Efficient Transformers: A Survey

Yi Tay, Mostafa Dehghani,
Dara Bahri, Donald Metzler

Long Range Arena: A Benchmark for Efficient Transformers

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, Donald Metzler

Outline

- Introduction
- Efficient Transformers
- Long Range Arena
- Experiments
- Conclusion

Introduction

The Transformer proposed in 2017 has a huge impact on deep learning models.
Its core method, Self Attention, has achieved excellent performance in NLP or CV related tasks.

However, during the calculation process, an attention matrix of L^2 size will be generated.

Introduction

As the input sequence increases, the amount of memory and calculation consumed also increases squarely.

Especially the high memory complexity will seriously hinder the feasibility of Transformer application in long sequence problems.

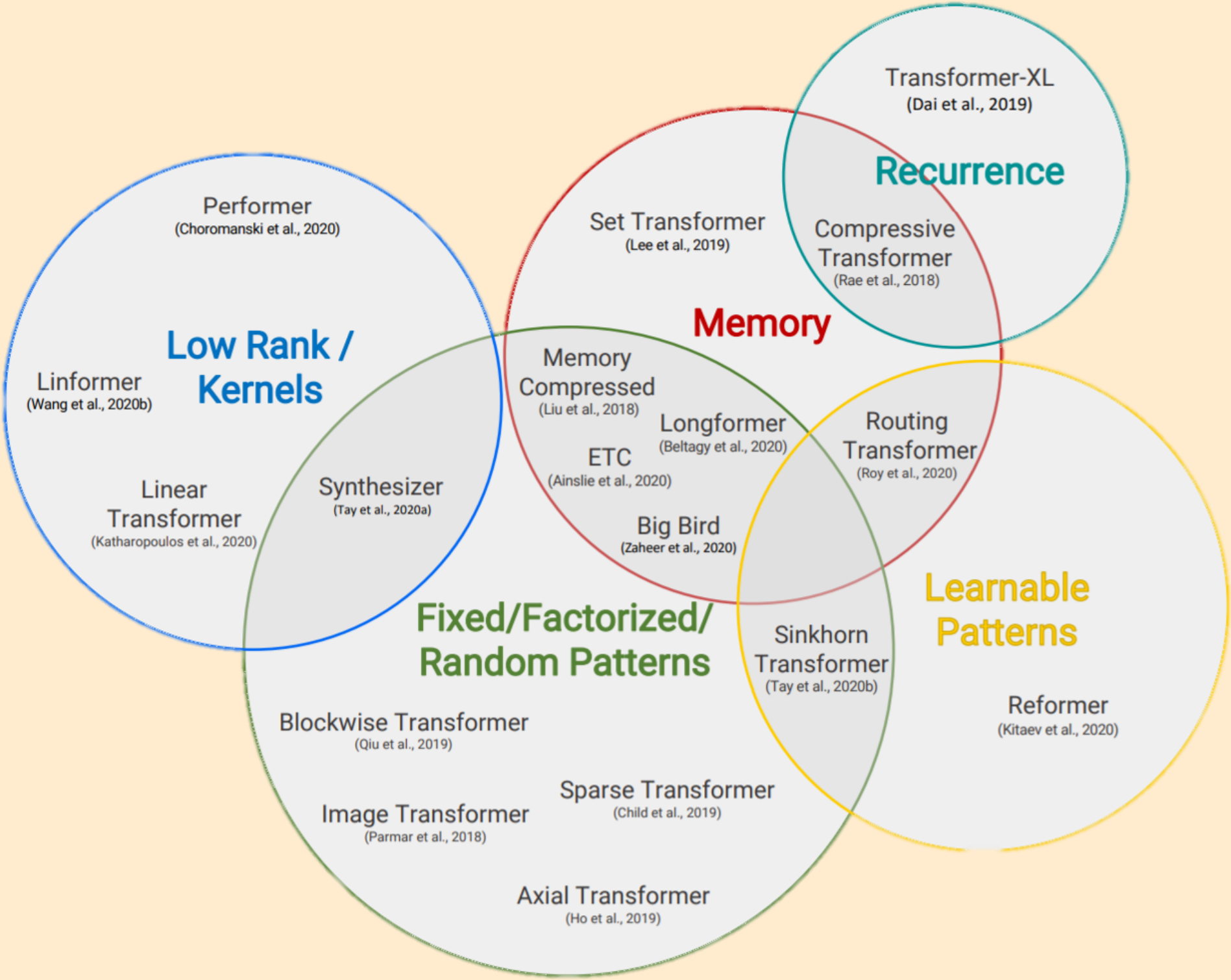
Therefore, many recent papers have proposed various X-formers to minimize the amount of memory required.

Introduction

However, there is currently no systematic benchmark to evaluate the performance of these models on different tasks.

To this end, Tay et al. proposed Long Range Arena, a benchmark composed of multiple tasks, to evaluate the effects of various X-formers on text, images, inference, and other issues.

This report will introduce Transformer improvement methods and Long Range Arena.


Transformer-XL
(Dai et al., 2019)
Recurrence
Performer
(Choromanski et al., 2020)
Set Transformer
(Lee et al., 2019)
Compressive
Transformer
(Rae et al., 2018)
Memory
Low Rank /
Kernels
Linformer
(Wang et al., 2020b)
Memory
Compressed
(Liu et al., 2018)
Longformer
(Beltagy et al., 2020)
Routing
Transformer
(Roy et al., 2020)
ETC
(Ainslie et al., 2020)
Linear
Transformer
(Katharopoulos et al., 2020)
Synthesizer
(Tay et al., 2020a)
Big Bird
(Zaheer et al., 2020)
Learnable
Patterns
Fixed/Factorized/
Random Patterns
Sinkhorn
Transformer
(Tay et al., 2020b)
Reformer
(Kitaev et al., 2020)
Blockwise Transformer
(Qiu et al., 2019)
Sparse Transformer
(Child et al., 2019)
Image Transformer
(Parmar et al., 2018)
Axial Transformer
(Ho et al., 2019)

Fixed Patterns

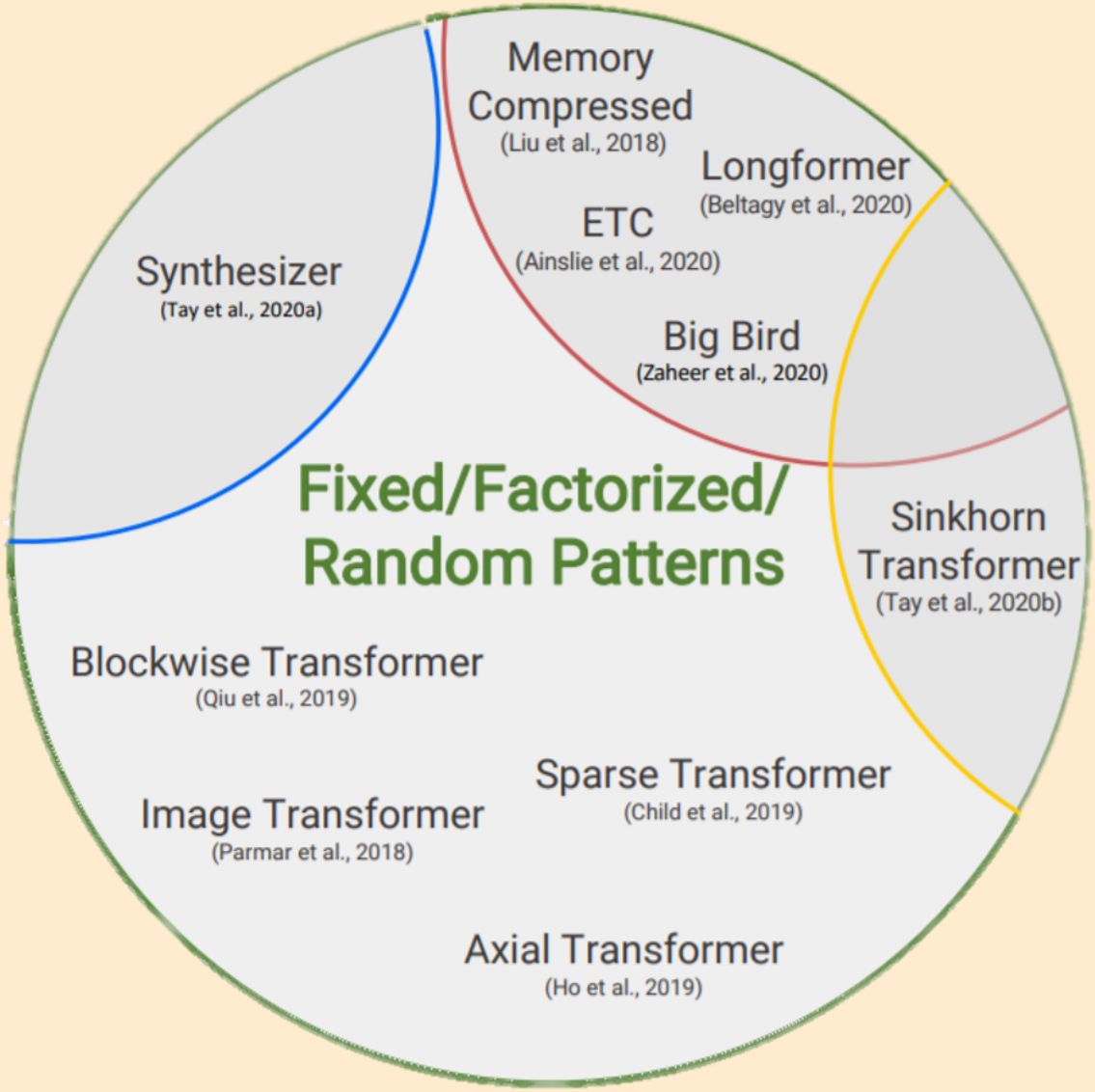
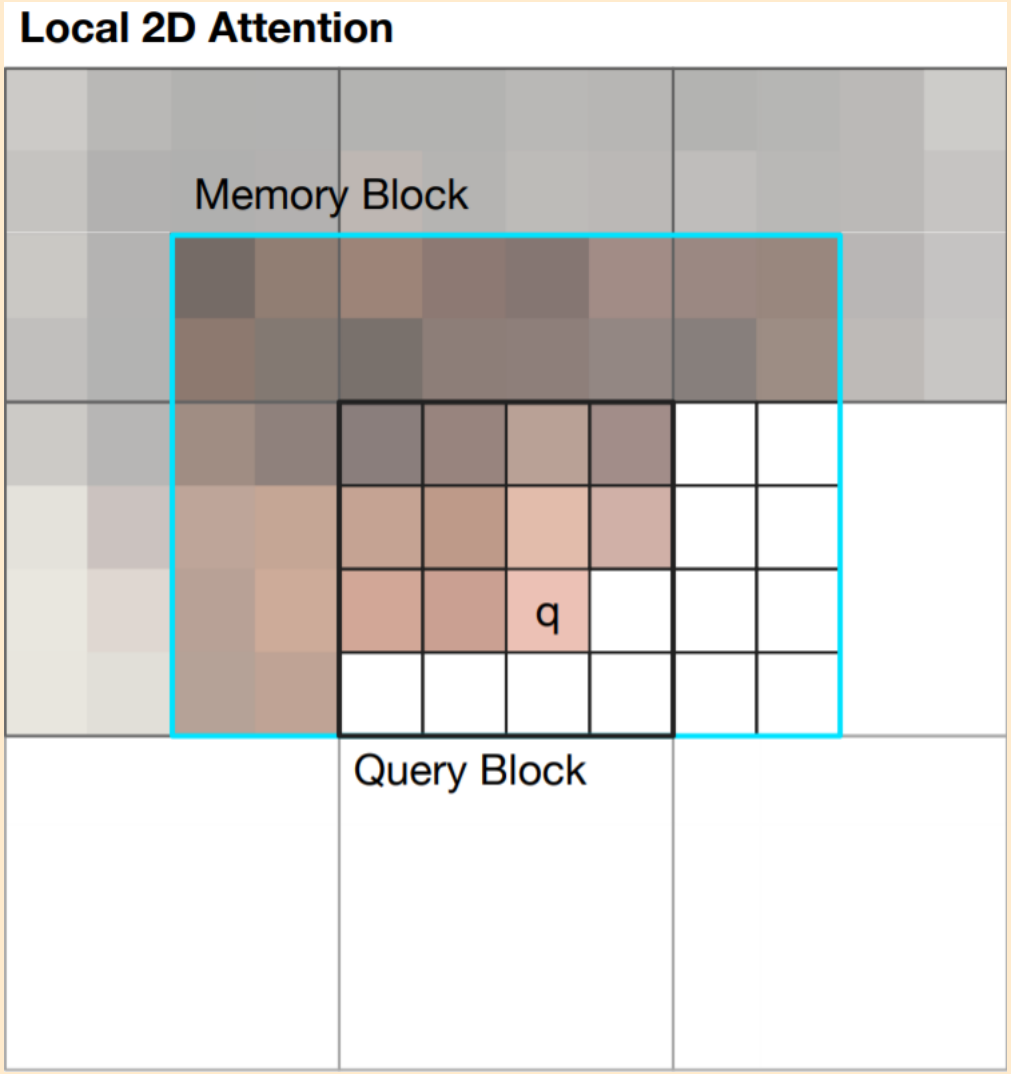
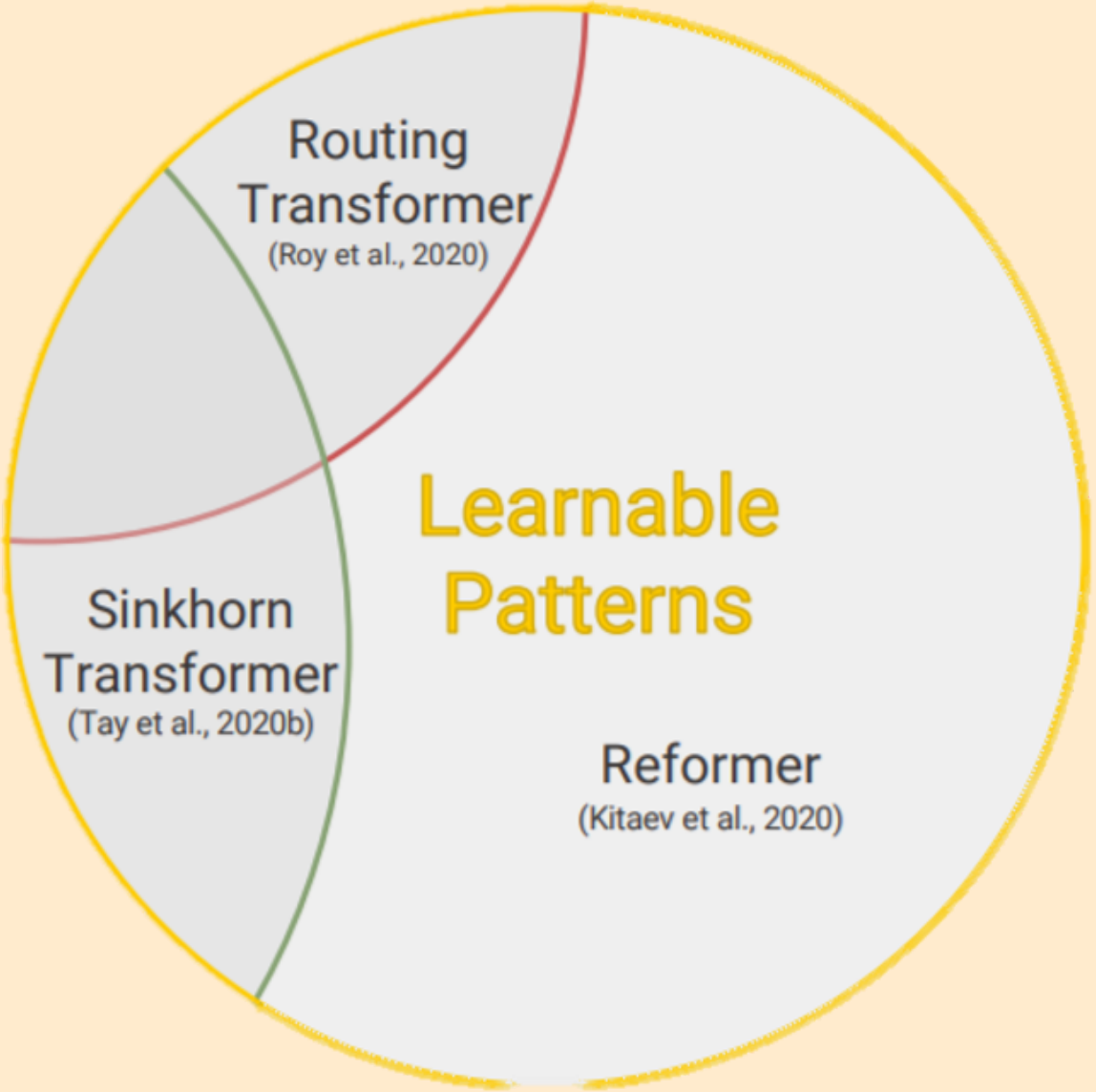


