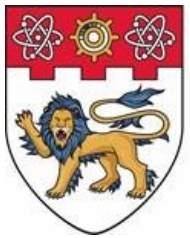


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

An Introduction of Transformer

A Survey of Transformers / *Efficient Transformers: A Survey*

Yige Xu

PhD Student, SCSE, NTU

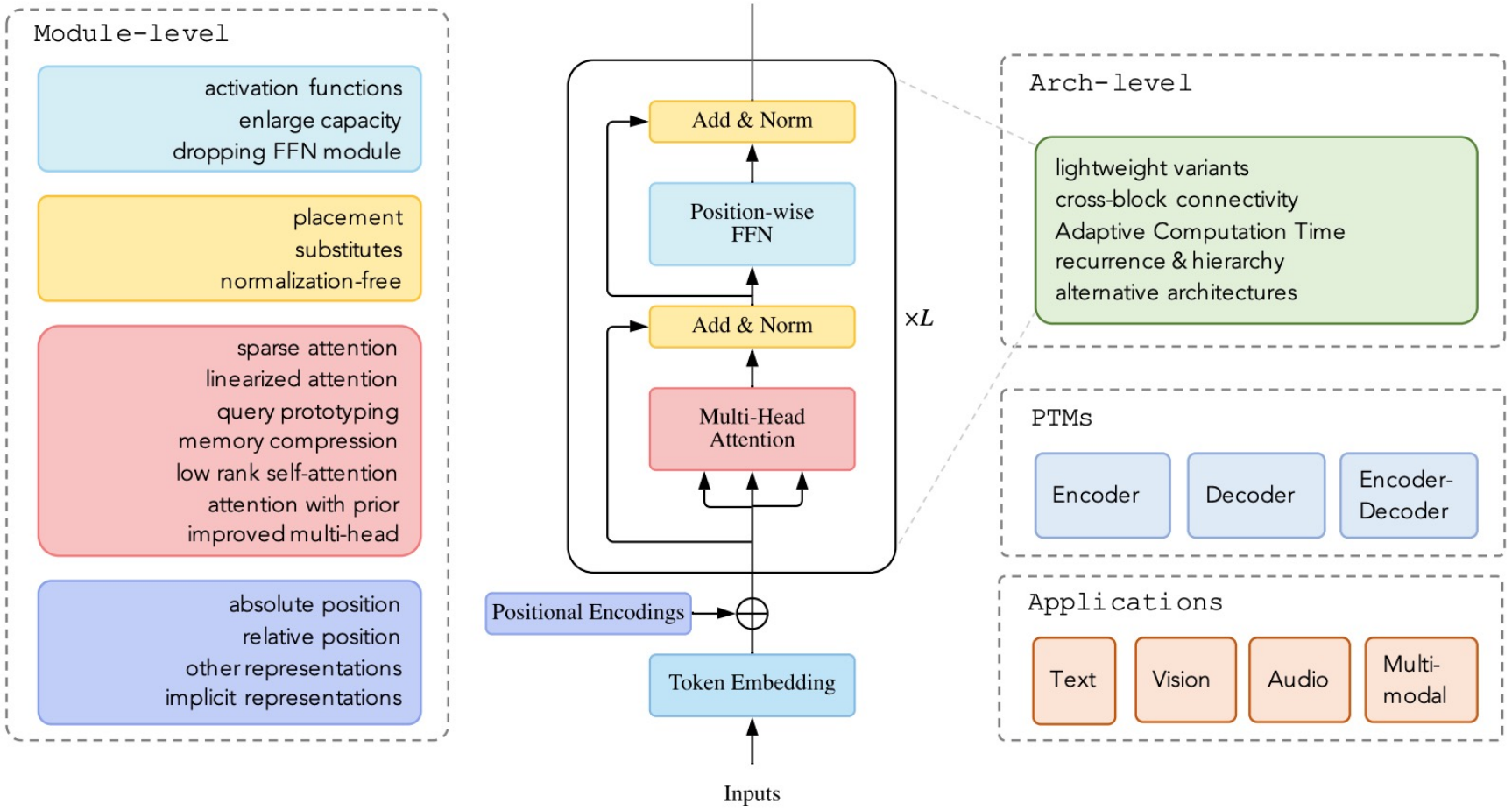
yige002@e.ntu.edu.sg

https://xuyige.github.io

Outlines

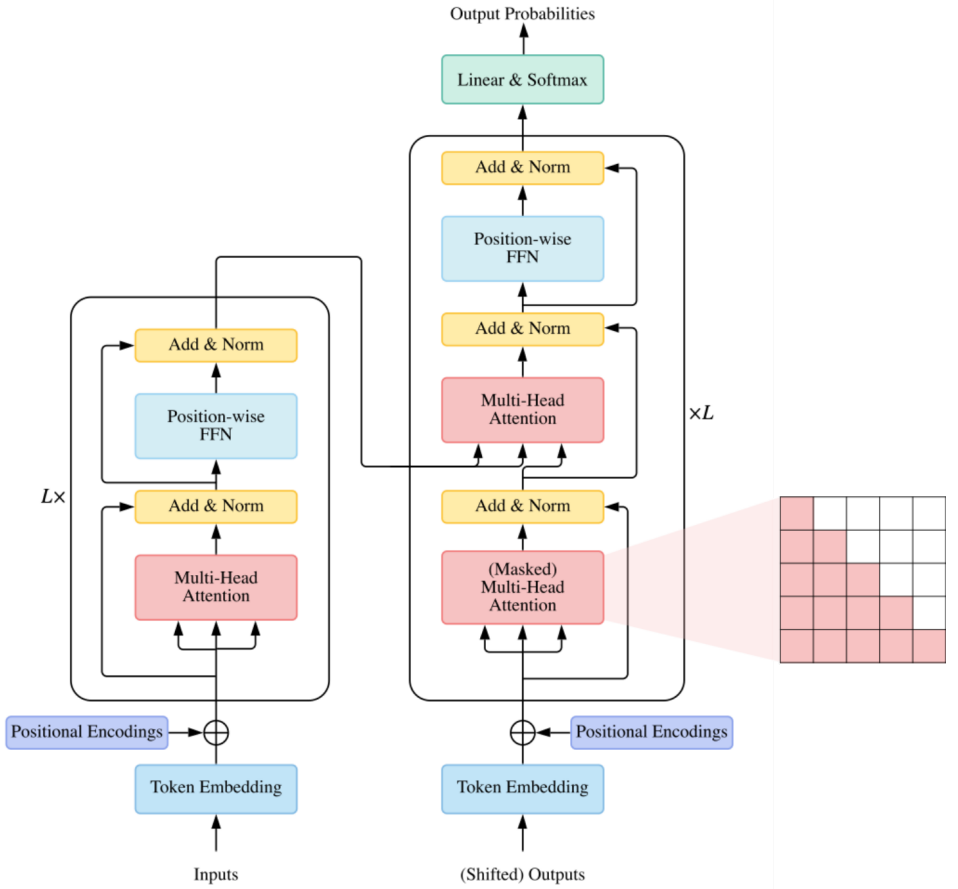
- Overview of Transformer
- Comparison with other modules
- X-formers
- Analysis of Transformer module

