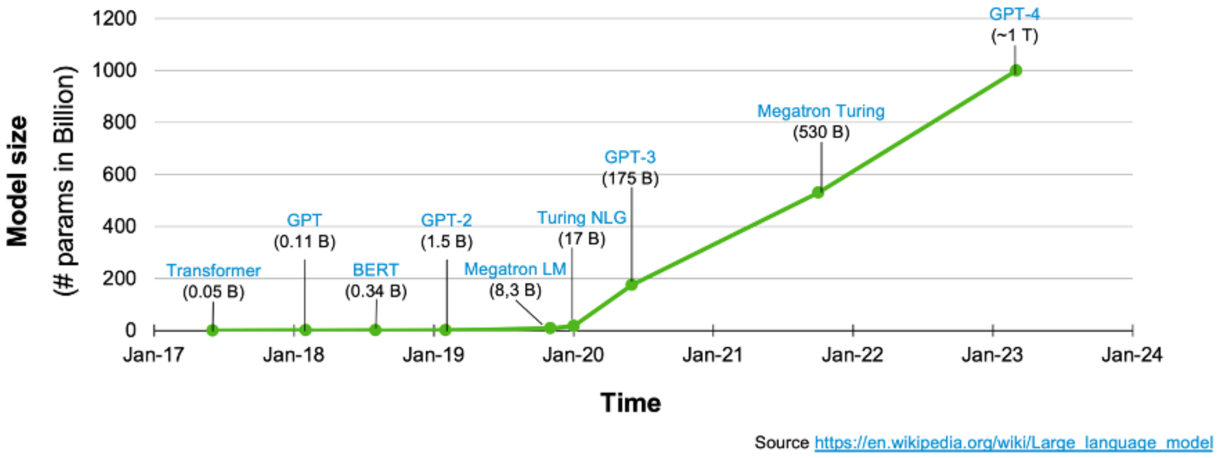
Building on Efficient Foundations: Effectively Training LLMs with Structured Feedforward Layers

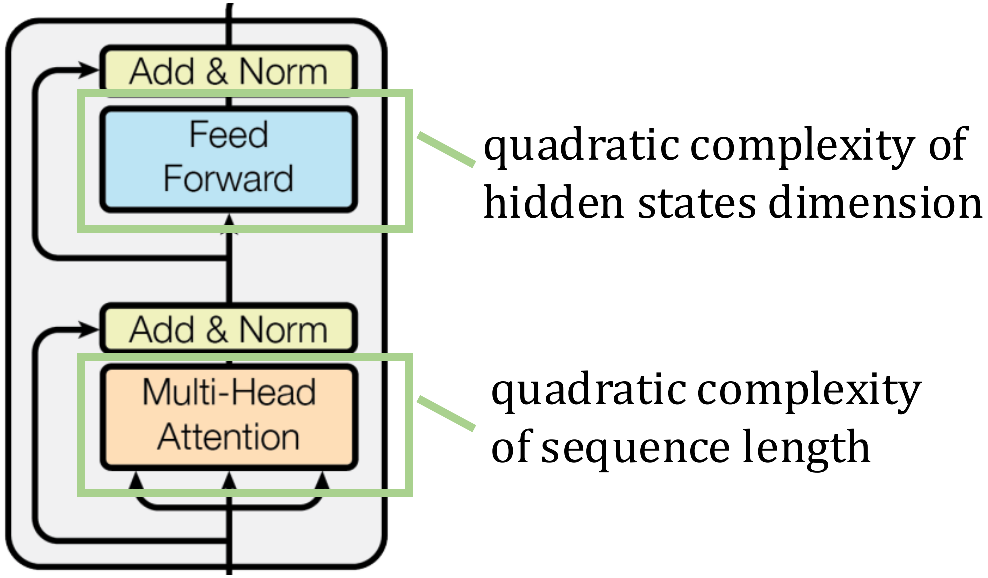



Xiuying Wei, Skander Moalla, Razvan Pascanu, Caglar Gulcehre

Today's models are becoming larger and larger



Model size grows in years



Transformer architecture

High pressure on both training and deployment

Efficient architectures!

Attention has been investigated much while FFN has not!