Image Transformer

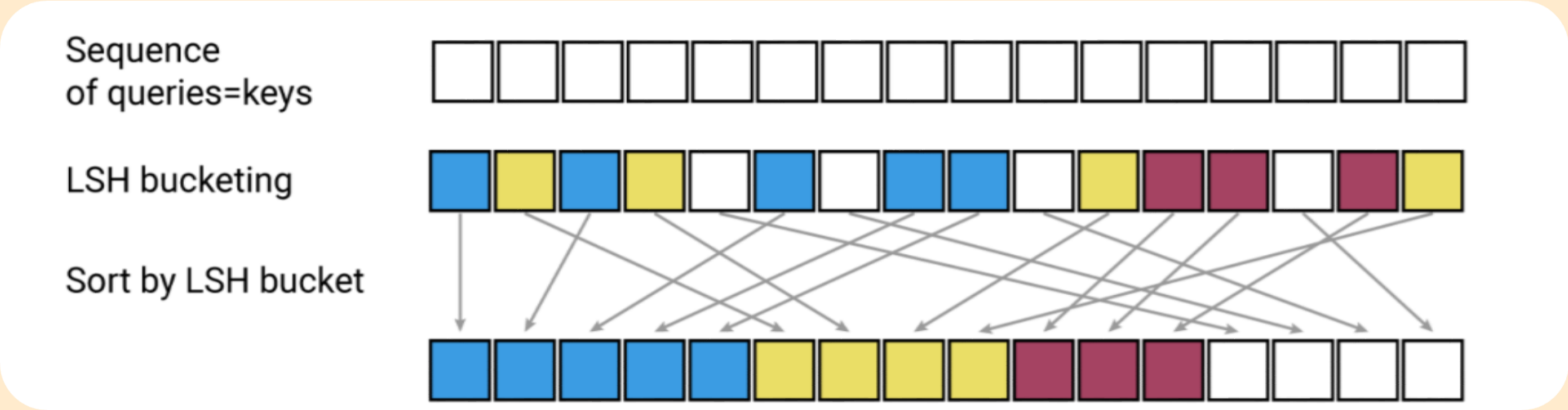


Fixed Patterns uses only partial areas of the attention weight to reduce the amount of calculation.

Learnable Patterns



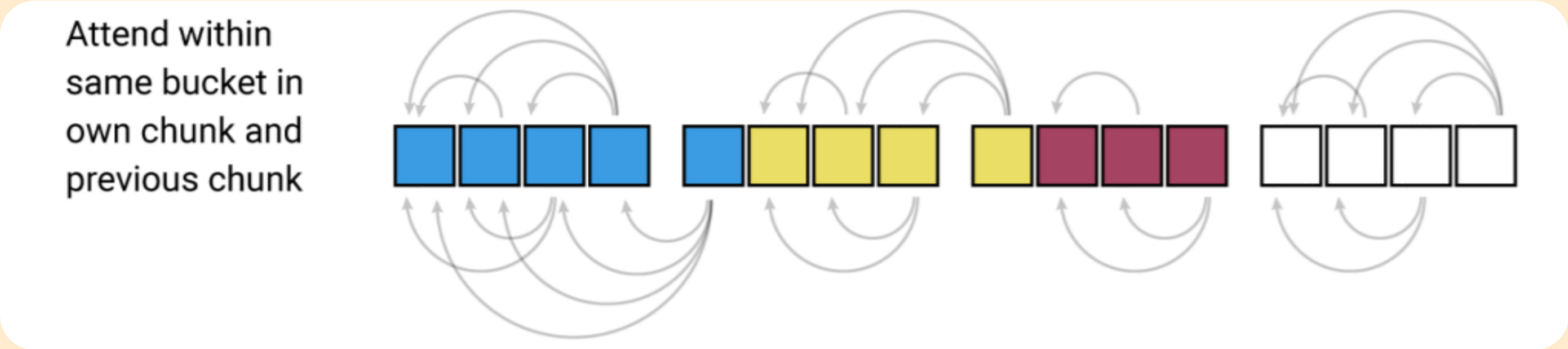
Sort



Group each key and query according to their similarity

- Use Locality Sensitive Hash grouping in Reformer
- Routing Transformer uses k-means grouping