Overview of Transformer

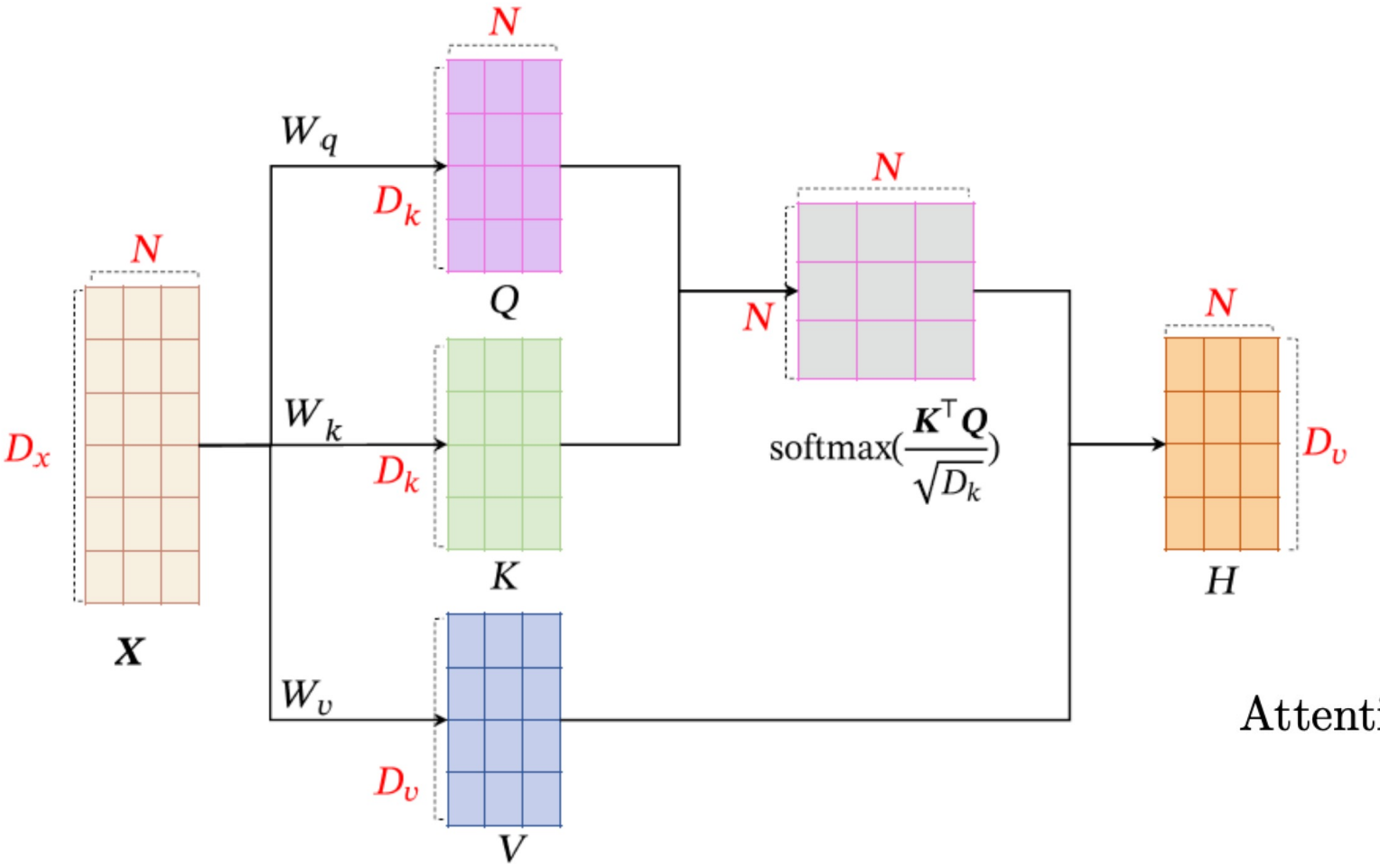


Overview of Transformer

- Broadly, Transformer is a model built with self-attention
  - Core module: Self-Attention
  - Besides self-attention:
    - Position representations
    - Residual Connection and Normalization
    - Skip connection
    - Position-wise FFN



Overview of Transformer

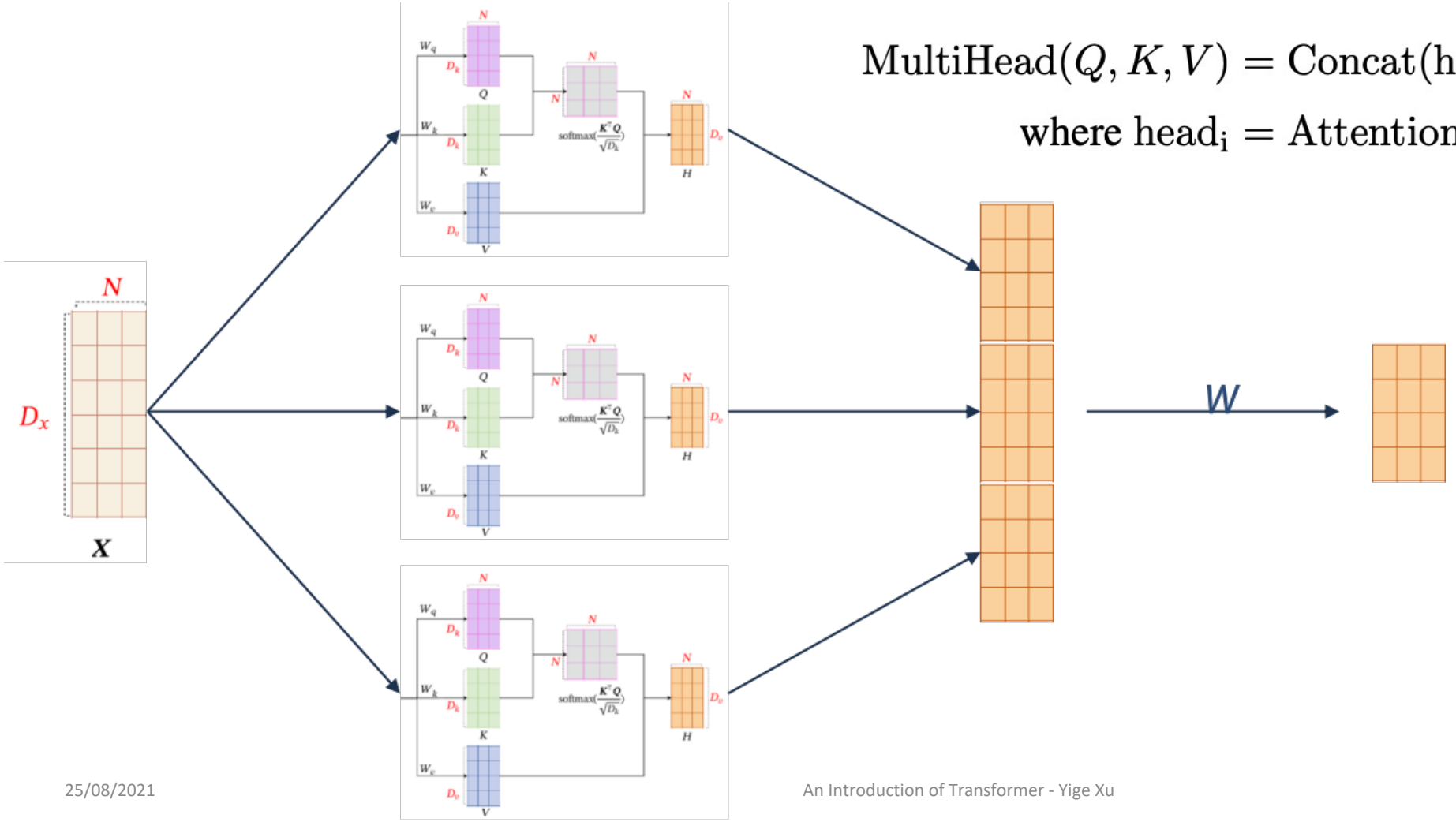


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Overview of Transformer

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



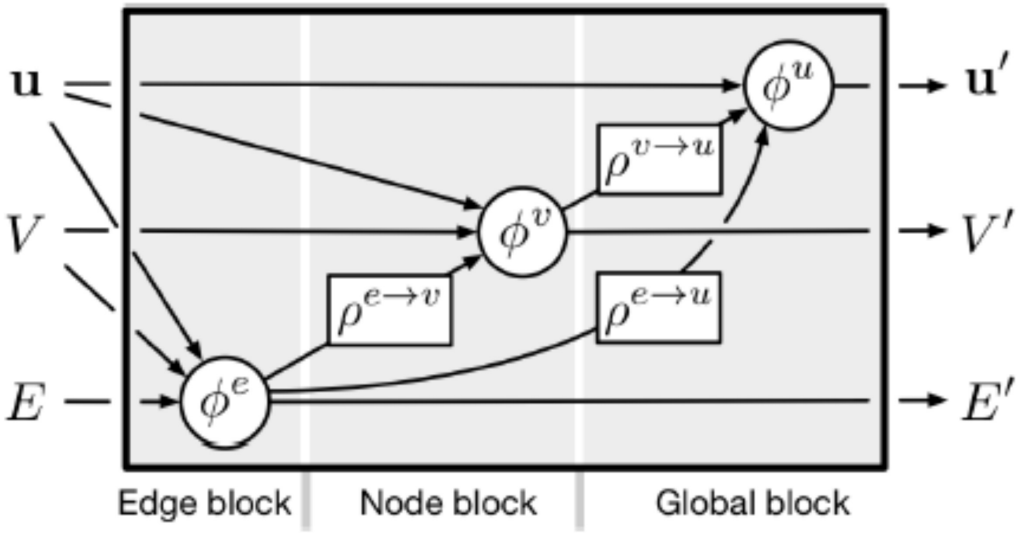
Comparison with other modules

- Inductive Bias of Transformer:
  - Fully-connected structure
  - No prior assumption
  - Use position embedding to model the sequential information
  - Computational Complexity: $O(T^2D)$

