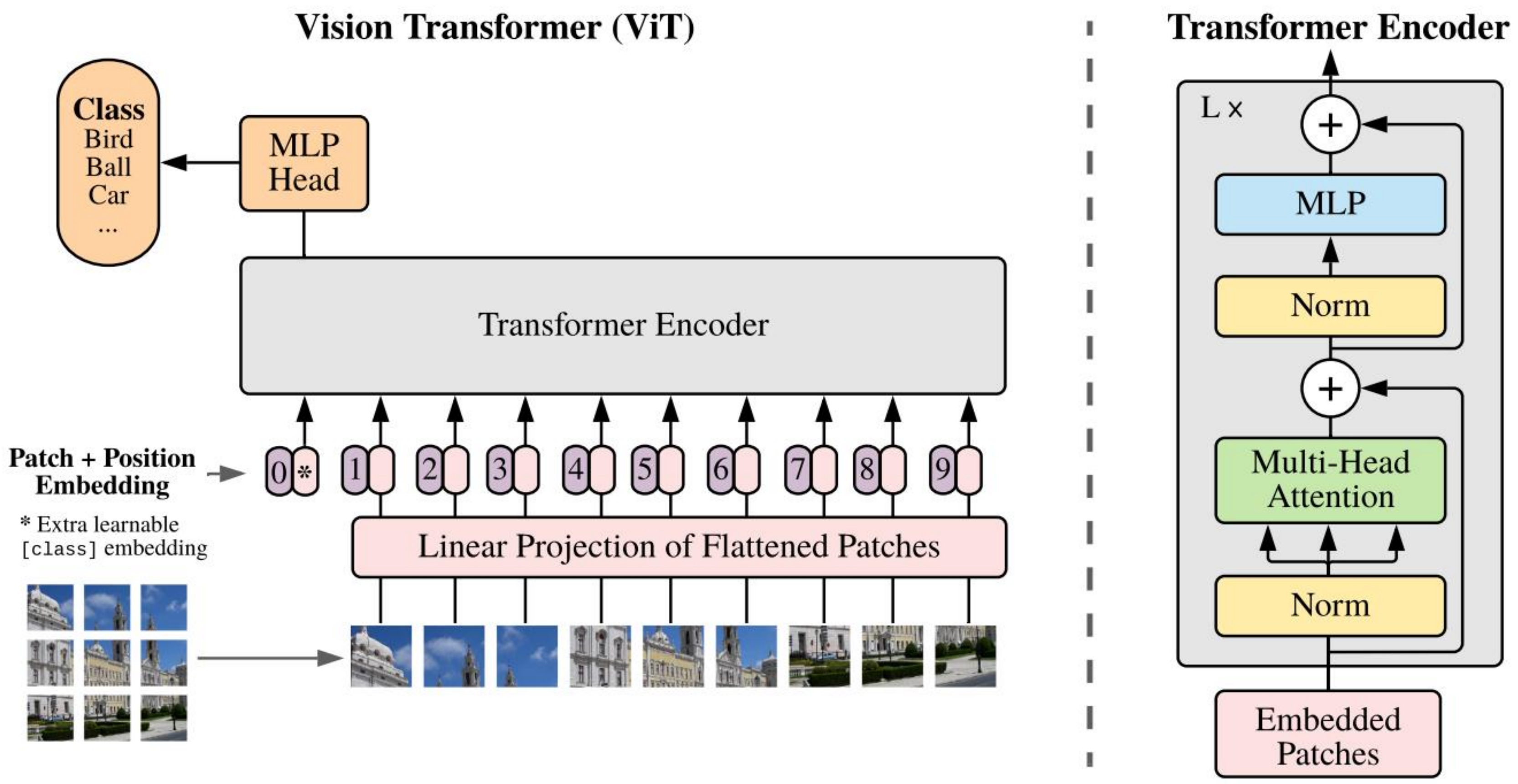
https://arxiv.org/pdf/2401.06066

7.3 vision transformer

a history


2017.6 | Transformer
Solely based on attention mechanism, the Transformer is proposed and shows great performance on NLP tasks.
2018.10 | BERT
Pre-training transformer models begin to be dominated in the field of NLP.
2020.5 | GPT-3
A huge transformer with 170B parameters, takes a big step towards general NLP model.
2020.5 | DETR
A simple yet effective framework for high-level vision by viewing object detection as a direct set prediction problem.
2020.7 | iGPT
The transformer model for NLP can also be used for image pre-training.
2020.10 | ViT
Pure transformer architectures work well for visual recognition.
End of 2020 | IPT/SETR/CLIP
Applications of transformer model on low-level vision, segmentation and multimodality tasks, respectively.
2021 | ViT Variants
Variants of ViT models, e.g., DeiT, PVT, TNT, and Swin.