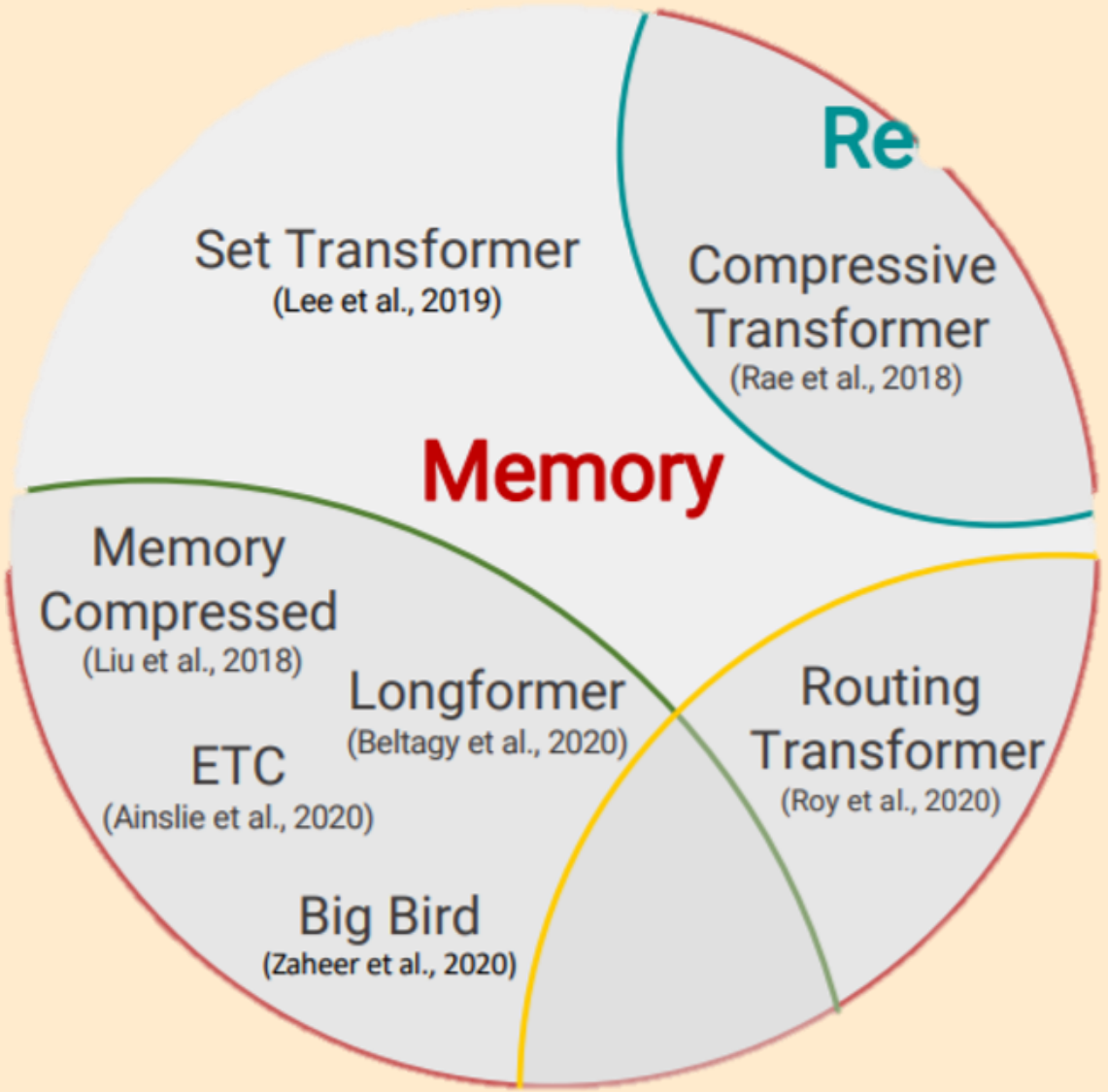
Attention



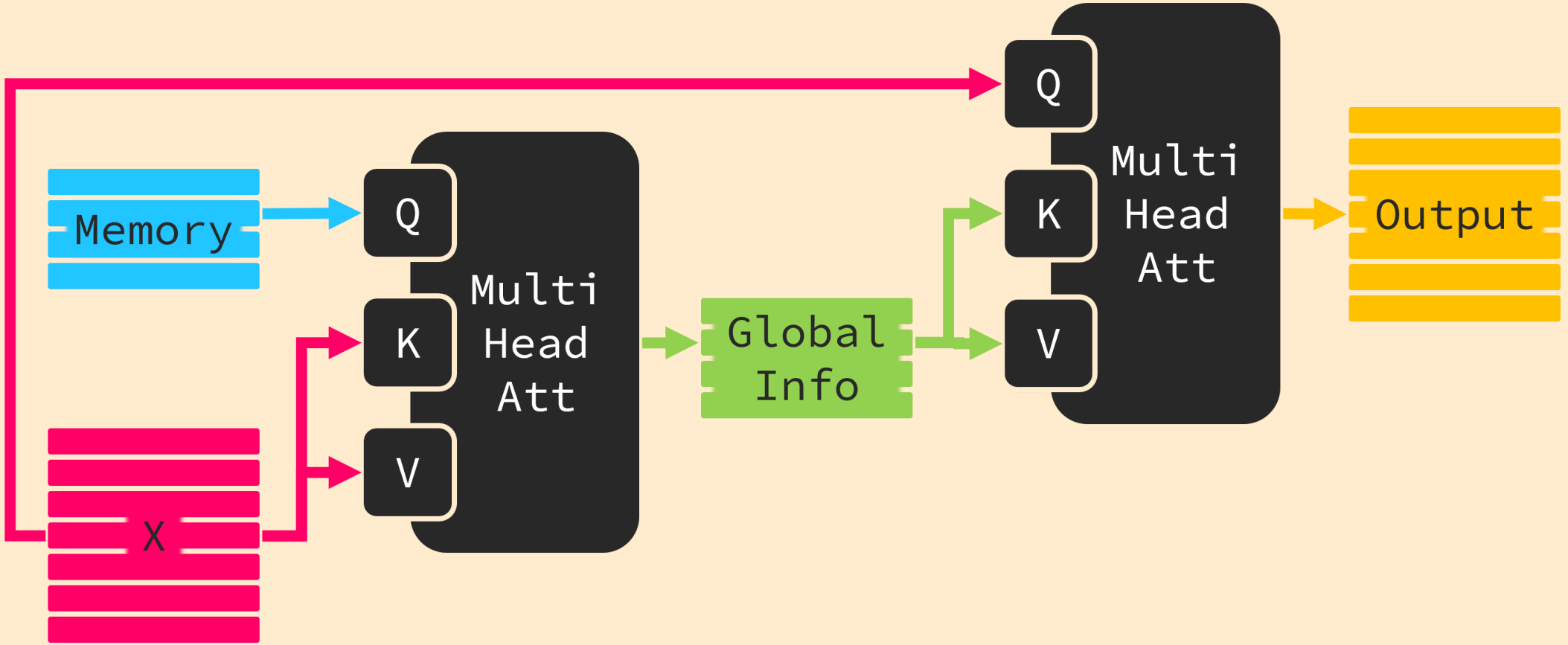
Self Attention in the group

Different groups represent low similarity, and there is no need to match each other

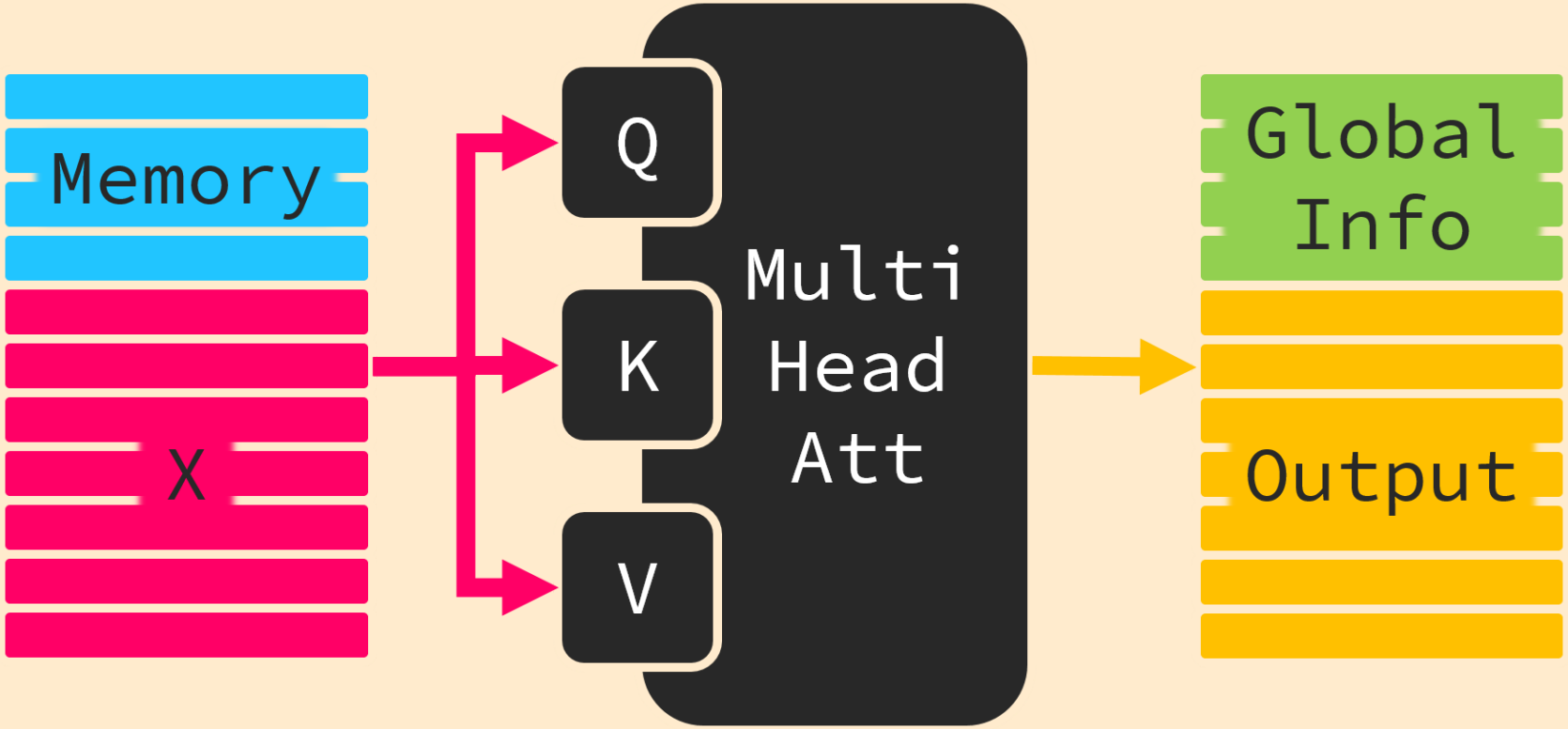
Memory


Set Transformer
(Lee et al., 2019)
Re
Compressive
Transformer
(Rae et al., 2018)
Memory
Memory
Compressed
(Liu et al., 2018)
Longformer
(Beltagy et al., 2020)
ETC
(Ainslie et al., 2020)
Big Bird
(Zaheer et al., 2020)
Routing
Transformer
(Roy et al., 2020)