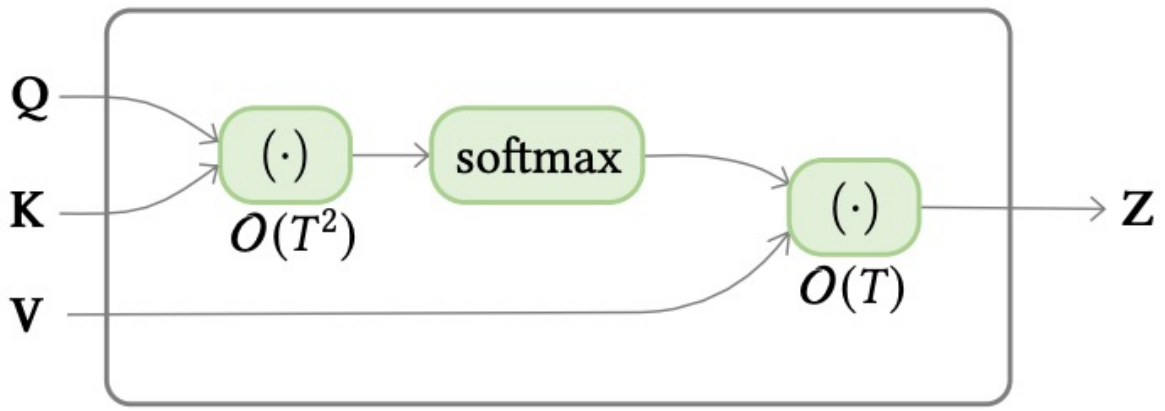
Table 1. Complexity and parameter counts of self-attention and position-wise FFN

Module	Complexity	#Parameters
self-attention	$O(T^2 \cdot D)$	$4D^2$
position-wise FFN	$O(T \cdot D^2)$	$8D^2$

Comparison with other modules



(a) Full GN block



(a) standard self-attention

Comparison with other modules

Table 2. Per-layer complexity, minimum number of sequential operations and maximum path lengths for different layer types. T is the sequence length, D is the representation dimension and K is the kernel size of convolutions [137].

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(T^2 \cdot D)$	$O(1)$	$O(1)$
Fully Connected	$O(T^2 \cdot D^2)$	$O(1)$	$O(1)$
Convolutional	$O(K \cdot T \cdot D^2)$	$O(1)$	$O(\log_K(T))$
Recurrent	$O(T \cdot D^2)$	$O(T)$	$O(T)$

Comparison with other modules

- Convolutional networks
  - Translation invariance (shared kernel across spatial positions)
  - Locality (restricted window)
- Recurrent networks
  - Temporal invariance (shared function across timesteps)
  - Locality (Markovian structure)
- Transformer
  - No structural prior (prone to overfitting in small-scale data)
  - Permutation equivariance (requires position representations to encode sequences)

Comparison with other modules

- Transformer is more parallelizable than recurrent layers
- Transformer has global receptive field, thus doesn't require stacking layers to model global dependencies (like convolutional layers).
- The bottleneck of the Transformer network lies in FFN for short inputs, and self-attention for long inputs.
- Transformer vs. GNN:
  - Transformer is a GNN defined over complete directed graphs (w/ self-loop)
  - No explicit structure. Message passing depends solely on similarity measures over contents.

Comparison with other modules

- Disadvantages:
  - Can hardly model long input
  - Sometimes overfitting
- Directions:
  - Improve model structure: (1) inductive bias; (2) task-specific structure
  - Introduce external knowledge: pre-training and then fine-tuning

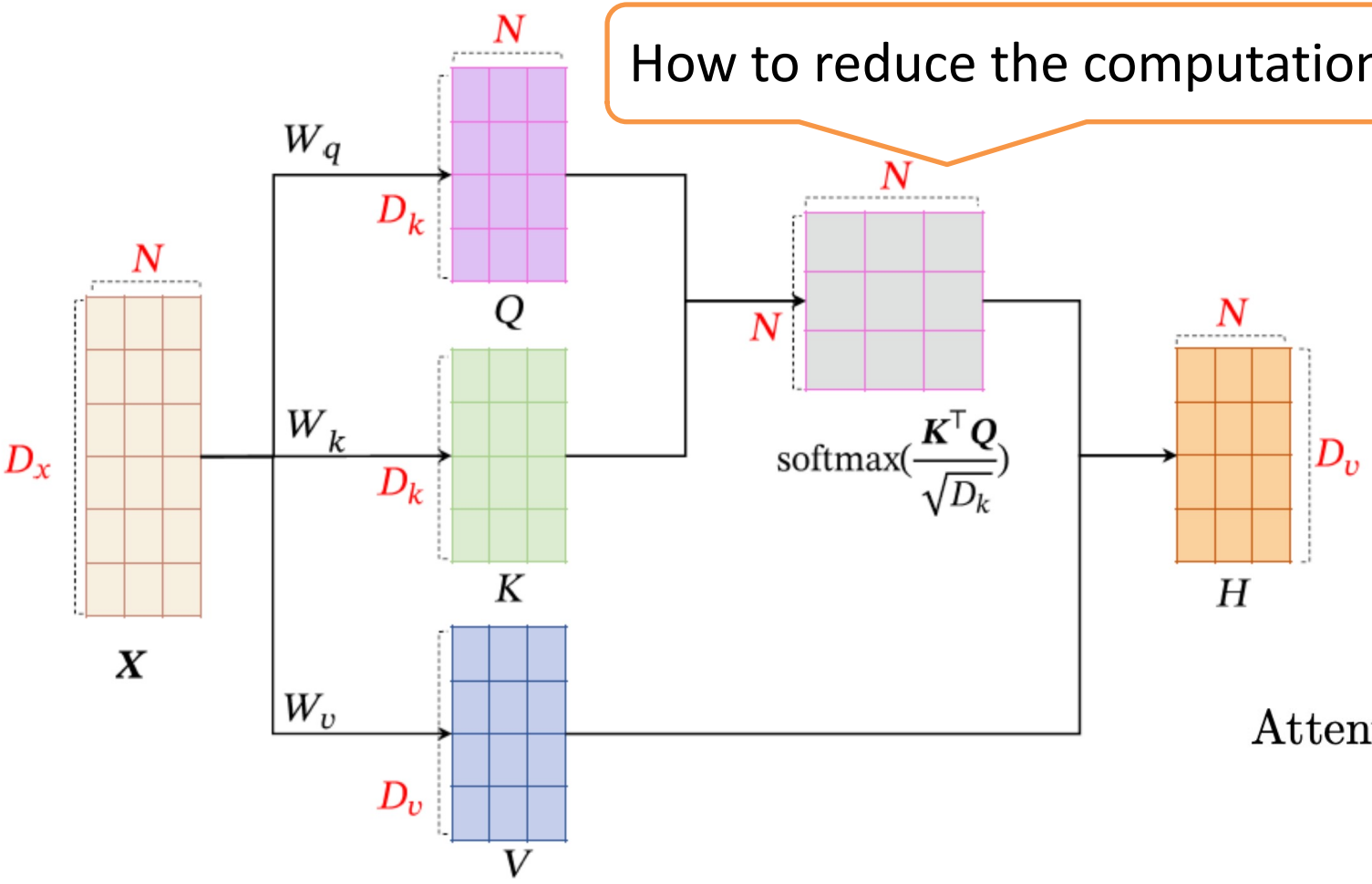
X-formers

2017	2018	2019	2020	2021
Transformer	Image Transformer	Star-Transformer	ETC	RFA
	Memory Compressed Attention	Sparse Transformer	Longformer	DPFP
	Local Transformer	BP-Transformer	BigBird	Informer
	Average Attention	Axial Transformer	Routing Transformer	Poolingformer
	Li et al., 2018	Set Transformer	Reformer	Luna
		Low-rank and locality constrained attention	SAC	Nyströmformer
		Gaussian Transformer	Sparse Sinkhorn Attention	LazyFormer
		Adaptive Attention Span	Linear Transformer	CAMTL
		Dynamic Routing	Performer	
			Clustered Attention	
			Linformer	
			CSALR	
			Predictive Attention Transformer	
			RealFormer	
			Hard-Coded Gaussian Attention	
			Synthesizer	
			Deshpande and Narasimhan, 2020	
			Talking-head Attention	
			Collaborative MHA	
			Multi-Scale Transformer	

- sparse attention (position based)
- sparse attention (content based)
- linearized attention
- query prototyping
- memory compression
- low-rank
- prior attention
- improved multi-head mechanism

