RAT: Bridging RNN Efficiency and Attention Accuracy via Chunk-based Sequence Modeling




Xiuying Wei, Anunay Yadav, Razvan Pascanu, Caglar Gulcehre

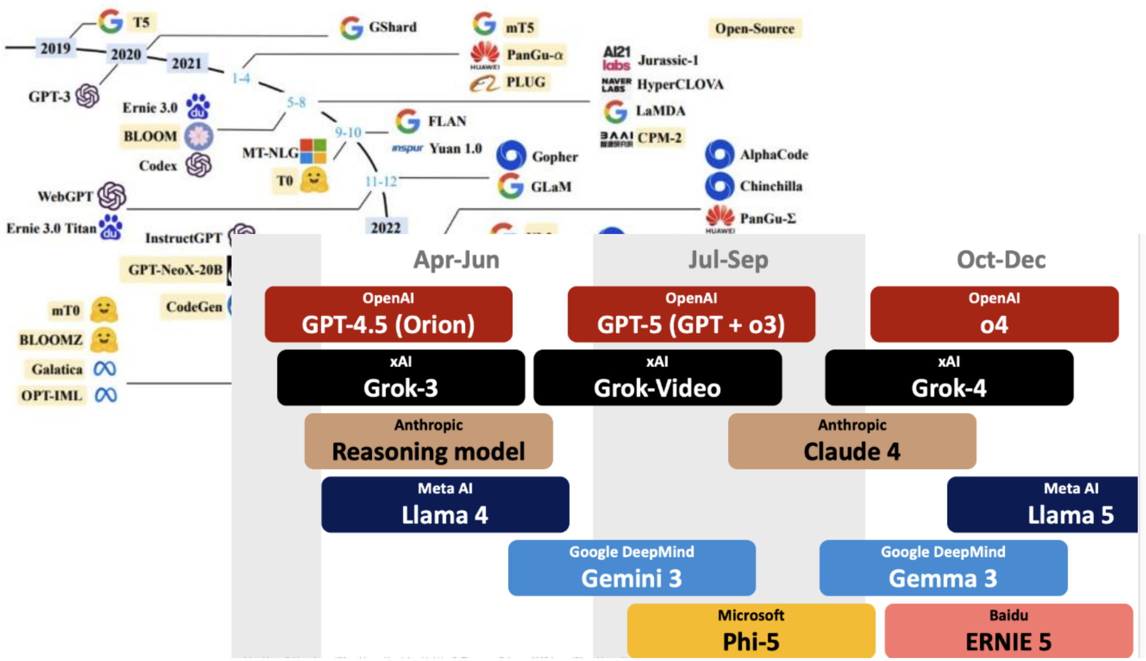
EPFL

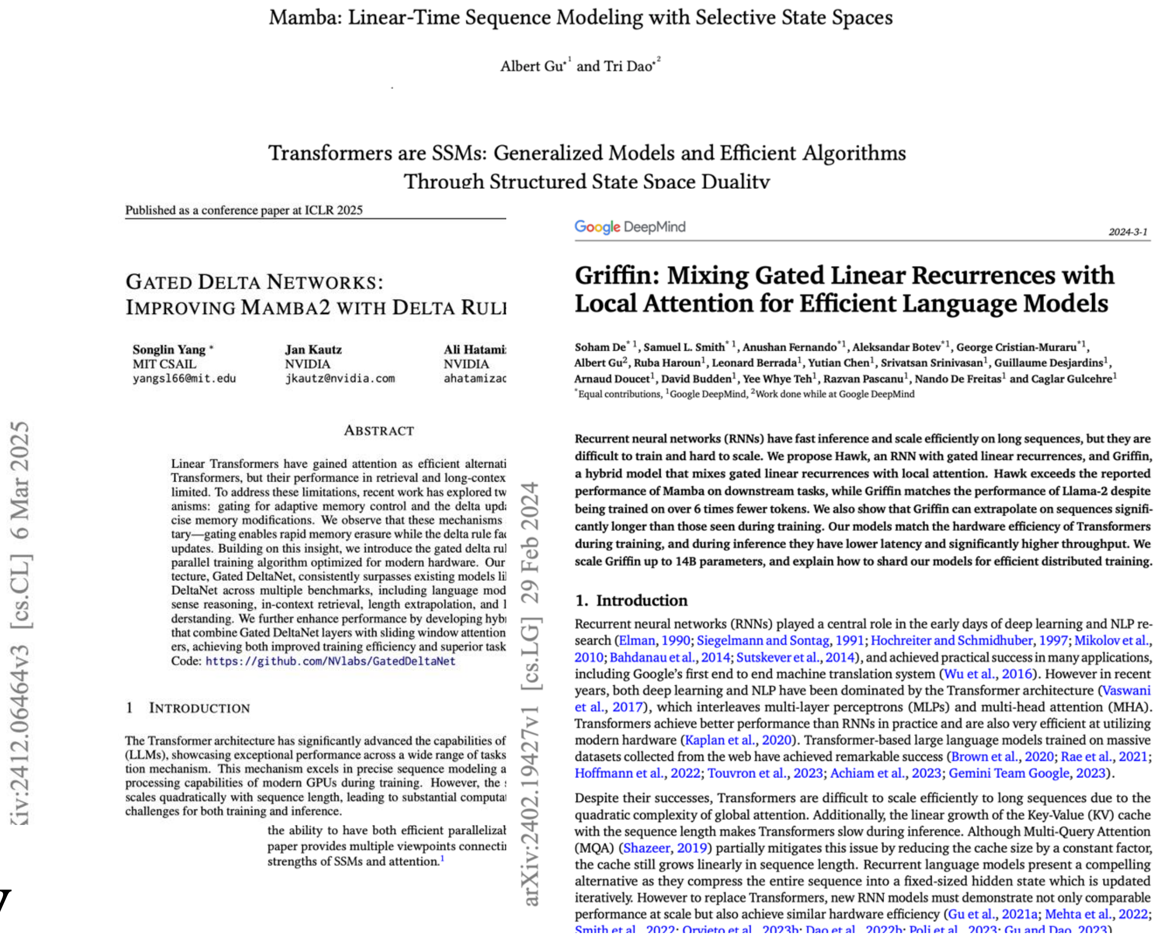
CLAIRE

Google DeepMind

NEURAL INFORMATION PROCESSING SYSTEMS

Many works on attention and recurrence!

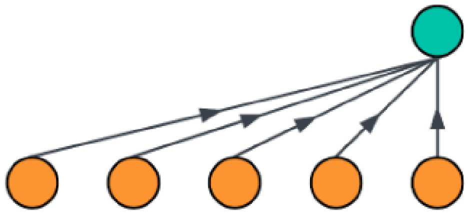




- Large language models emerge very quickly these years, and attention governs their architecture!

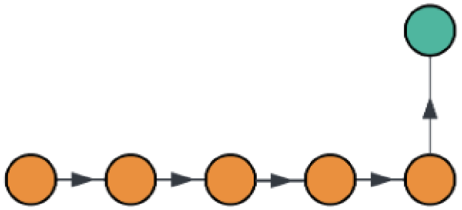
- A wave of recent efforts to revisit recurrent models or propose novel linear recurrent models

There lacks an intermediate design!



(1) Attention

Attention vs. Recurrence!

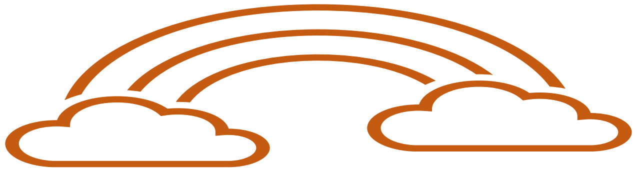


(2) RNN

- **Full-token access**
  - full-size memory and precise history retrieval
  - heavy computation

- **Full-sequence compression**
  - fixed-size and holistic representation
  - degraded memory and limited precise information retrieval
  - cheap computation

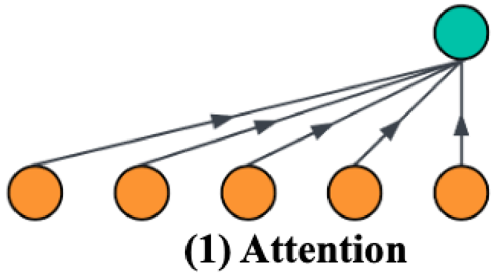
Strong performance!