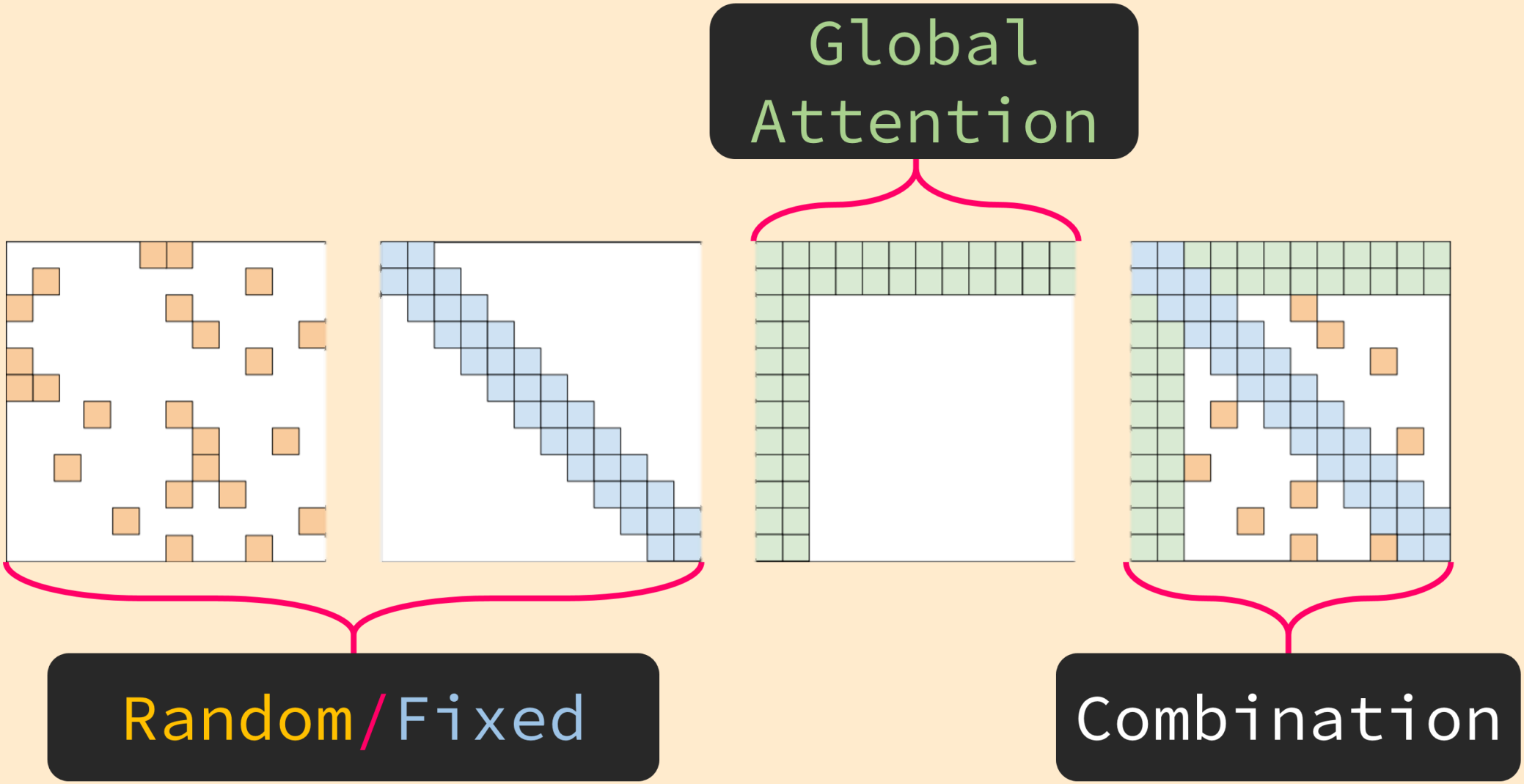
Set Transformer



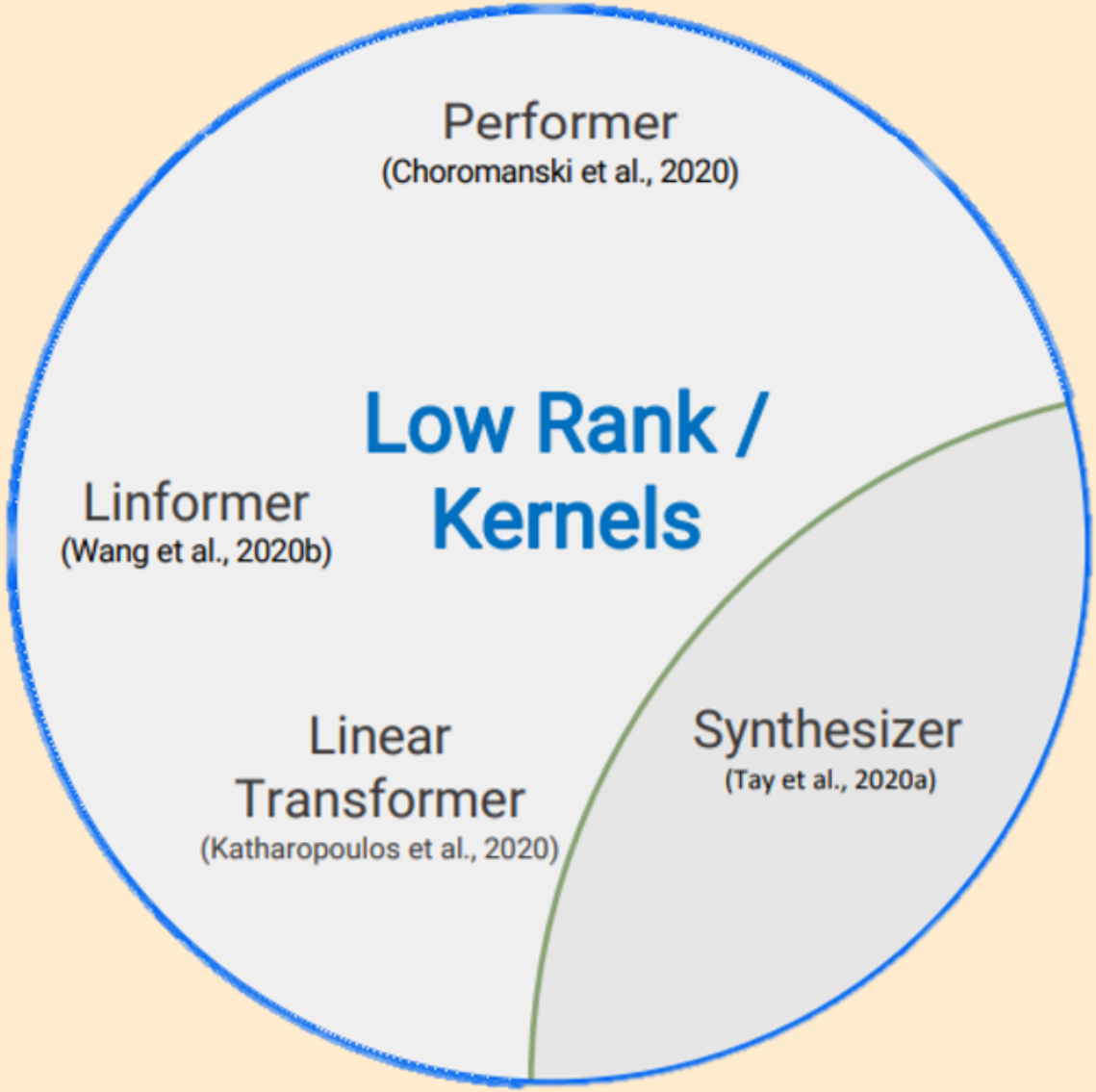
Global Attention



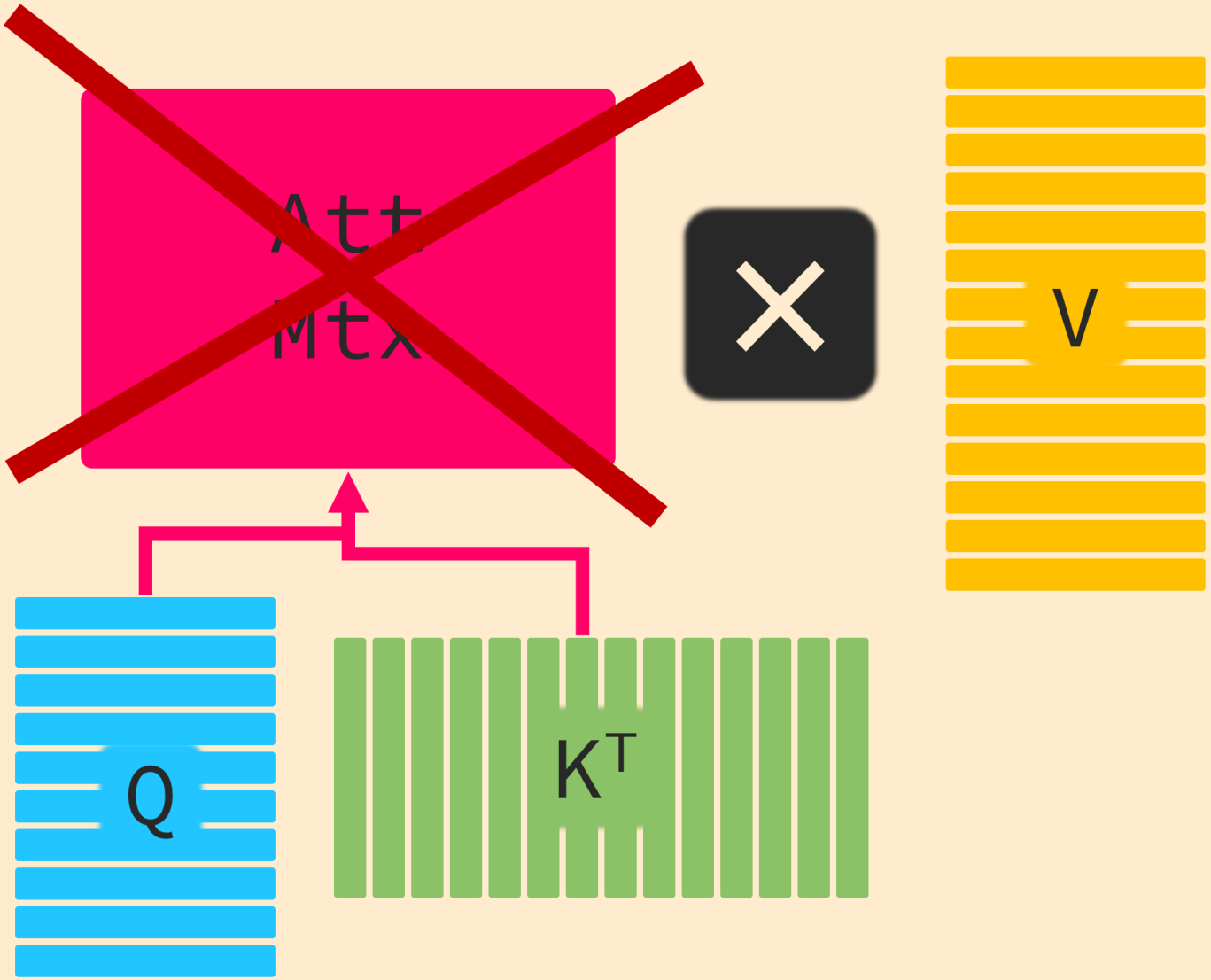
Big Bird



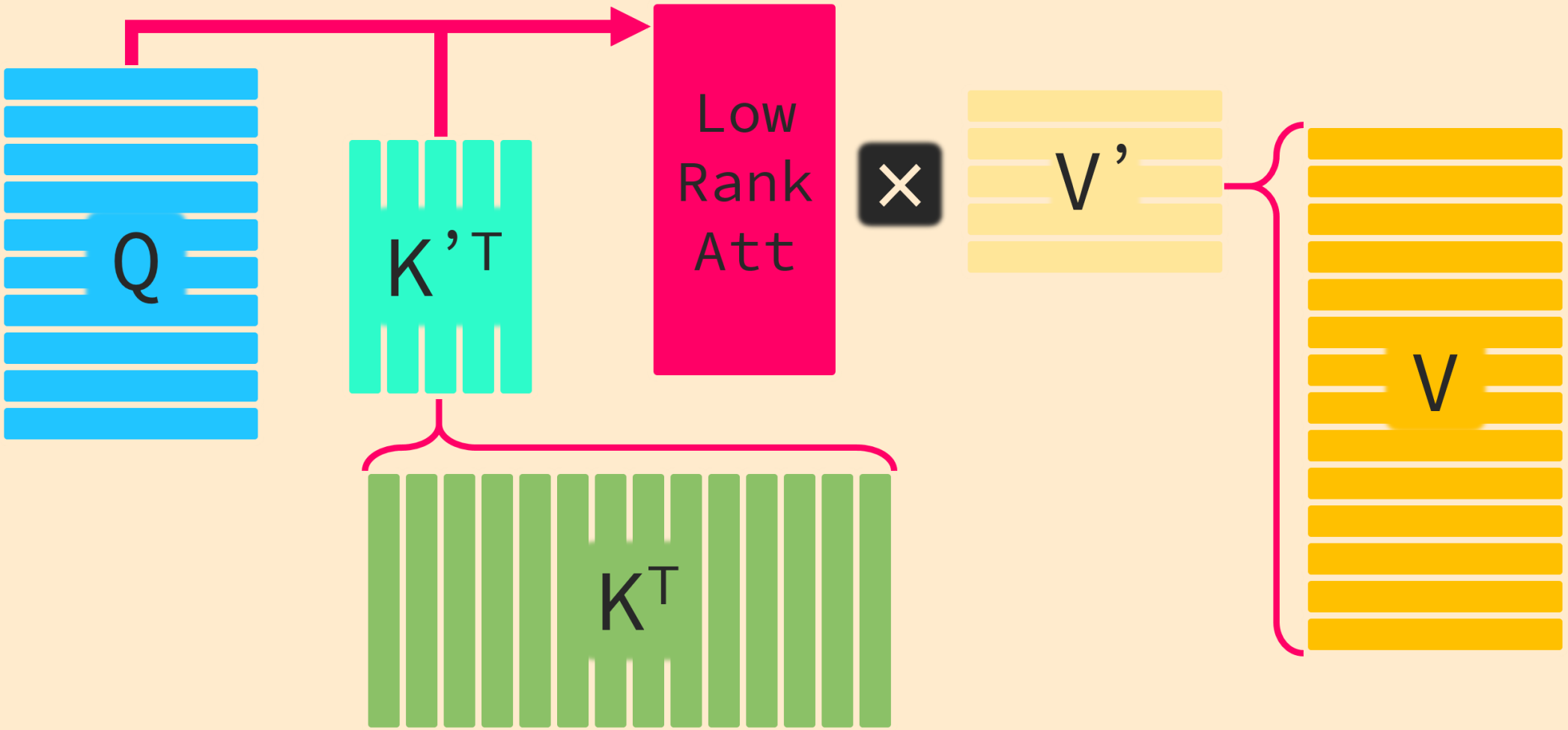
Low Rank & Kernel



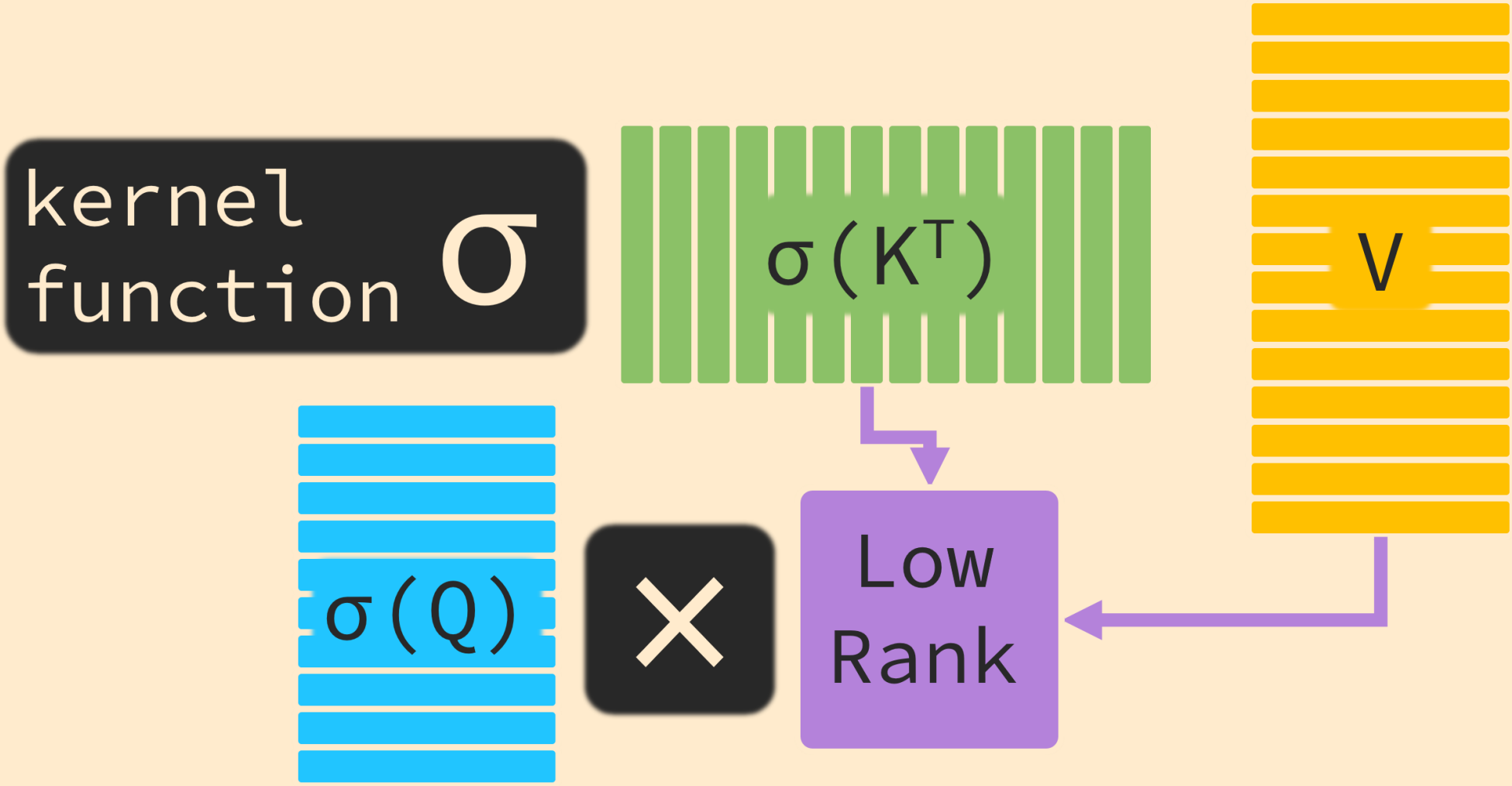
No Attention Matrix



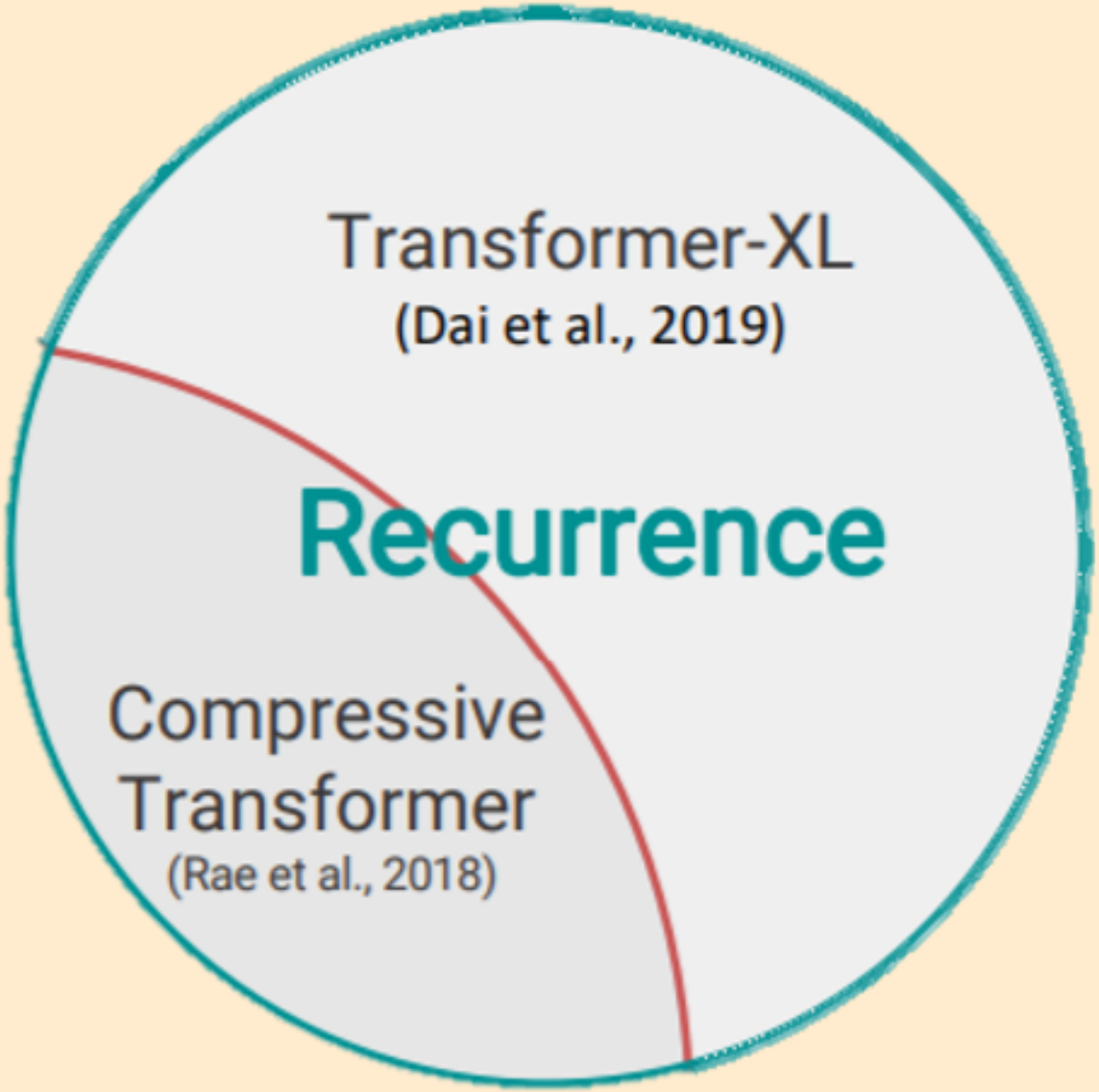
Linformer



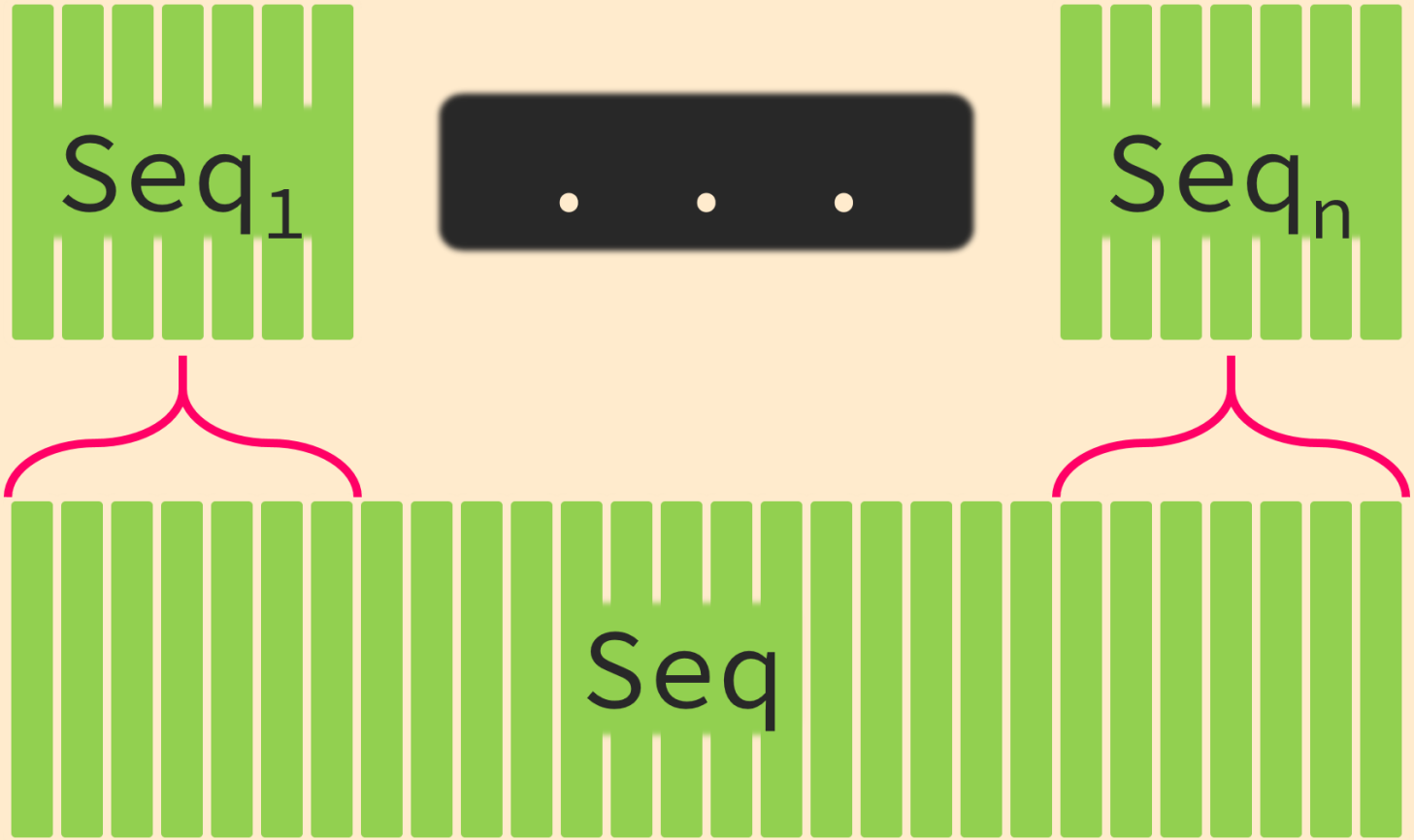