X-formers

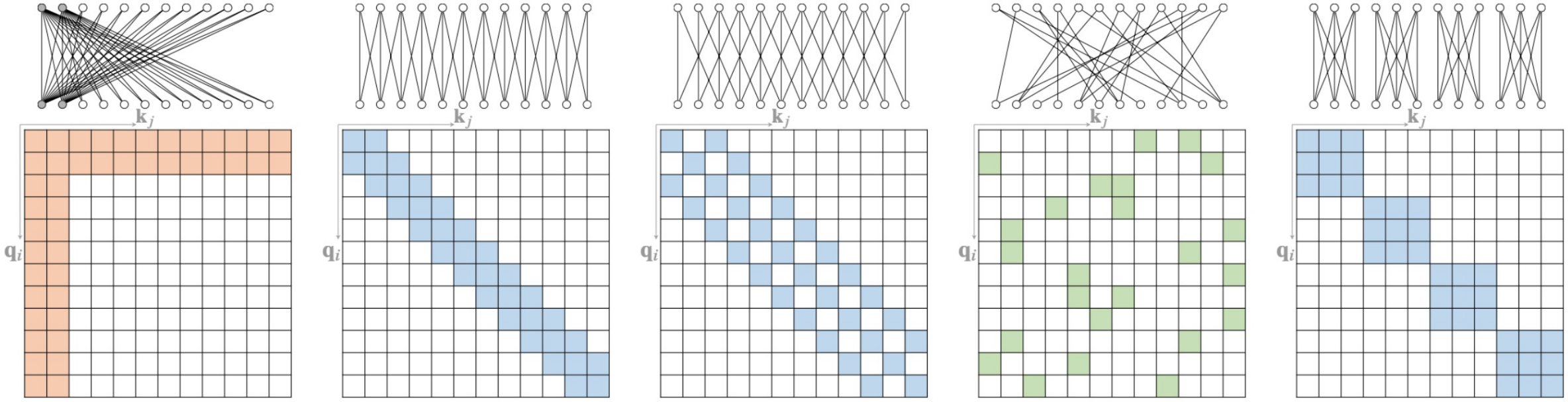
How to reduce the computational complexity?



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

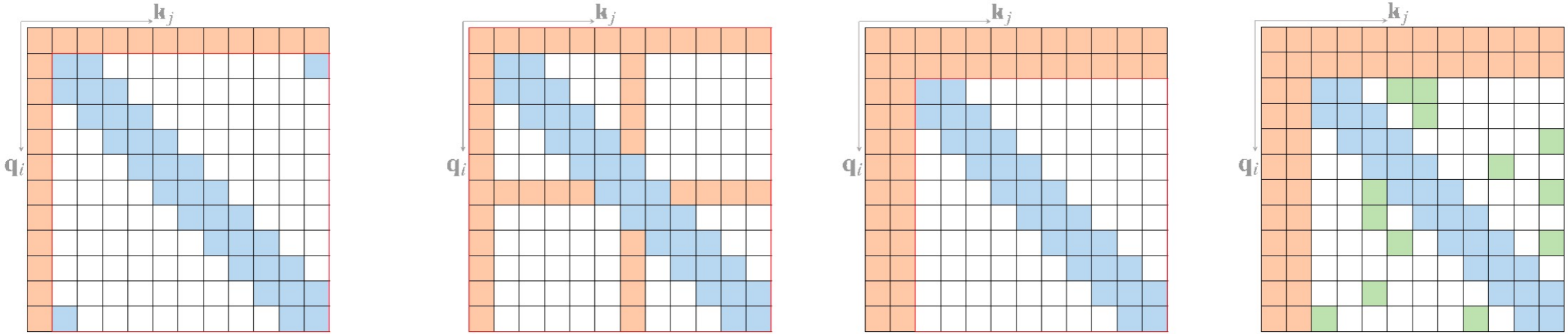
X-formers

- Position-based Sparse Attention



X-formers

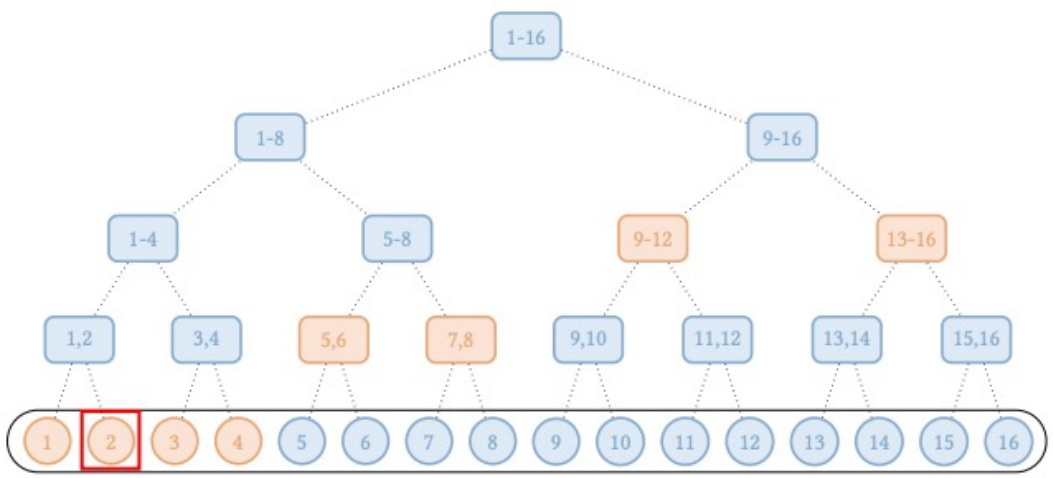
- Position-based Sparse Attention



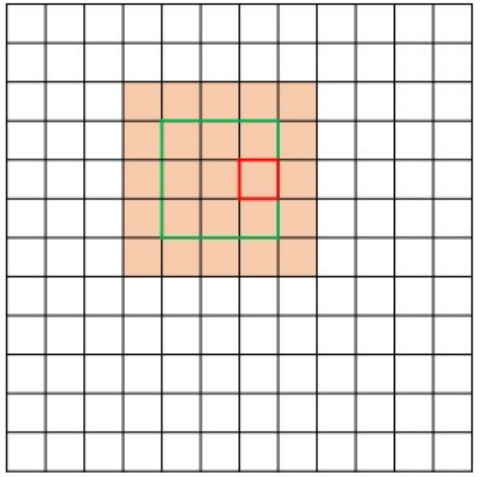
(a) Star-Transformer (b) Longformer (c) ETC (d) BigBird

X-formers

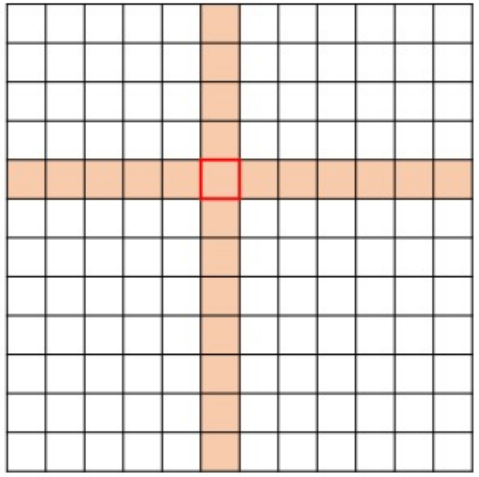
- Position-based Sparse Attention



(a) BPT



(b) block local (2D)



(c) axial (2D)

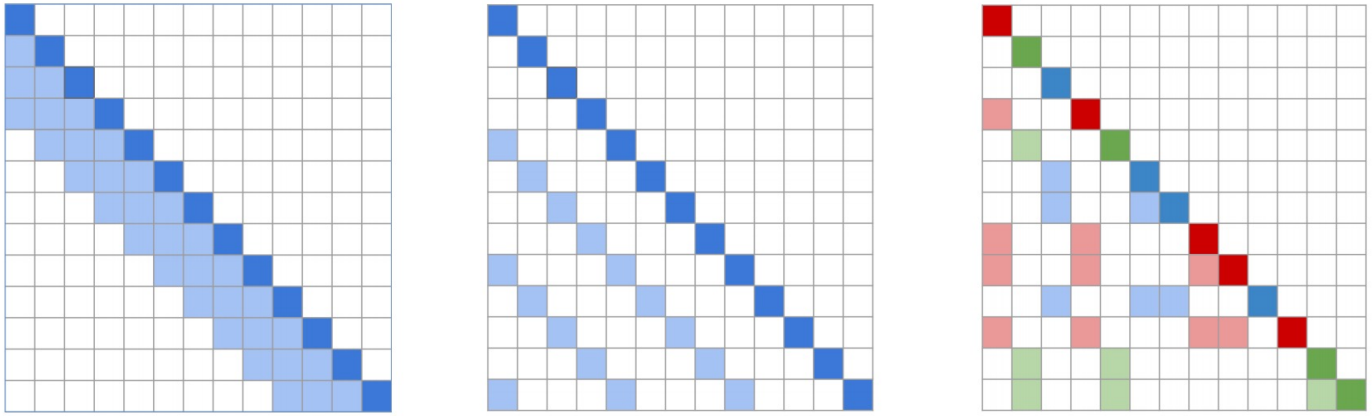
X-formers

- Content-based Sparse Attention

  Condition the sparse connection on the inputs, e.g., use some low-complexity methods to filter out the key-value pairs of high similarity to each query.