- Big FFN module!
  - over 60% of the Transformer’s parameters
  - 54% of total latency in a 1.3B
  - even bigger FFN size in Llama-3, Gemma

- Not many works on FFN training!
  - a key component for achieving strong performance [1, 2].
  - limited knowledge of structured matrices within FFN at a sufficient scale

[1]. FNet: Mixing Tokens with Fourier Transforms
[2]. Attention Is All You Need But You Don’t Need All Of It For Inference of Large Language Models

Structured matrices

Matrices	Example	#Params.	FLOPs	Examples of modern architectures
Dense $\boldsymbol{W}$	$\begin{pmatrix} 3 & 7 & 2 & 9 \\ 1 & 4 & 8 & 6 \\ 5 & 2 & 7 & 3 \\ 8 & 1 & 4 & 10 \end{pmatrix}$	N^2	$O(N^2)$	CNN [25], RNN [26, 19], Transformer [8, 4]
Low-rank $\boldsymbol{UV}$	$\begin{pmatrix} 2 \\ 3 \\ 5 \\ 1 \end{pmatrix} \begin{pmatrix} 7 & 4 & 9 & 1 \end{pmatrix}$	$2NR$	$O(NR)$	ScatterBrain [9], DeepSeek-V2 [10]
Diagonal $\boldsymbol{D}$	$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 9 \end{pmatrix}$	N	$O(N)$	ACDC [27], SSMs [12, 14]
Block-diagonal $\boldsymbol{K}$	$\begin{pmatrix} 2 & 1 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 5 & 6 \\ 0 & 0 & 7 & 8 \end{pmatrix}$	$\frac{N^2}{K}$	$O(\frac{N^2}{K})$	Monarch [3], Monarch Mixer [28], ShuffleNet [29]
Toeplitz $\boldsymbol{T}$	$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 1 & 2 & 3 \\ 6 & 5 & 1 & 2 \\ 7 & 6 & 5 & 1 \end{pmatrix}$	$2N-1$	$O(N \log N)$	TNN [18], Block-Toeplitz [30]
DFT $\boldsymbol{F}$	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}$	0	$O(N \log N)$	BPBP [31], F-Net [2], GFNet [32]

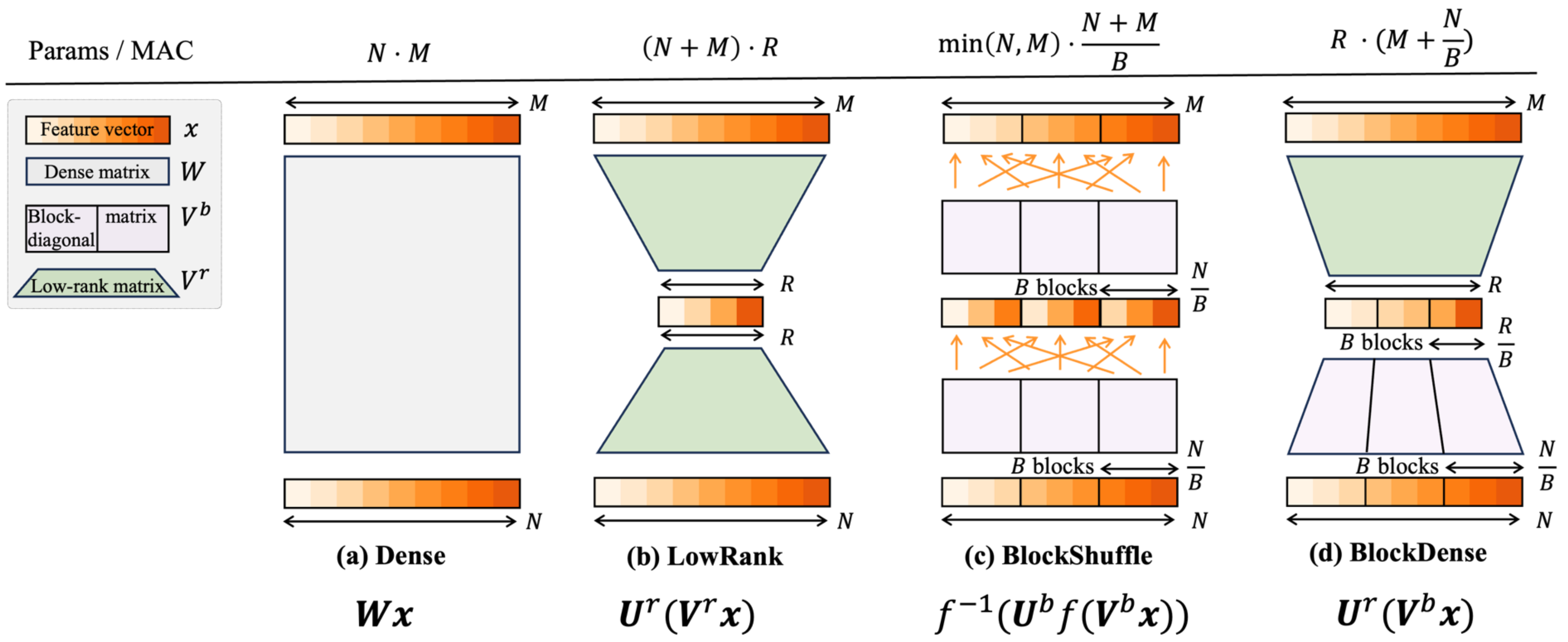
They have not yet been thoroughly explored at a sufficient scale in modern LLM architecture training