High Efficiency!

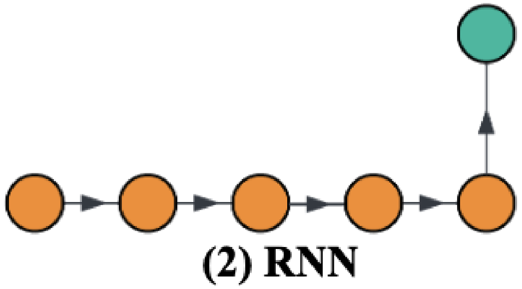
RAT: Chunk-Based Intermediate design

RAT: Chunk-Based Intermediate design

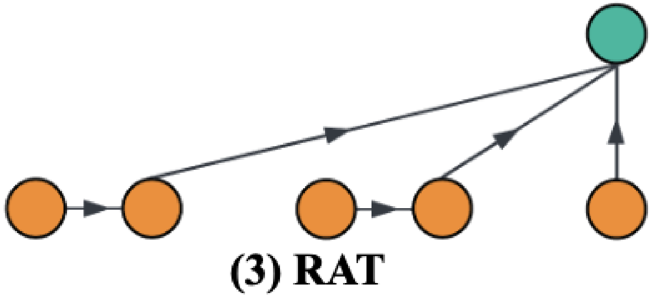


Chunk size as 1

- **Interpret the input as a sequence of shorter chunks**
  - Intra-chunk: Recurrence can excel on short sequences.
  - Inter-chunk: Attention has the direct distant access but with reduced computation.
  - Intermediate design by adjusting the chunk size.



Chunk size as the sequence length



Flexible chunk size: mitigate the fixed-size representation limitation of (2) and the inefficiency of (1)

RAT architecture

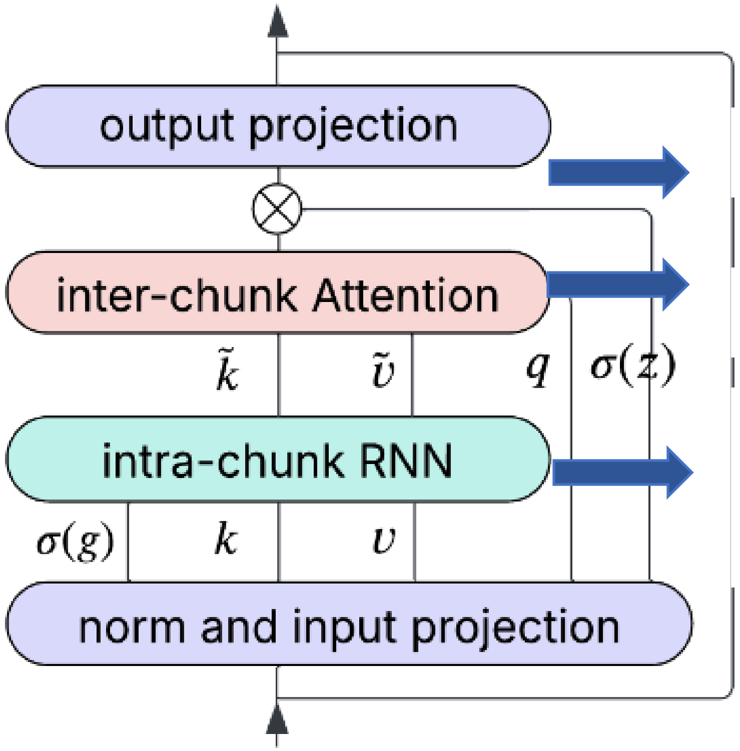
- **Attention**

$$\boldsymbol{y}_t = f(\boldsymbol{q}_t \boldsymbol{K}_:^\top)\boldsymbol{V}_:$$

- **Recurrence**: a simple linear recurrence [1, 2] but not limited to this

$$\tilde{\boldsymbol{v}}_t = \boldsymbol{g}_t \odot \tilde{\boldsymbol{v}}_{t-1} + (1-\boldsymbol{g}_t)\odot \boldsymbol{v}_t$$
$$\boldsymbol{y}_t = \boldsymbol{z}_t \odot \tilde{\boldsymbol{v}}_t,$$