  – Routing Transformer (Roy et al., TACL 2020)

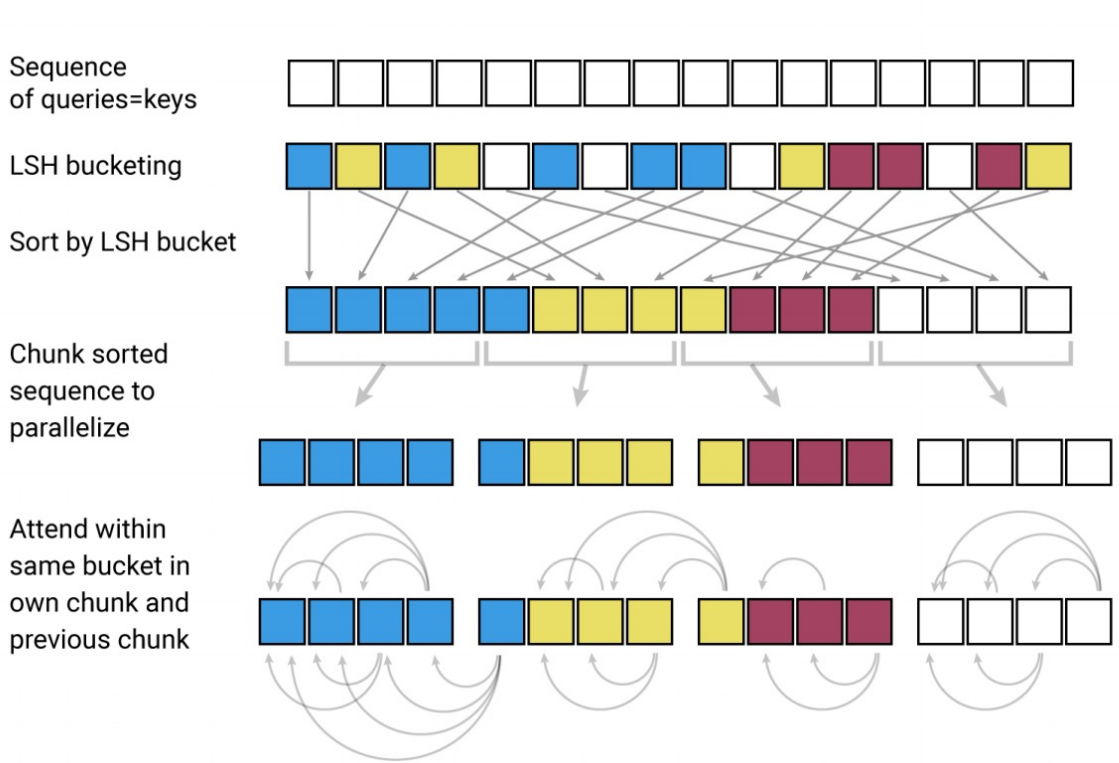
    • uses k-means clustering to cluster both queries and keys

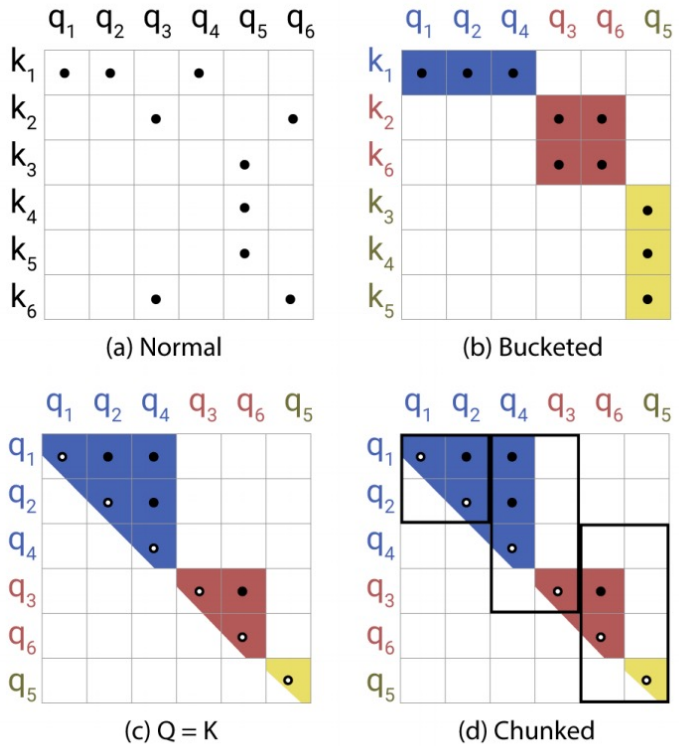


(a) Local attention (b) Strided attention (c) Routing attention

X-formers

- Content-based Sparse Attention
  - Reformer (Kitaev et al., ICLR 2020)

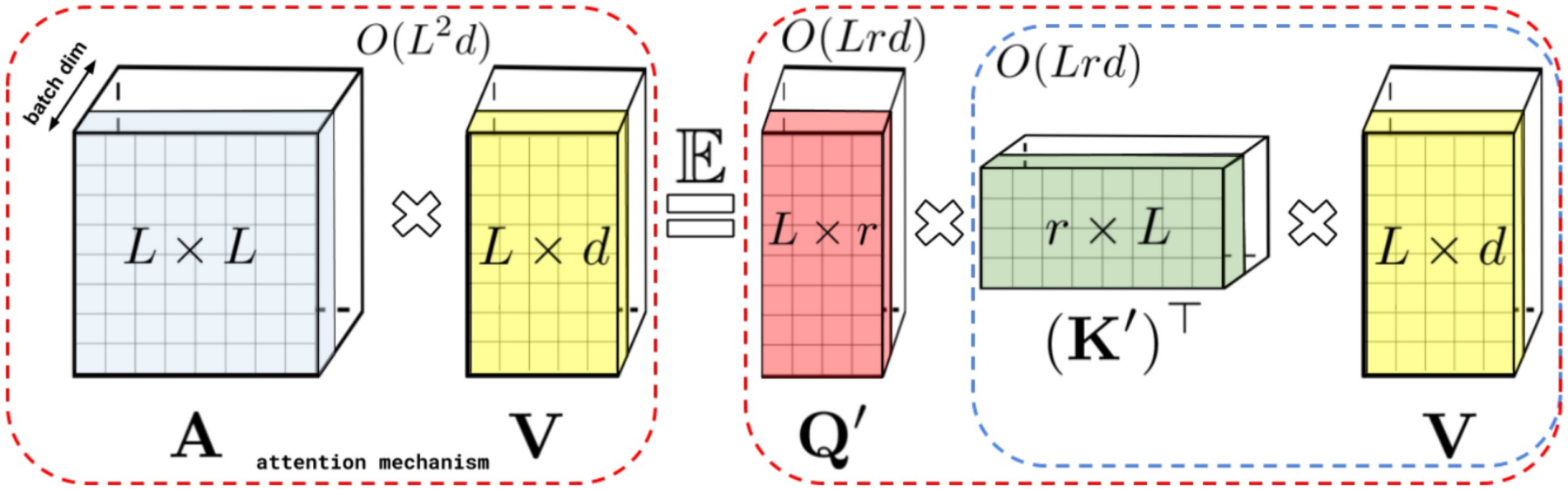




X-formers

- Linearized Attention
  - Performer (Choromanski et al., ICLR 2021)

$$\mathbf{z}_i = \sum_j \frac{\phi(\mathbf{q}_i)\phi(\mathbf{k}_j)^\top}{\sum_{j'} \phi(\mathbf{q}_i)\phi(\mathbf{k}_{j'})^\top}\mathbf{v}_j = \frac{\phi(\mathbf{q}_i)\sum_j \phi(\mathbf{k}_j)\otimes \mathbf{v}_j}{\phi(\mathbf{q}_i)\sum_{j'} \phi(\mathbf{k}_{j'})^\top},$$



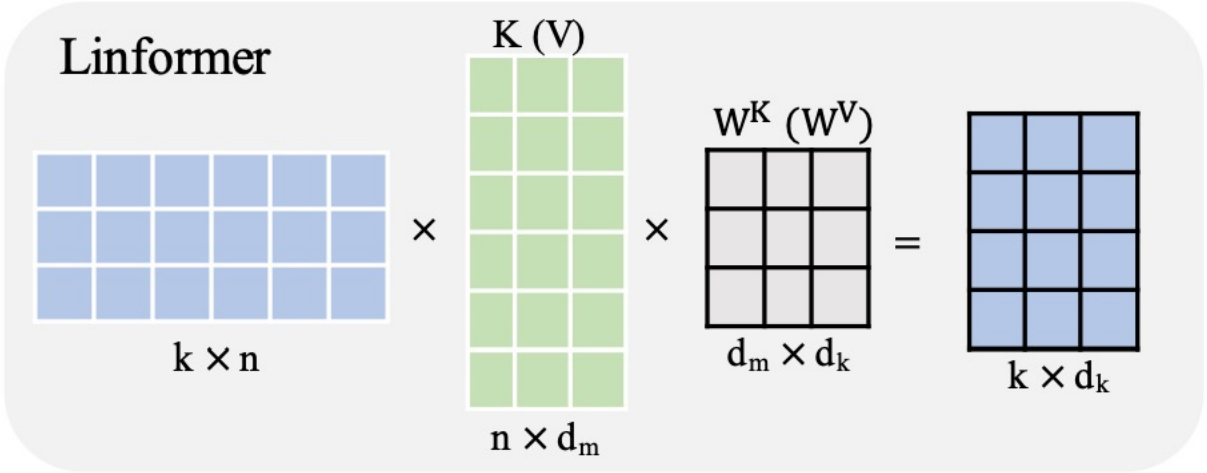
X-formers

- Informer
  - If a query generates attention distribution that is close to uniform, the attention result for this query is a trivial average of value and is redundant for the attention mechanism.