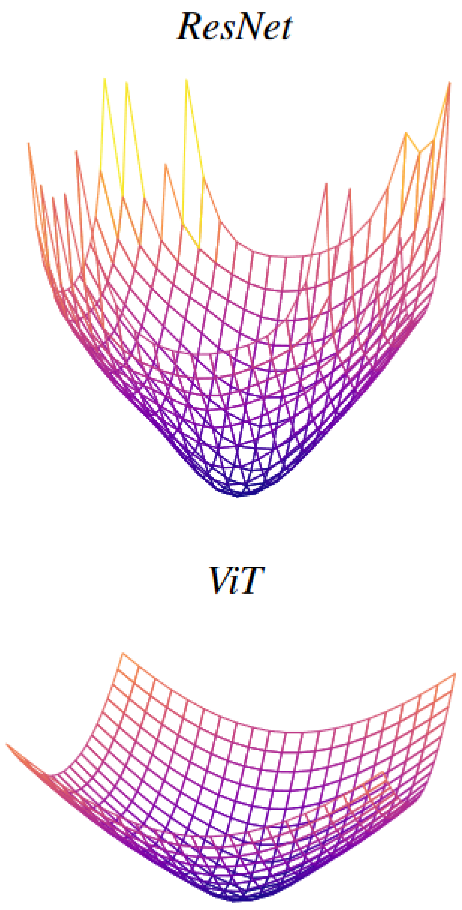
vision transformer



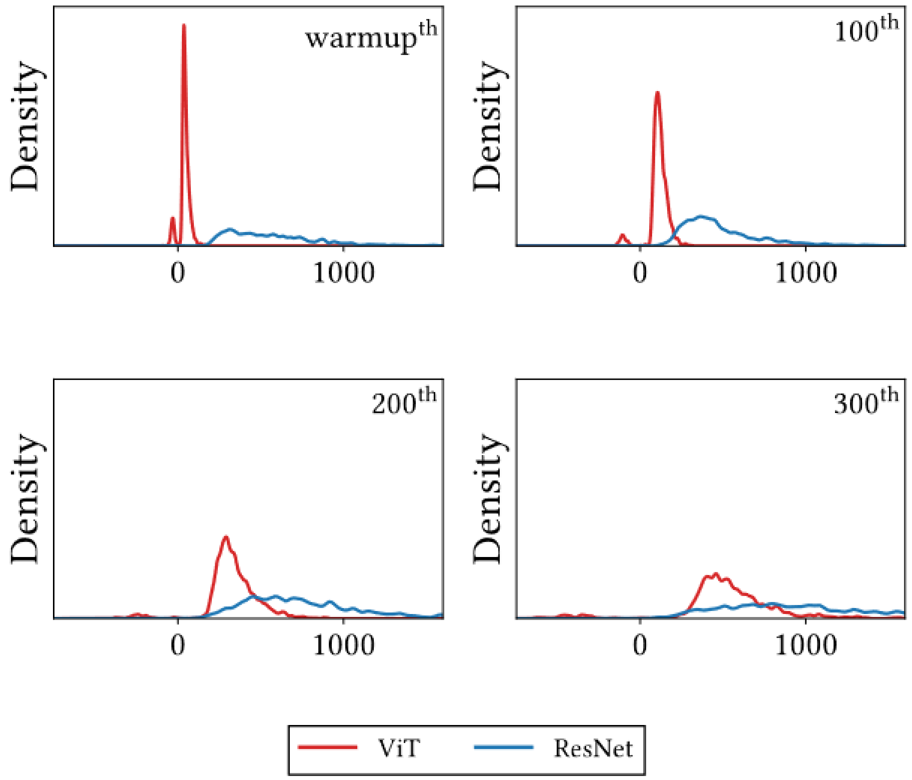
advantages of vision transformers

- global context understanding
- scalability
- parameter efficiency
- …

How does vision transformer work?