- **RAT:** interpret a token t as chunk index and position within a chunk (c, l) .



$$\boldsymbol{y}_{c,l} = \boldsymbol{z}_{c,l} \odot \boldsymbol{y}_{c,l}$$

$$\boldsymbol{y}_{c,l} = f([\boldsymbol{q}_{c,l}\tilde{\boldsymbol{K}}_{:,-1}^\top; \boldsymbol{q}_{c,l}\tilde{\boldsymbol{k}}_{c,l}^\top])[\tilde{\boldsymbol{V}}_{:,-1}; \tilde{\boldsymbol{v}}_{c,l}]$$

$$\tilde{\boldsymbol{v}}_{c,l} = \boldsymbol{g}_{c,l} \odot \tilde{\boldsymbol{v}}_{c,l-1} + (1-\boldsymbol{g}_{c,l})\odot \boldsymbol{v}_{c,l}$$
$$\tilde{\boldsymbol{k}}_{c,l} = \boldsymbol{g}_{c,l} \odot \tilde{\boldsymbol{k}}_{c,l-1} + (1-\boldsymbol{g}_{c,l})\odot \boldsymbol{k}_{c,l}$$

[1]. Parallelizing linear recurrent neural nets over sequence length.
[2]. Were rnns all we needed?

RAT: scalable and efficient modeling

- **Design details**
  - Parameter allocations: decrease from $6D^2$ to $4D^2$.
  - Positional encoding: inter-chunk positions and better length generalization.
  - Hybrid design with local attention: long-range dependency and local region highlight!

- **Efficiency**
  - Reduced FLOPs: $\mathcal{O}(C \cdot D)$ of RAT, $\mathcal{O}(D)$ of recurrence, and $\mathcal{O}(T \cdot D)$ of attention.
  - Causal masking in training: online softmax
  - Easy impl. without customized kernels: flex attention and parallel scan in training, normal single step update and flash attention in inference
  - Compatible with parallelisms

D: model dimension
T: sequence length
C: number of chunks
L: chunk length

Efficiency and accuracy results!

Efficiency

Figure 2: **Latency of the temporal mixing block** (including linear projections) with a model dimension of 2048. (a): full-sequence latency with 200K tokens; (b): generation of 512 tokens at specified positions. We adopt *flash attention* for Attention.