$$M(\mathbf{q}_i, \mathbf{K}) = \ln \sum_{j=1}^{L_K} e^{\frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}}} - \frac{1}{L_K} \sum_{j=1}^{L_K} \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} \longrightarrow \overline{M}(\mathbf{q}_i, \mathbf{K}) = \max_j \{ \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} \} - \frac{1}{L_K} \sum_{j=1}^{L_K} \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}}$$

X-formers

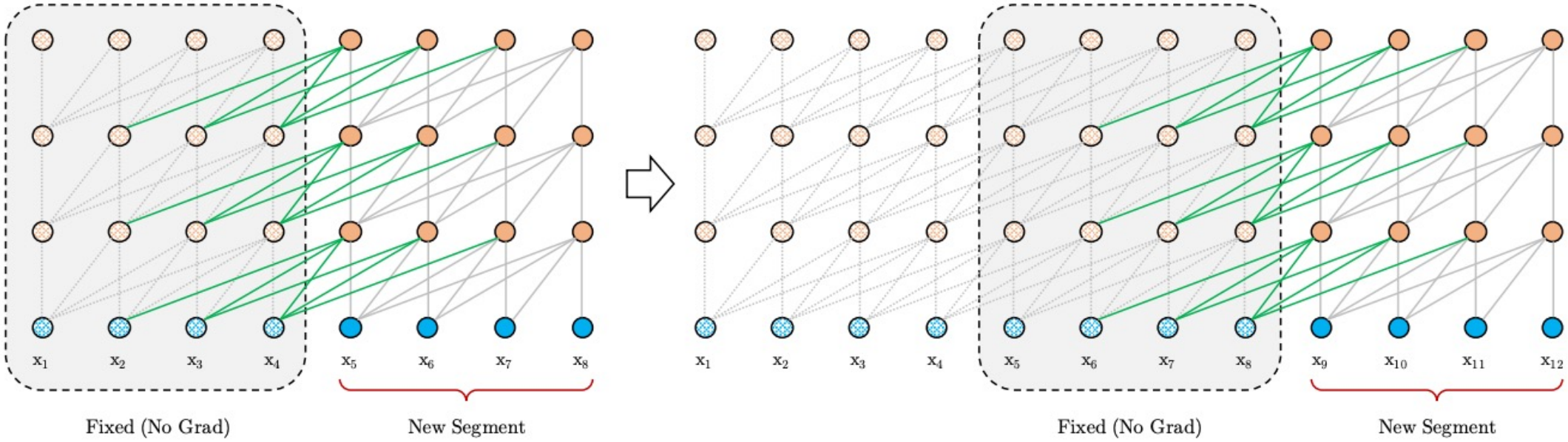
- Linformer



$$\overline{\text{head}_i} = \text{Attention}(QW_i^Q, E_iKW_i^K, F_iVW_i^V)$$
$$= \underbrace{\text{softmax}\left(\frac{QW_i^Q(E_iKW_i^K)^T}{\sqrt{d_k}}\right)}_{\bar{P}:n\times k} \cdot \underbrace{F_iVW_i^V}_{k\times d},$$

X-formers

- Recurrent Transformer
  - Transformer-XL



X-formers

- Recurrent Transformer
  - Transformer-XL

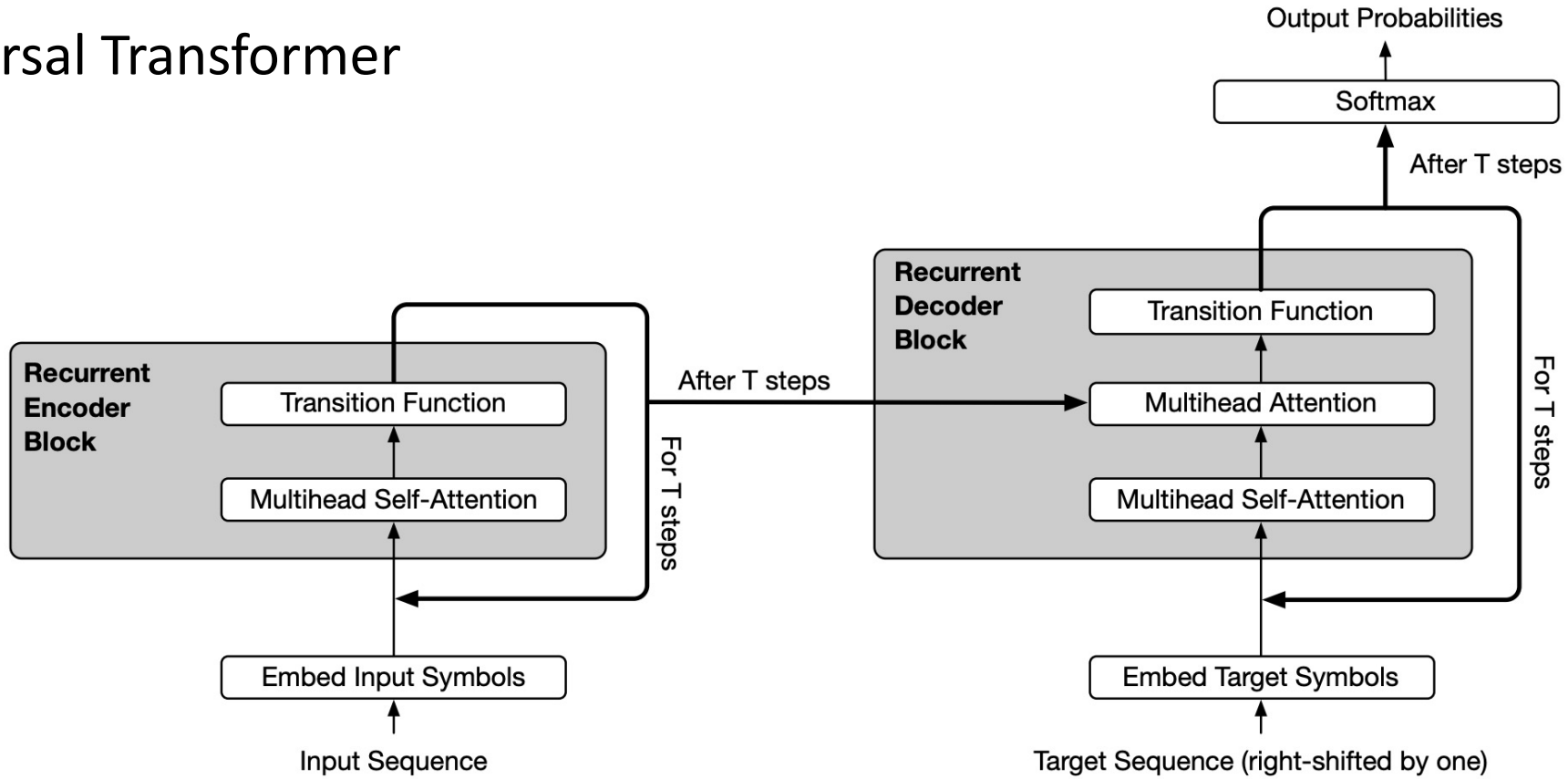
$$\mathbf{A}^{\text{abs}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}.$$

$$\Rightarrow$$

$$\mathbf{A}^{\text{rel}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} + \underbrace{u^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{v^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}.$$