Linear Transformer
Performer



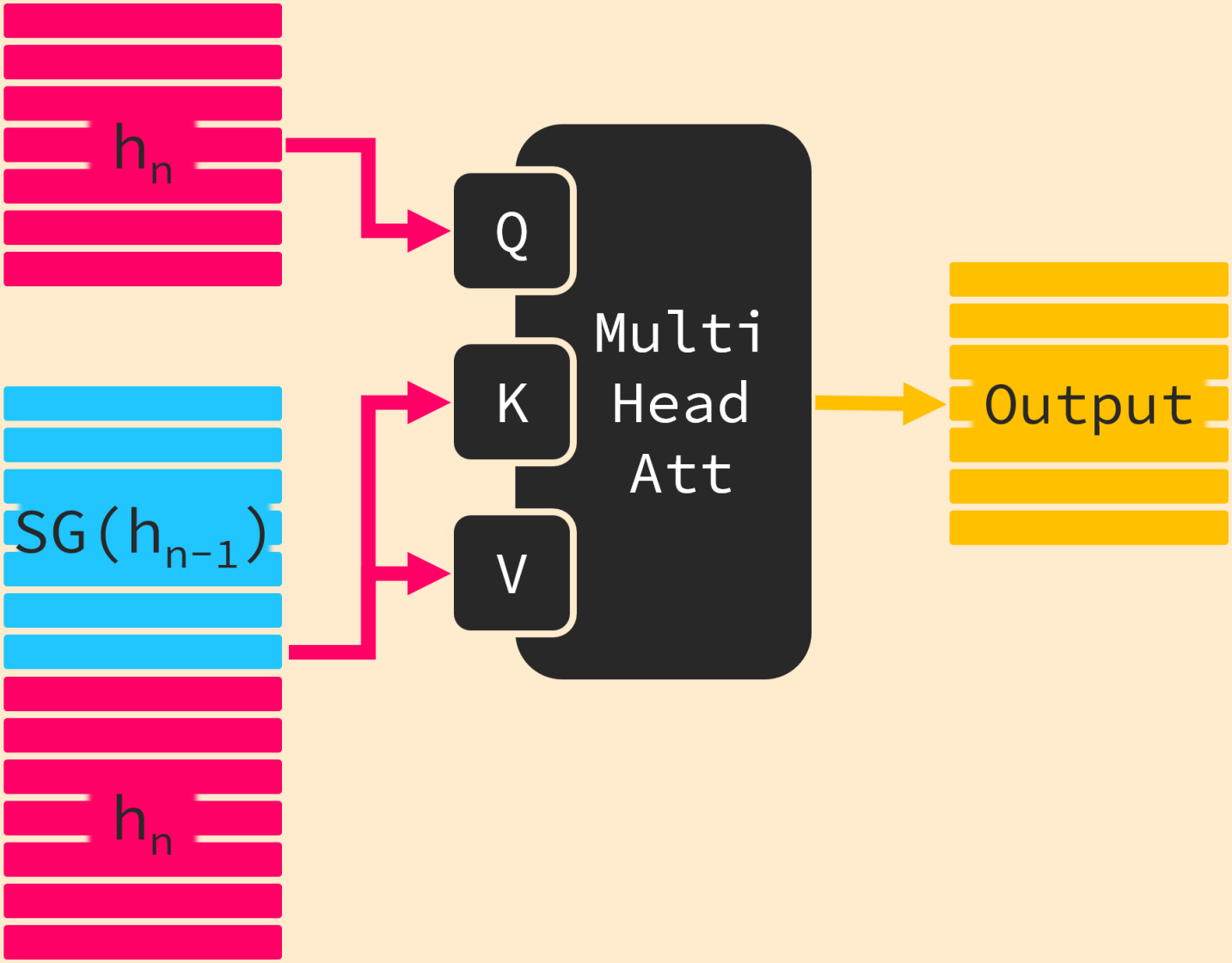
Recurrence


Transformer-XL
(Dai et al., 2019)
Recurrence
Compressive
Transformer
(Rae et al., 2018)

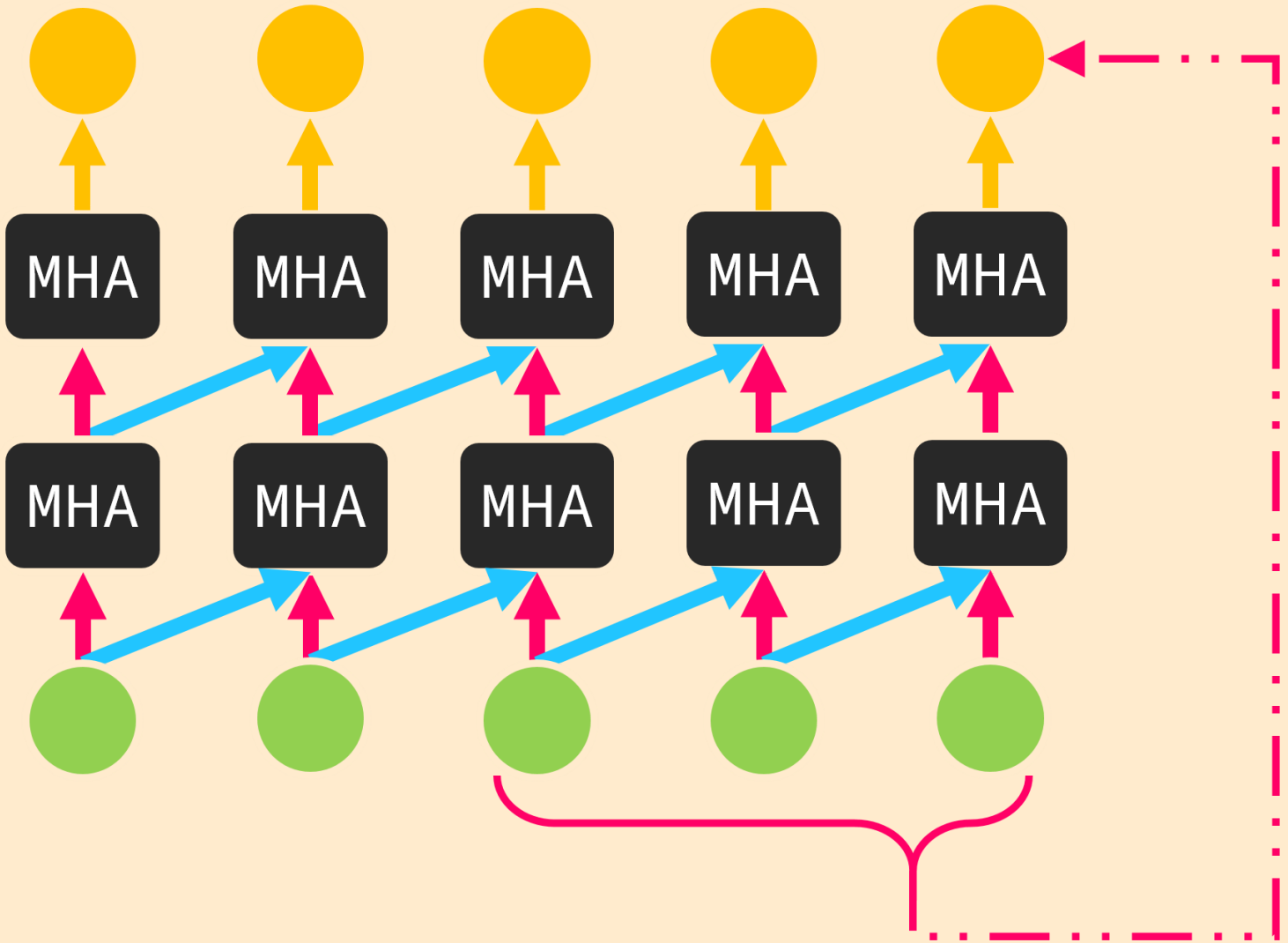
Transformer XL



Transformer XL

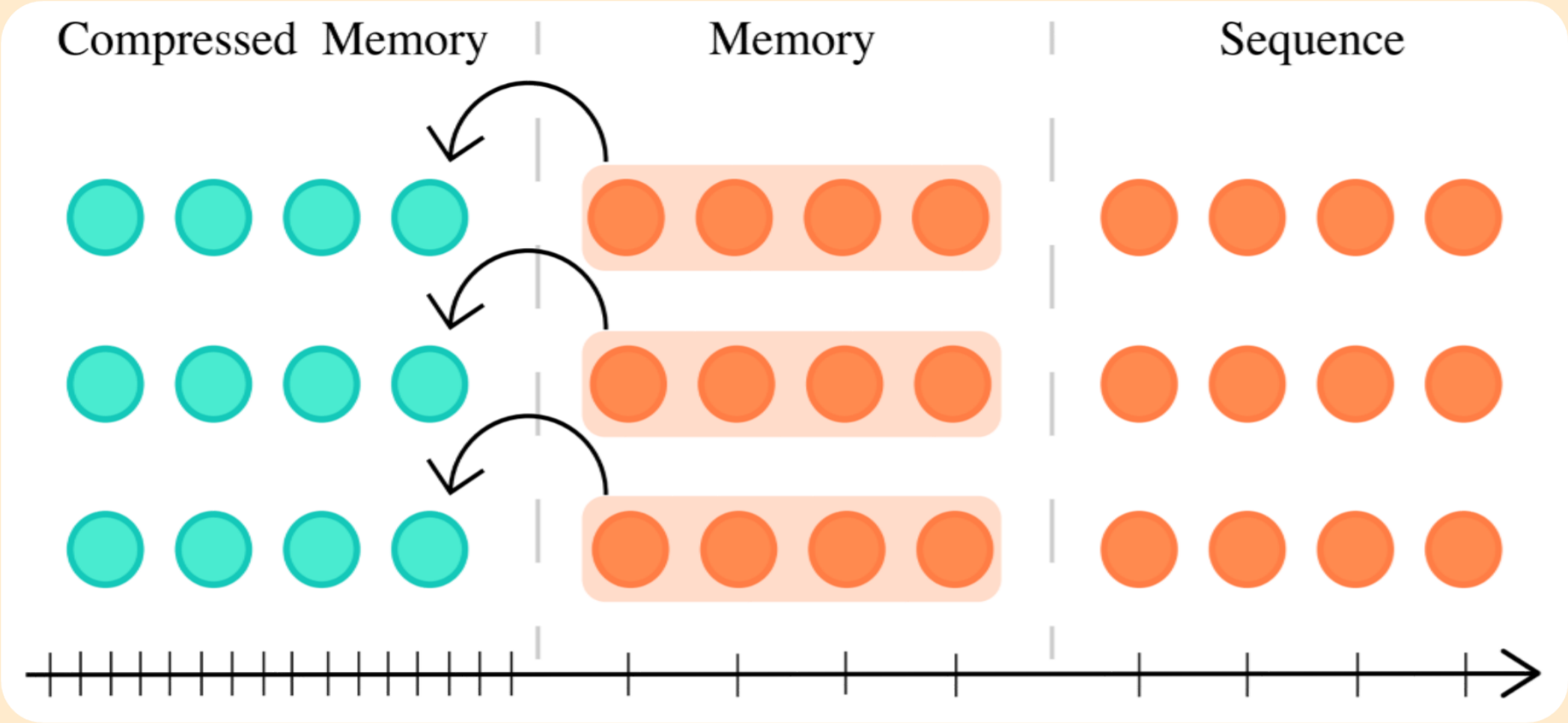


Transformer XL



Recurrence

Compressive Transformer



Shortcomings

- Fixed Patterns
  - The fixed focus area is prone to lack of information.
- Learnable Patterns
  - In the process of grouping, overhead will be generated, and the calculation speed cannot be improved.
- Memory
  - Because the context information is compressed, it cannot be directly applied to autoregressive tasks.
  - And compression will also cause information loss.