Outline

- Three structured matrices for FFN module in pretraining transformer language models → Good scaling performance
- Efficiency study across various scenarios → Pre-merge technique
- Optimization challenges → Self-guided training

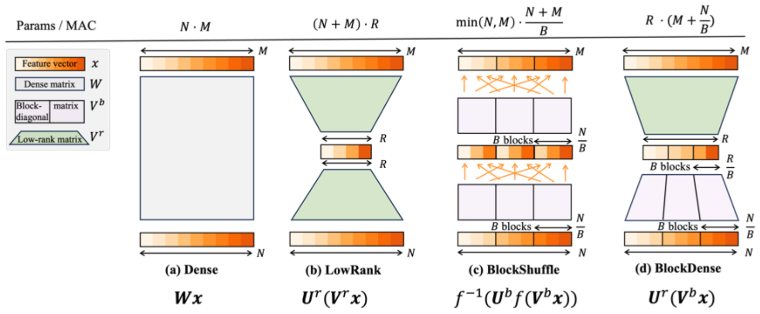
Method

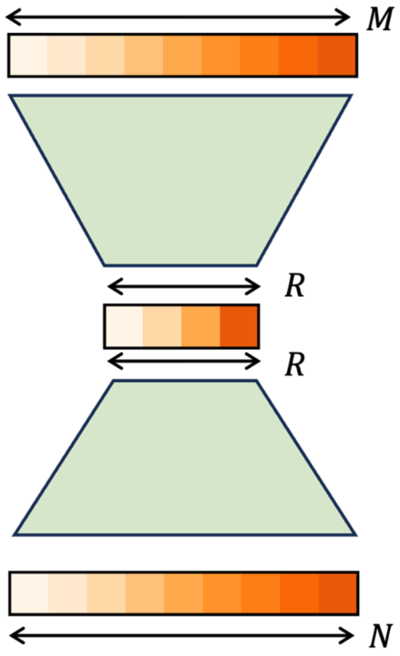
Three structured matrices for efficient and accurate FFN training



Superscript r: low-rank projection
Superscript b: block-diagonal projection

Three structured matrices: LowRank





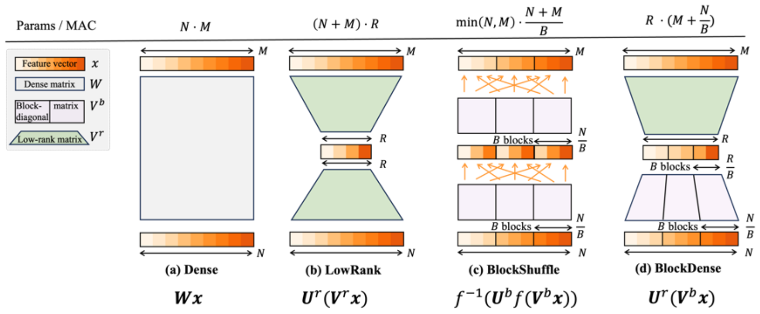
(b) LowRank $\boldsymbol{U}^r(\boldsymbol{V}^r\boldsymbol{x})$