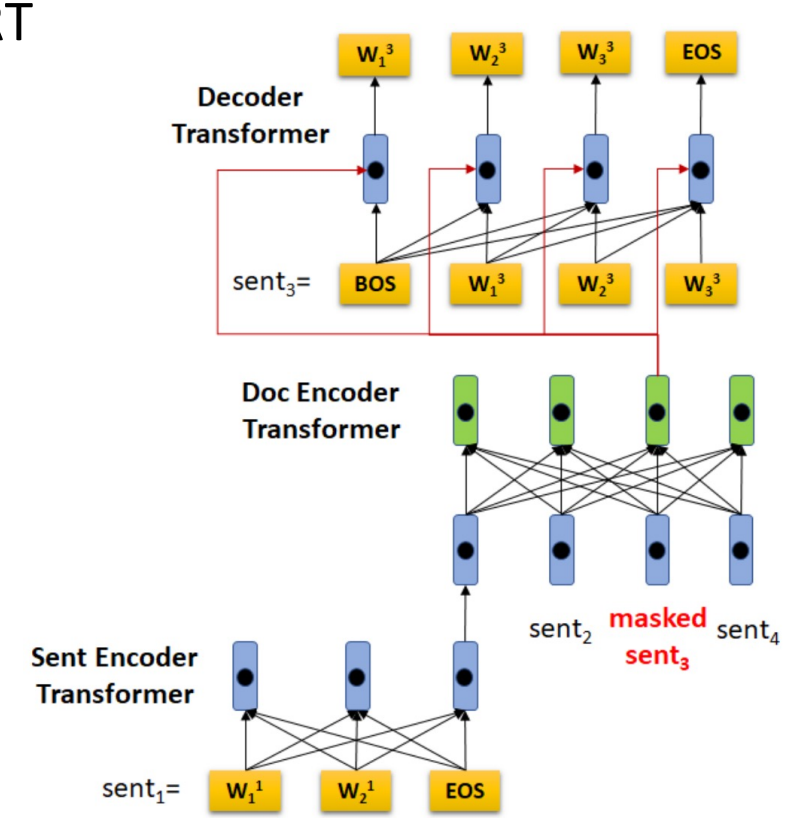
X-formers

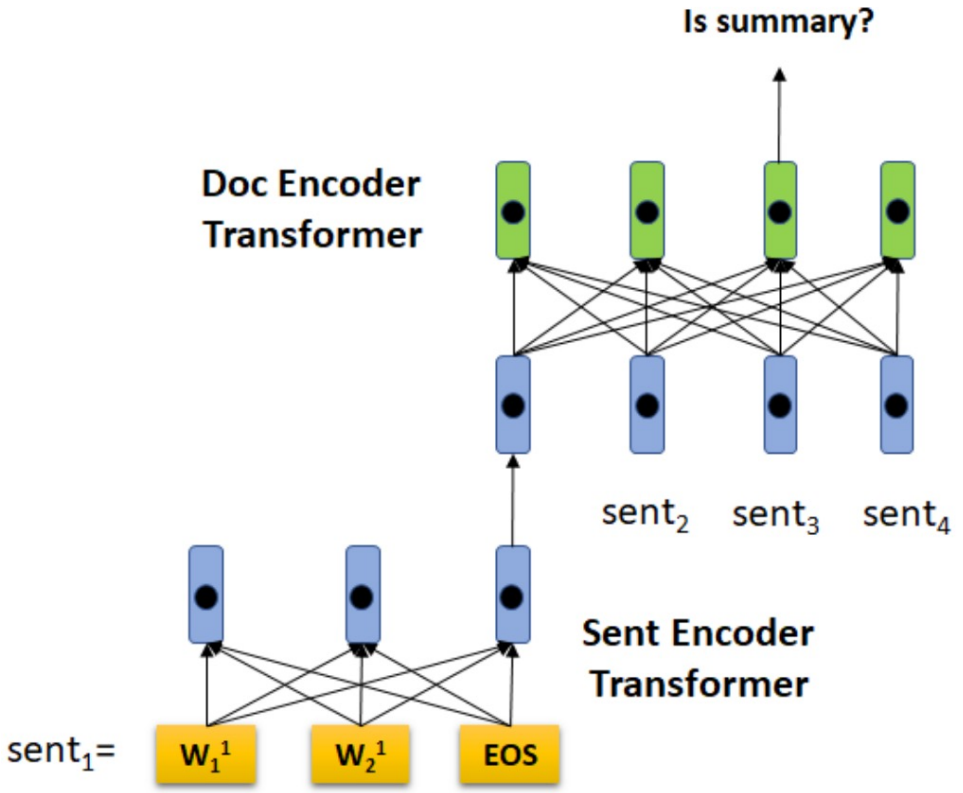
- Recurrent Transformer
  - Universal Transformer