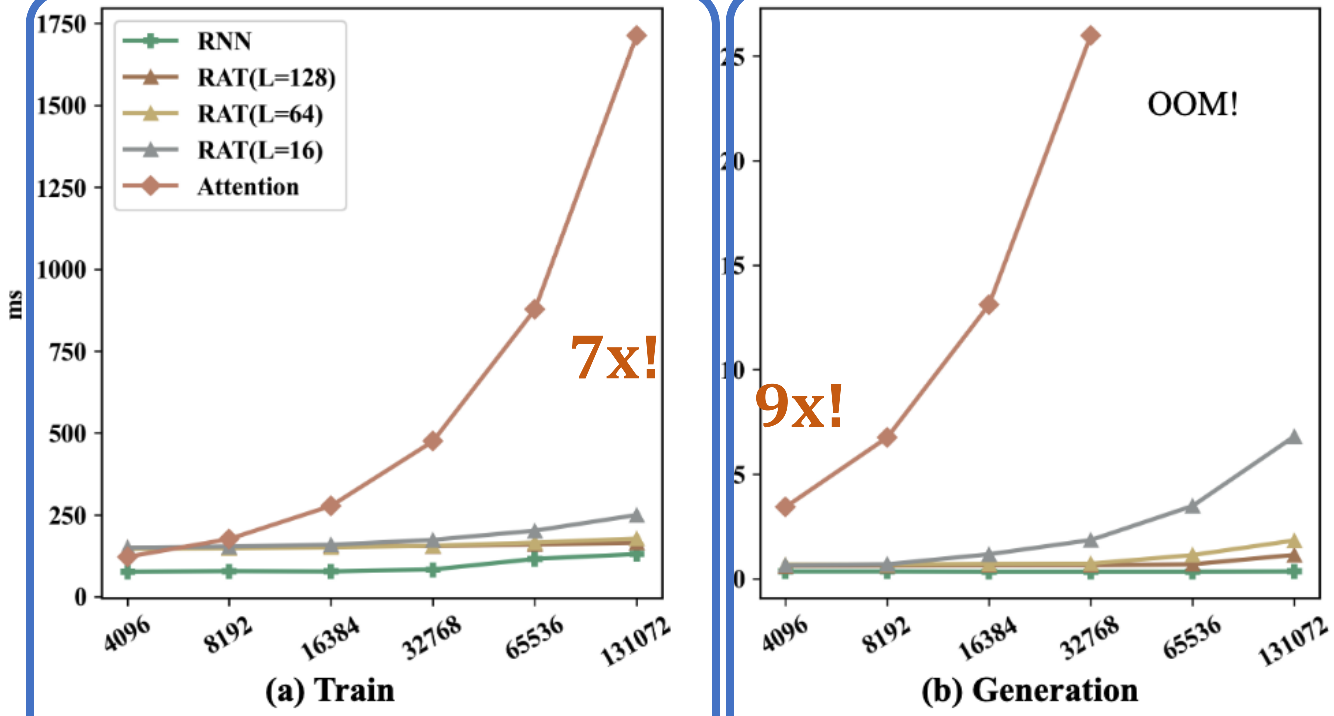


Table 2: **Maximum throughput of full models** (tokens/sec), measured by generating 1024 tokens from a 3072-token prompt. By reducing the KV cache memory and boosting speed, we achieve 10× maximum throughput compared to *flash attention* , and even more on 13B models, as attention suffers from poor GPU utilization at larger scale.

Model	**1.3B**	**7B**	**13B**
RAT(L=16)	31170	10103	5749
Attention	3152	983	534
Ratio	10.2×	10.3×	10.8×

Accuracy

Table 1: Representative results for 1.3B models across pretraining, direct evaluation, and SFT. -SWA denotes interleaving with sliding-window attention (SWA) (window size 1024). Maximum throughput is measured by generating 1024 tokens given a prompt of 3072 tokens on a H100 GPU in GH200 system. See Sec. 4 for details.

Model	**Throughput**	**Pretrain**	**Direct Evaluation**				**SFT**	
	token/sec	Val. PPL	CSR Avg. acc	SQA Avg. F1	Summ Avg. Rouge-L	Code Avg. EditSum	NQA[1] F1	QMSum Rouge-L
Attention	3052	7.61	56.9	18.2	<u>19.5</u>	<u>23.9</u>	61.3	<u>23.4</u>
`RAT(L=16)`	31170	7.67	56.7	**19.6**	**20.2**	17.4	60.8	23.3
Attention-SWA	4605	<u>7.61</u>	<u>57.1</u>	17.4	19.4	21.7	**63.3**	<u>23.4</u>
`RAT(L=16)-SWA`	13582	**7.57**	**58.0**	<u>18.8</u>	<u>19.5</u>	**28.2**	<u>63.2</u>	**24.6**

- 1.3B model
- 100B web token pretrain
- Commonsense reasoning: short context and general understanding
- Longbench: long context and instruction
- Supervised-finetuning

Takeaways

- Intermediate architecture between recurrence and attention by adjusting the chunk size
  - a single-layer design: trade-off between them.
  - hybrid modelling: greater flexibility with different chunk sizes.
- Memory capacity scales with sequence length with a fixed FLOPs reduction ratio
  - Either classic or advanced recurrence (state space or linear attention models) rely on fixed-size and holistic representations.
  - Partial compression with direct access to prior chunks allows precise retrieval.
- Work with local attention well
  - local attention highlights local computation while RAT focuses more on the long-range dependencies!

Thanks