Shortcomings

- Low Rank & Kernel
  - It is difficult to implement parallel training in autoregressive tasks.
  - Performer's query and key need a greater depth after conversion to maintain accuracy.
  - Linformer compresses the length L of keys&values to k, but in order to maintain accuracy, the size of k is still related to L.

Task

1. Long ListOpt
2. Byte-Level Text Classification
3. Byte-Level Document Retrieval
4. Image Classification on Sequences of Pixels
5. Pathfinder
6. ~~Pathfinder X~~

Long ListOpt

- Input
  [MAX 4 3
  [MIN 2 3]
  1 0
  [MEDIAN 1 5 8 9 2]]

- Output：5

- Max Seq Len：2K

The preorder expression composed of several operations of max, min, medium, and sum_mod is used as input, and the answer of the operation is used as output.

Byte-Level Text Classification

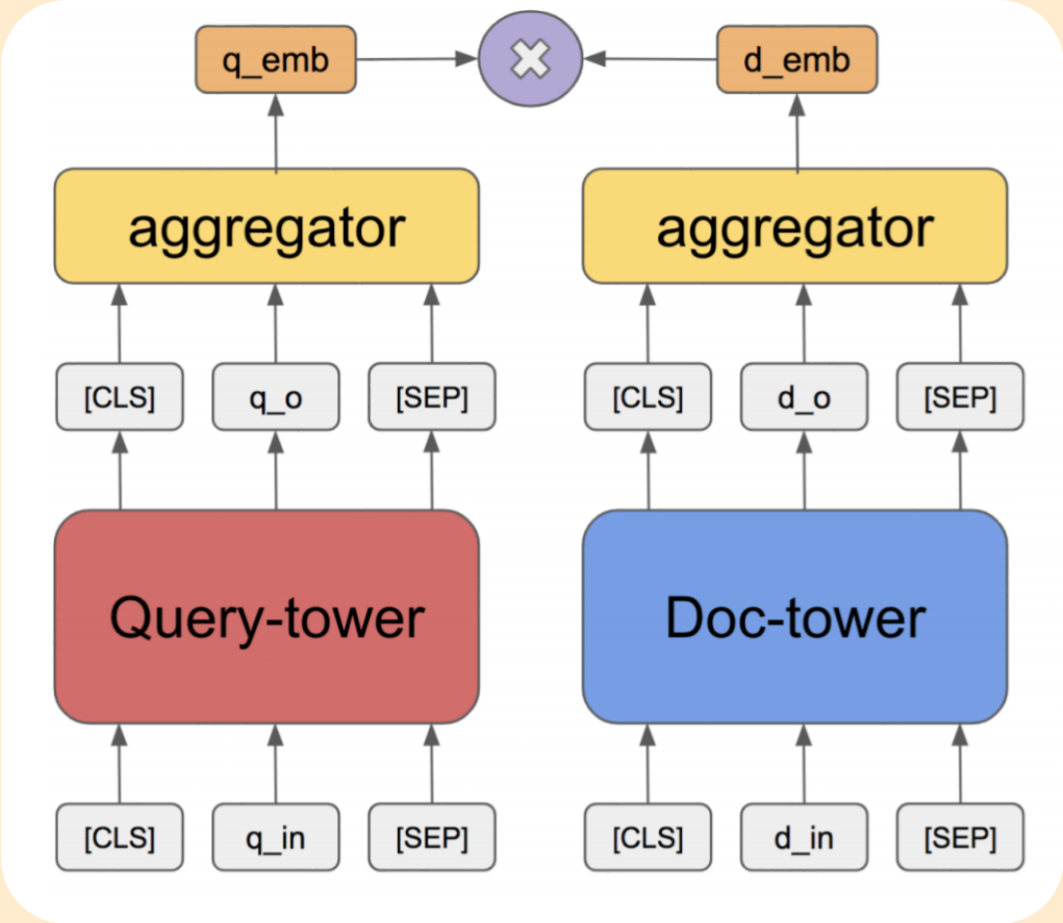
- Dataset：IMDb reviews
- Binary Sentiment Classification
- Max Seq Len：4K

Byte-Level Document Retrieval Task

- Dataset：ACL Anthology Network
- Two Tower Model, Not Cross-Attention
- Similarity Score
- Max Seq Len：4K & 4K

Byte-Level Document Retrieval

Two Tower



Cross Attention

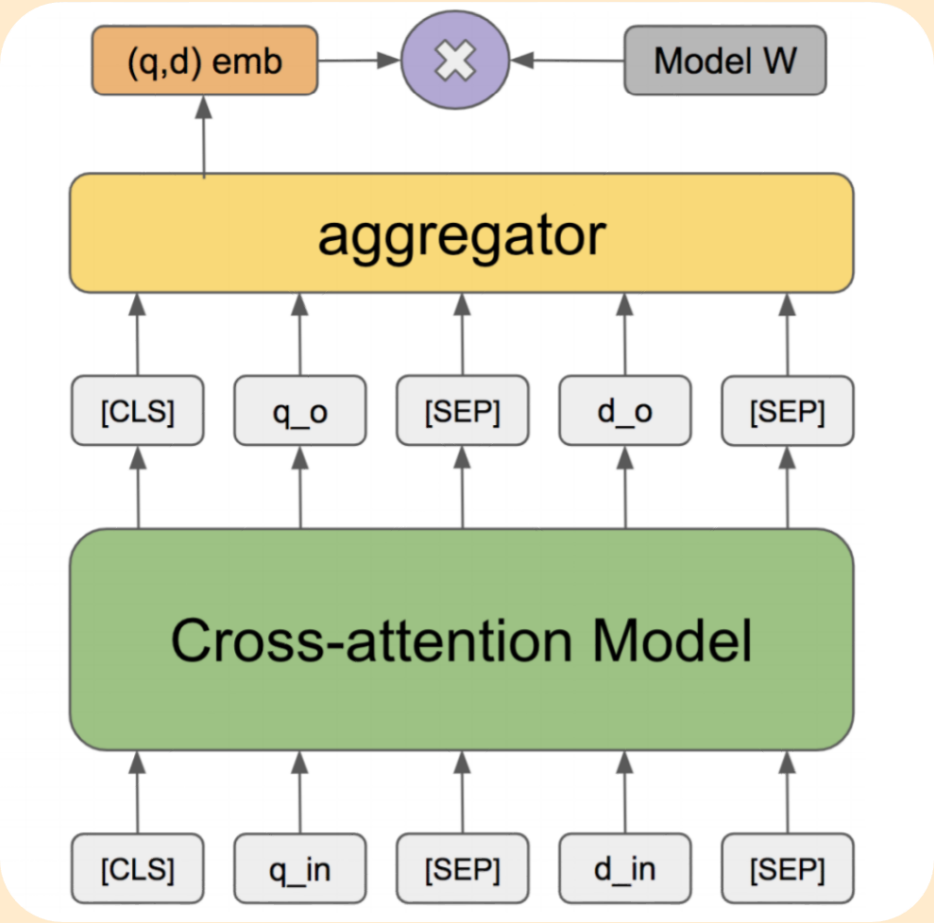
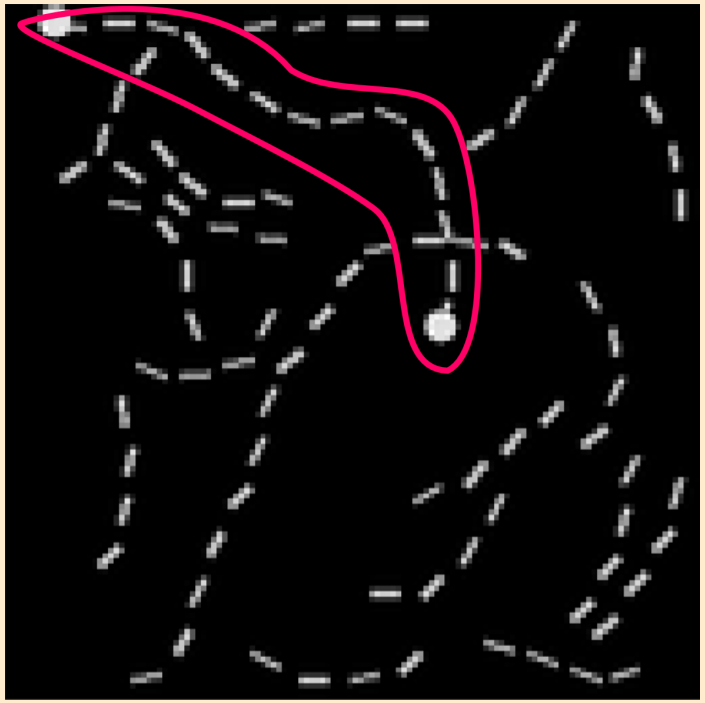


Image Classification

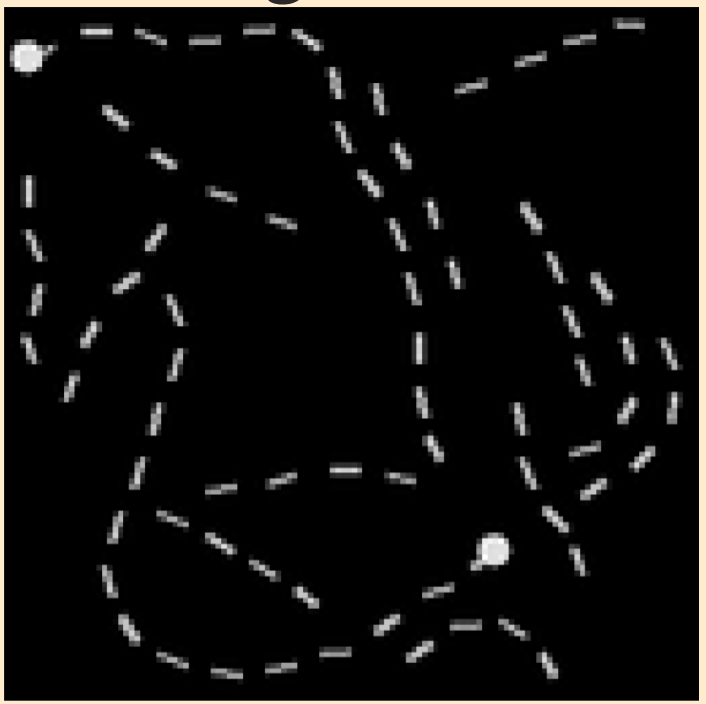
- Dataset：CIFAR-10
- Gray Scale to embedding index
- Vocabulary size of 256(=Gray Scale)
- Max Seq Len：1024 (32x32)