X-formers

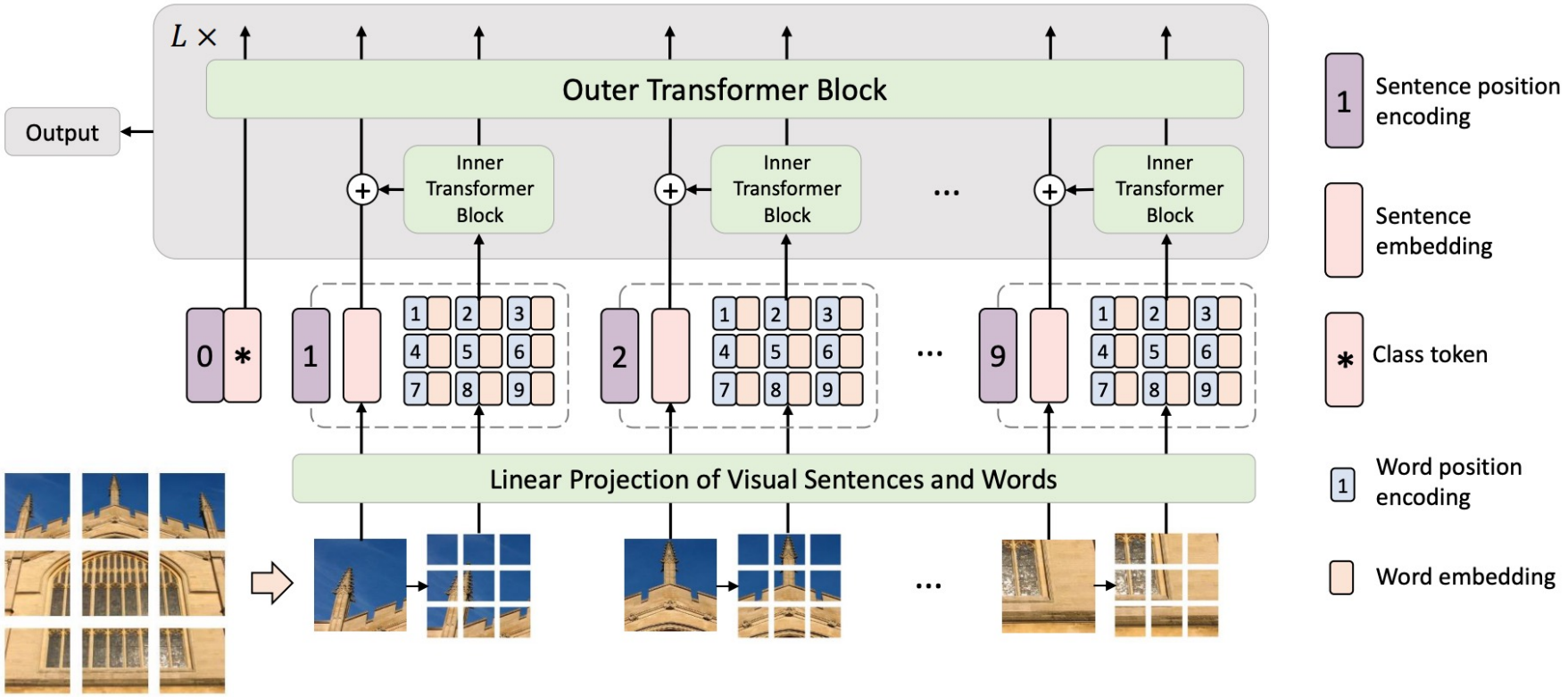
- Hierarchical Transformers
  - HIBERT





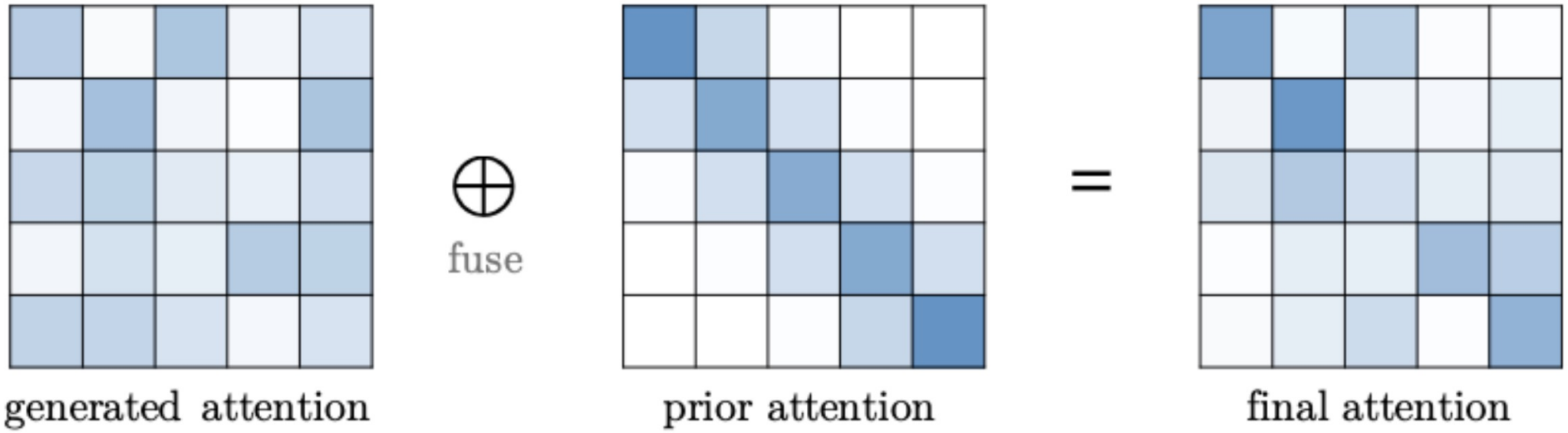
X-formers

- Hierarchical Transformers
  - Transformer in Transformer



Analysis of Transformer module

- Attention with Prior



Analysis of Transformer module

- Attention with Prior
  - Modeling Locality: Local Transformer / Gaussian Transformer
  - Prior from lower modules: Predictive Attention Transformer / RealFormer
  - Task related prior: Conditionally Adaptive Multi-Task Learning
  - Attention with only prior
    - Uniform: Average Attention Network (Zhang et al., ACL 2018)
    - Gaussian: Hard-Coded Gaussian Attention (You et al., ACL 2020)
    - Learnable: Random Synthesizer (Tay et al., ICML 2021)

Analysis of Transformer module

- Improved Multi-head Mechanism
  - Head Behavior Modeling
    - Li et al., Multi-Head Attention with Disagreement Regularization, EMNLP 2018
    - Talking-head Attention (Sukhbaatar et al., 2020)
    - Collaborative multi-head Attention (Cordonnier et al., 2020)
  - Restricted Span
    - Adaptive Attention Span (Sukhbaatar, ACL 2019)
    - Multi-scale Transformer (Guo et al., AAAI 2020)

Analysis of Transformer module

- Improved Multi-head Mechanism
  - Information Aggregation with Dynamic Routing
    - Li et al., Information Aggregation for Multi-Head Attention with Routing-by-Agreement, NAACL 2019
    - Gu and Feng, Improving Multi-head Attention with Capsule Networks, NLPCC 2019
  - Other variants
    - Multi-query Attention (Shazeer, 2019)： shared key-value pairs between heads for computation efficiency；
    - Bhojanapalli et al., Low-Rank Bottleneck in Multi-head Attention Models, ICML 2020：establish that head dimension should be decoupled from the number of heads

Analysis of Transformer module

- Improved Position Embeddings
  - Absolute position:
    - Fixed sinusoidal encoding (vanilla)
    - Learnable embeddings (BERT)
    - Learnable sinusoidal encodings
  - Relative position: Shaw et al., 2018 / Transformer-XL / T5
  - Other representations: TUPE / Roformer
  - Implicit representations: Complex Embedding / R-Transformer / CPE

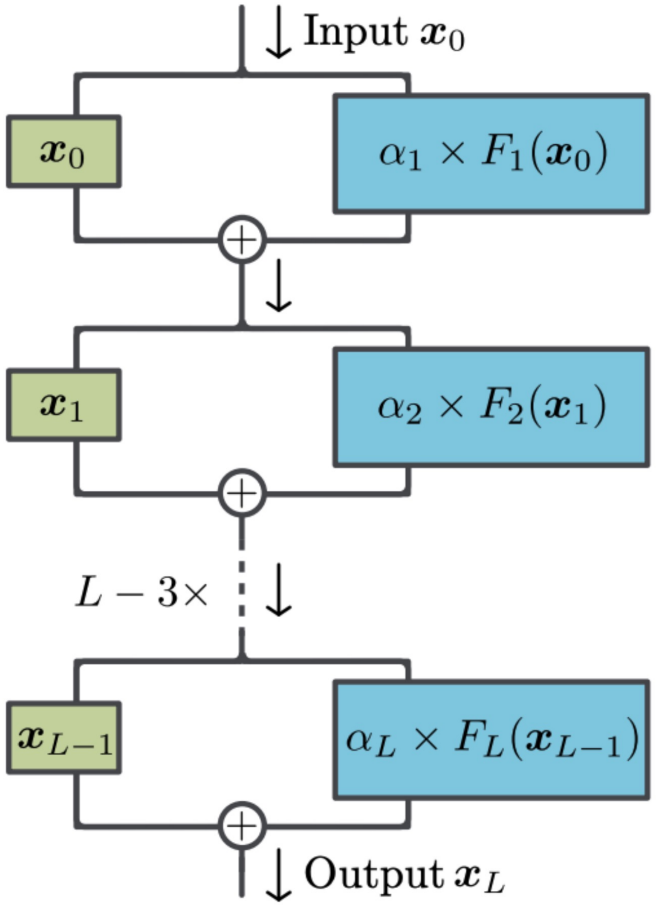
Analysis of Transformer module

- Layer Normaliztion
  - Placement of LN
    - Pre-LN: More stable training;
    - Post-LN: Training could diverge – requires learning rate warm-up, but could lead to better performance when the model converges.
  - Substitutes of LN
    - AdaNorm
    - scaled ℓ_2 normalization
    - PowerNorm (PN)
  - Norm-free Transformer
    - ReZero-Transformer

Analysis of Transformer module

- Layer Normaliztion
  - ReZero
    - Inserts a learnable, zero-initialized parameter α into each residual block

$$\mathbf{H}' = \mathbf{H} + \alpha \cdot F(\mathbf{H}),$$



Analysis of Transformer module

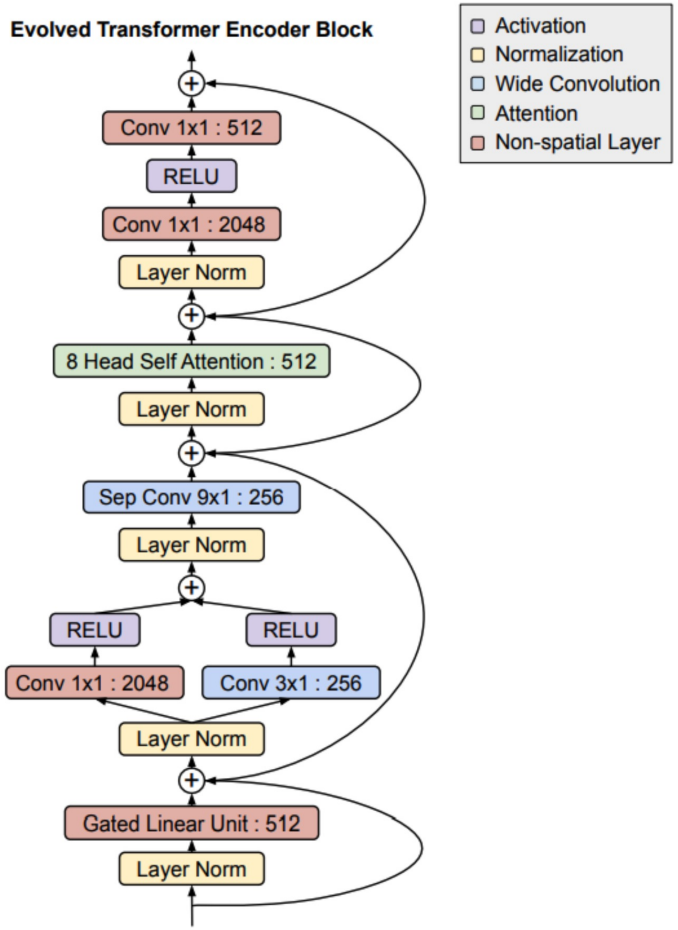
- Position-wised FFN
  - Activation: ReLU, GELU (Gaussian Error Linear Unit), GLU (Gated Linear Unit)
  - Using FFN to enlarge capacity: Product-key memory layer (Lample et al., NeurIPS 2019) / Mixture-of-Experts (Gshard, Switch Transformer, Hash Layer)
  - Can we drop FFN?
    - all-Attention layer (Sukhbaatar et al., 2019): merges FFN into attention module；
    - Yang et al., On the Sub-layer Functionalities of Transformer Decoder: points out that decoder FFN (in encoder-decoder Transformer) can be dropped without affecting performance.

Analysis of Transformer module

- Exploring Alternative Architectures

Is SAN-FFN optimal?

  - Neural ODE: Macaron Transformer
  - NAS: Evolved Transformer / DARTSformer
  - Layer re-ordering: Sandwich Transformer



Thank you