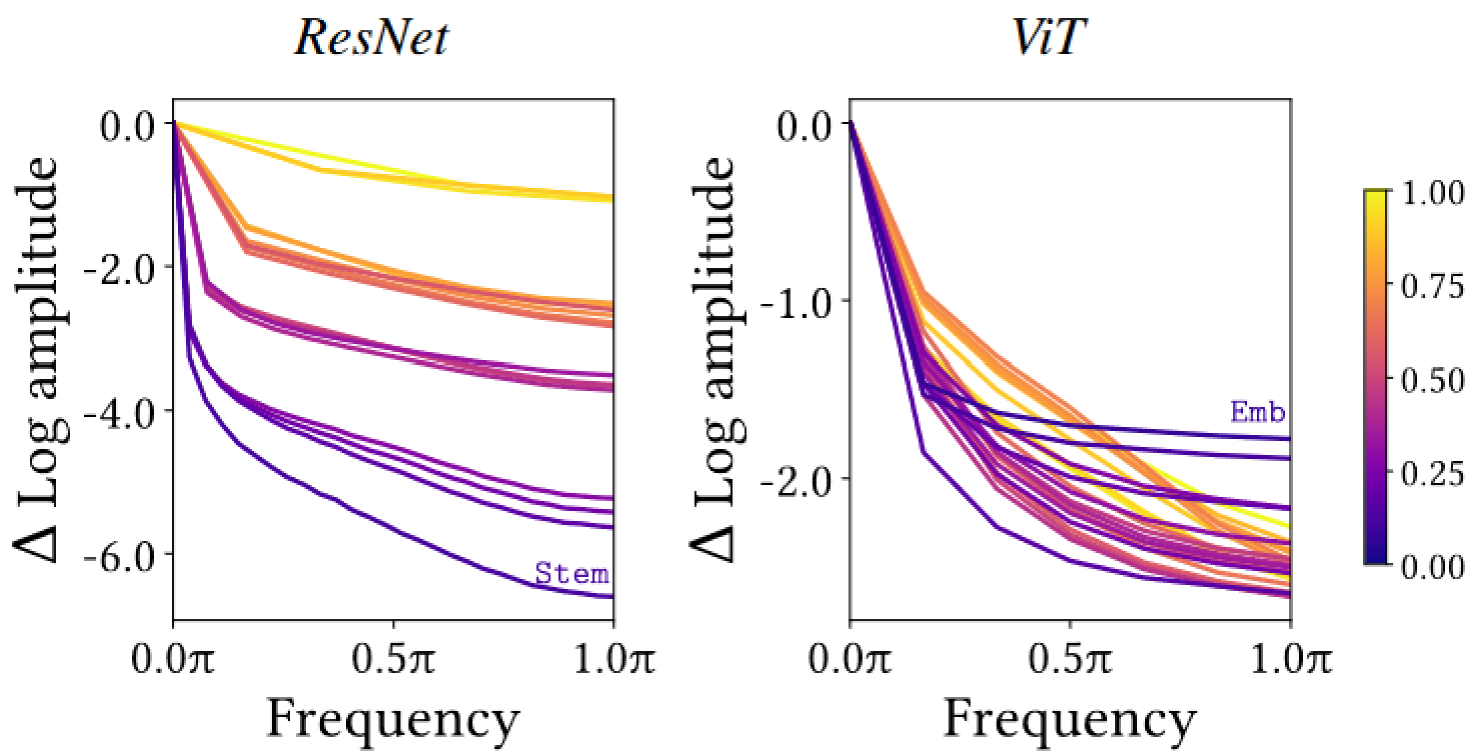
(a) Loss landscape visualizations



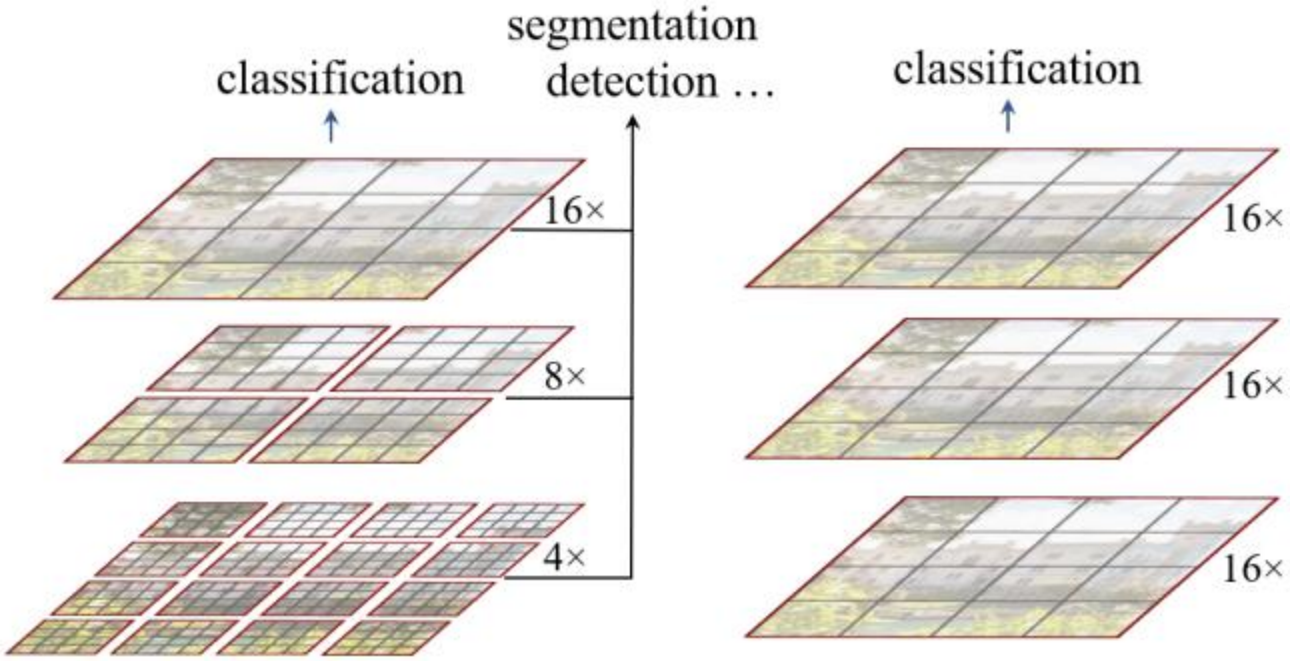
(b) Hessian max eigenvalue spectra

https://openreview.net/forum?id=D78Go4hVcxO

How does vision transformer work?