Pathfinder

Positive

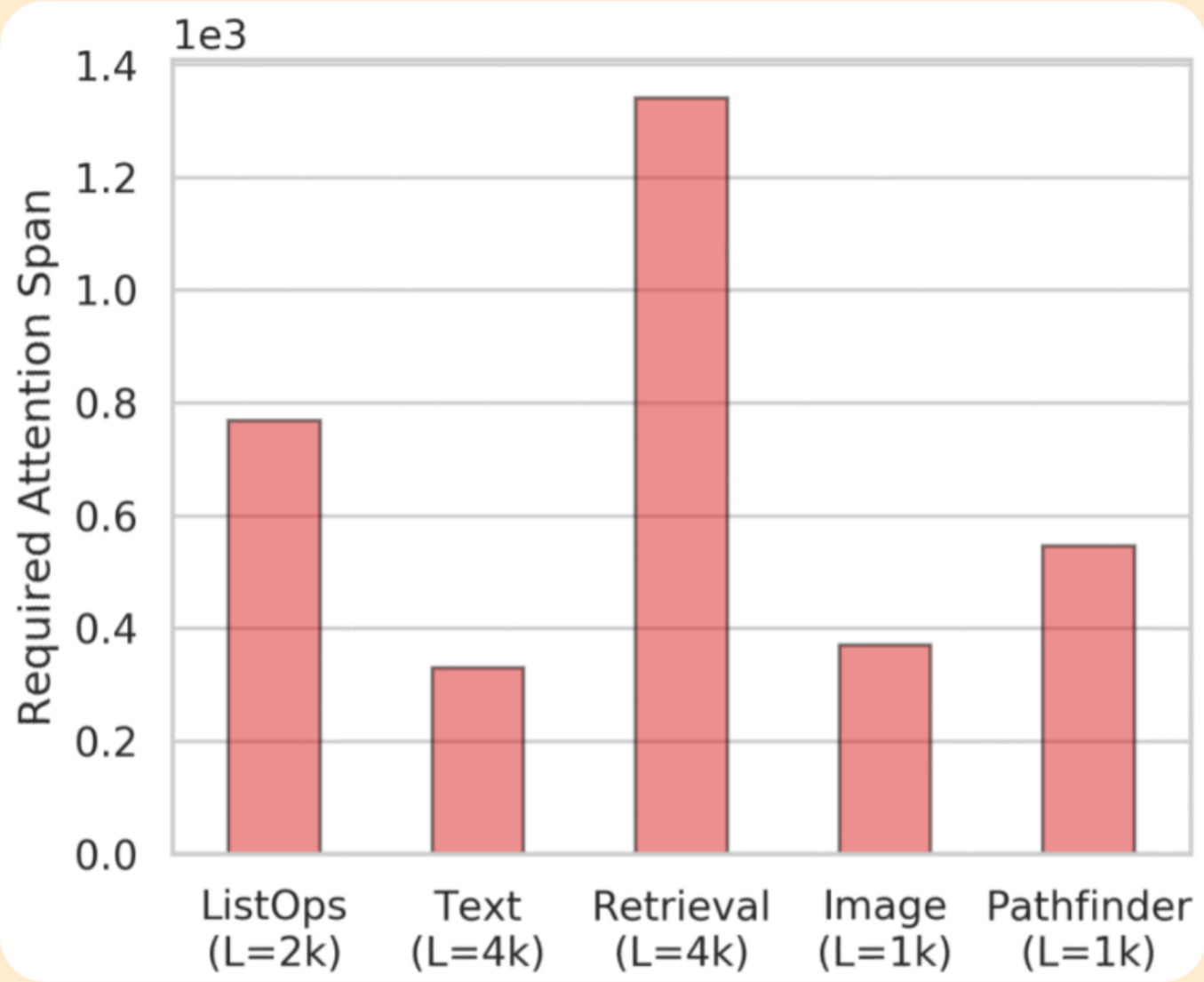


Negative



Max Seq Len：1024 (32x32)

Required Attention Span



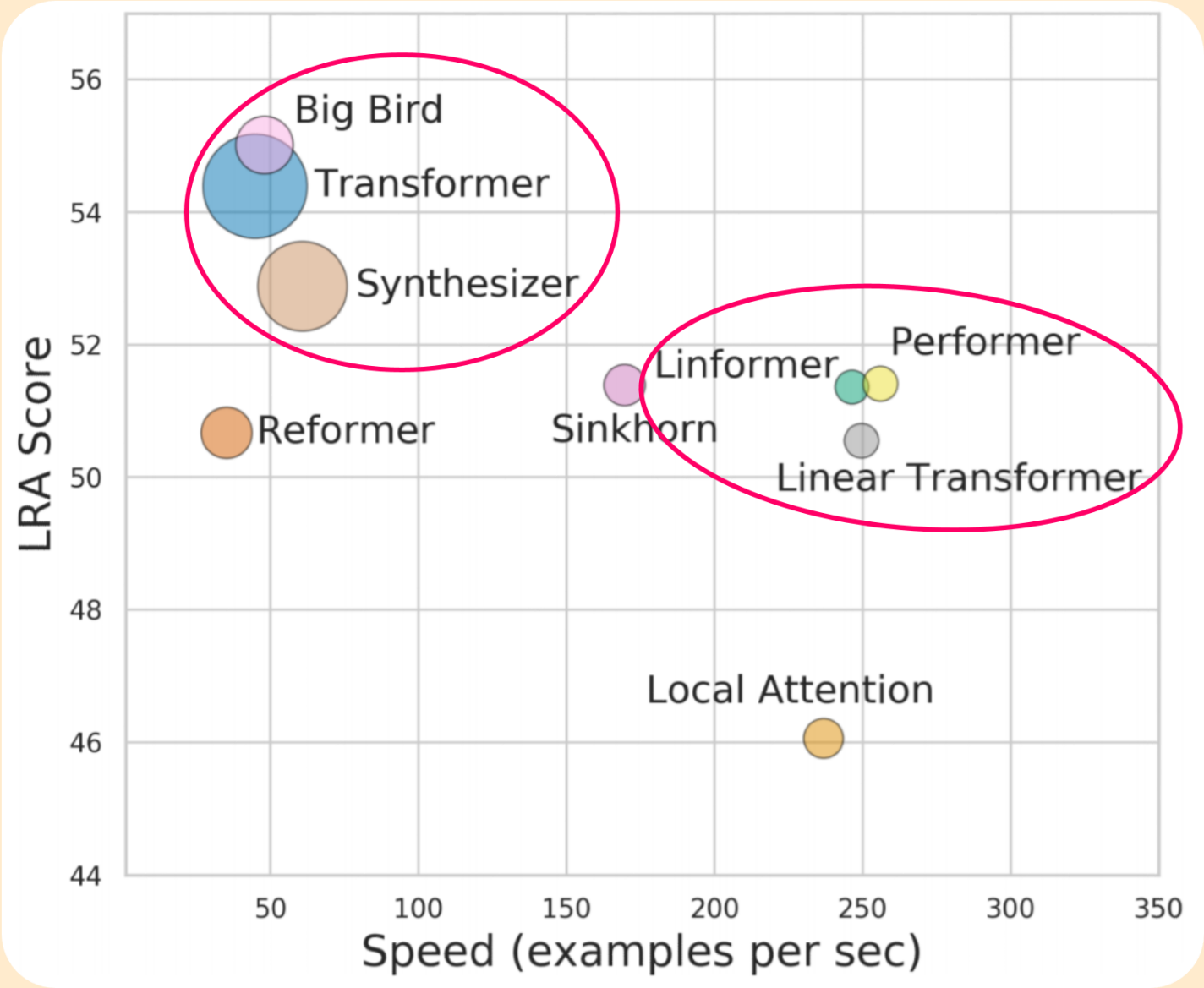
Task Score

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Transformer	36.37	64.27	57.46	42.44	71.40	FAIL	54.39
Local Attention	15.82	52.98	53.39	41.46	66.63	FAIL	46.06
Sparse Trans.	17.07	63.58	**59.59**	**44.24**	71.71	FAIL	51.24
Longformer	35.63	62.85	56.89	42.22	69.71	FAIL	53.46
Linformer	35.70	53.94	52.27	38.56	76.34	FAIL	51.36
Reformer	**37.27**	56.10	53.40	38.07	68.50	FAIL	50.67
Sinkhorn Trans.	33.67	61.20	53.83	41.23	67.45	FAIL	51.39
Synthesizer	36.99	61.68	54.67	41.61	69.45	FAIL	52.88
BigBird	36.05	64.02	59.29	40.83	74.87	FAIL	**55.01**
Linear Trans.	16.13	**65.90**	53.09	42.34	75.30	FAIL	50.55
Performer	18.01	65.40	53.82	42.77	**77.05**	FAIL	51.41
Task Avg (Std)	29 (9.7)	61 (4.6)	55 (2.6)	41 (1.8)	72 (3.7)	FAIL	52 (2.4)

Power

Model	Steps per second 1K	2K	3K	4K	Peak Memory Usage (GB) 1K	2K	3K	4K
Transformer	8.1	4.9	2.3	1.4	0.85	2.65	5.51	9.48
Local Attention	9.2 (1.1x)	8.4 (1.7x)	7.4 (3.2x)	7.4 (5.3x)	0.42	0.76	1.06	1.37
Linformer	9.3 (1.2x)	9.1 (1.9x)	8.5 (3.7x)	7.7 (5.5x)	**0.37**	**0.55**	0.99	**0.99**
Reformer	4.4 (0.5x)	2.2 (0.4x)	1.5 (0.7x)	1.1 (0.8x)	0.48	0.99	1.53	2.28
Sinkhorn Trans	9.1 (1.1x)	7.9 (1.6x)	6.6 (2.9x)	5.3 (3.8x)	0.47	0.83	1.13	1.48
Synthesizer	8.7 (1.1x)	5.7 (1.2x)	6.6 (2.9x)	1.9 (1.4x)	0.65	1.98	4.09	6.99
BigBird	7.4 (0.9x)	3.9 (0.8x)	2.7 (1.2x)	1.5 (1.1x)	0.77	1.49	2.18	2.88
Linear Trans.	9.1 (1.1x)	9.3 (1.9x)	8.6 (3.7x)	7.8 (5.6x)	**0.37**	0.57	**0.80**	1.03
Performer	**9.5** (1.2x)	**9.4** (1.9x)	**8.7** (3.8x)	**8.0** (5.7x)	**0.37**	0.59	0.82	1.06

Performance



Conclusion

- No Silver Bullet.
- The combination of Local Attention and Global Attention can maintain accuracy while reducing memory complexity, but there is a bottleneck in speed.

Conclusion

- The method based on Low Rank has excellent speed and low memory complexity on "long sequence tasks", but it is not suitable for autoregressive tasks.

- Long sequence autoregressive tasks are suitable for using Recurrence-based methods.

Problem

為何生成任務都要用自回歸模型？