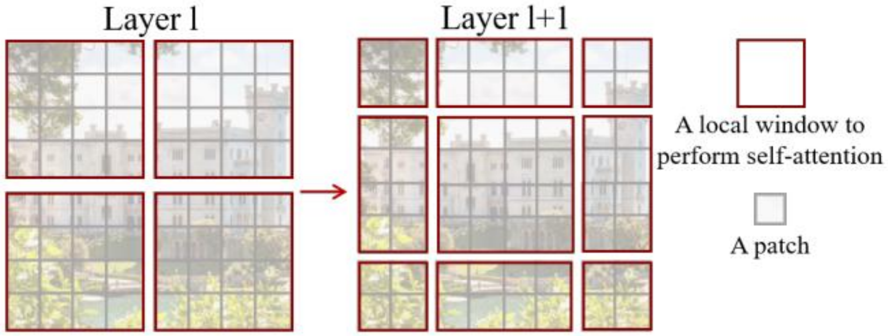


swin (shifted windows) transformer



(a) Swin Transformer (ours)

(b) ViT



swin (shifted windows) transformer

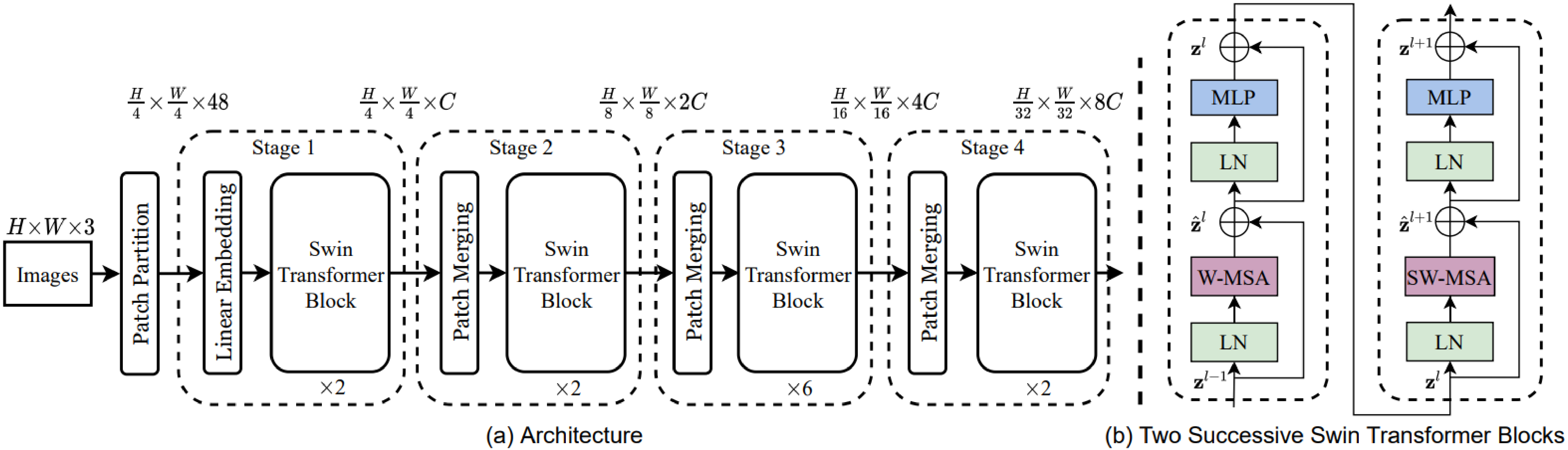


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

Thank you!

Reference

- Ch 10, 11 & 16, Jurafsky, D., & Martin, J. H. (2023). *Speech and language processing (draft)*. https://web.stanford.edu/~jurafsky/slp3/
- Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57, 345-420.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Huang, C. Z. A., Dinculescu, M., Vaswani, A., & Eck, D. (2018). Visualizing music transformer. In *NeurIPS Workshop on Interpretability and Robustness in Audio, Speech, and Language*.
- Kenton, J. D. M. W. C., & Toutanova, L. K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT* (pp. 4171-4186).
- Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., ... & Tao, D. (2022). A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1), 87-110.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR* 2021.
- Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 10012-10022).
- Cao, H., Wang, Y., Chen, J., Jiang, D., Zhang, X., Tian, Q., & Wang, M. (2022). Swin-UNet: Unet-like pure transformer for medical image segmentation. In *ECCV* 2022.