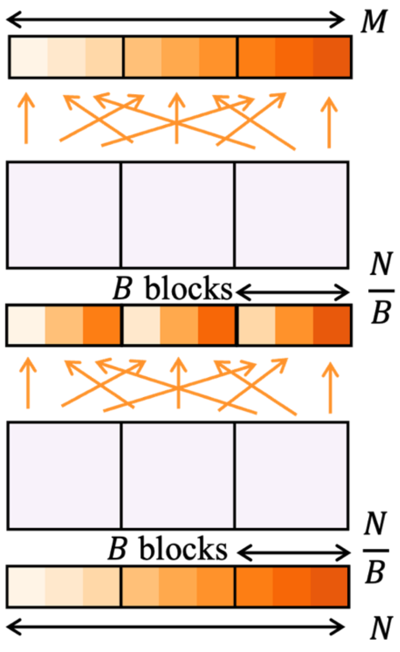
Params./MAC
$M \cdot N \rightarrow (N + M) \cdot R$

[1]. The truth is in there:Improvingreasoning in language models with layer-selective rank reduction
[2]. Lora: Low-rank adaptation of large language models.
[3]. Implicit regularization in deep matrix factorization

Superscript r: low-rank projection
Superscript b: block-diagonal projection

Three structured matrices: BlockShuffle





(c) BlockShuffle

Params./MAC

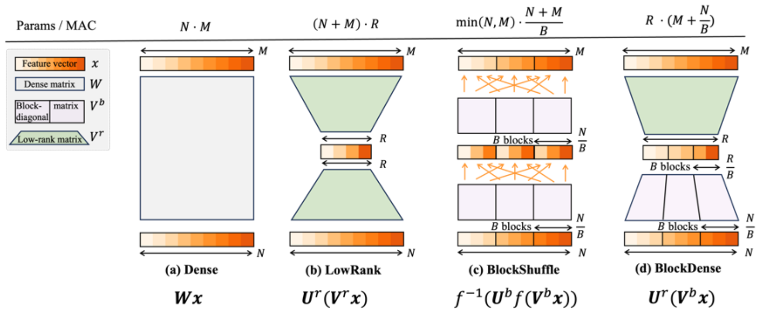
$$M \cdot N \;\rightarrow \min(N, M) \cdot \frac{N+M}{B}$$

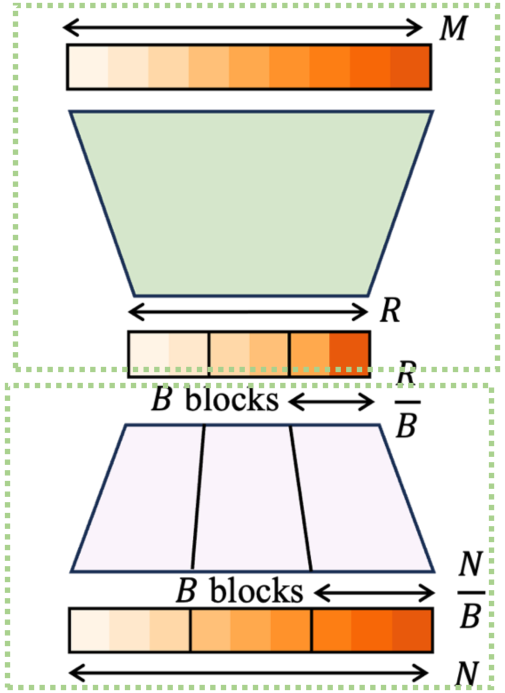
$$f^{-1}(\boldsymbol{U}^b f(\boldsymbol{V}^b \boldsymbol{x}))$$

[1]. Monarch: Expressive structured matrices for efficient and accurate training.
[2]. Shufflenet: An extremely efficient convolutional neural network for mobile devices.
[3]. Mobilenetv2: Inverted residuals and linear bottlenecks

Superscript r: low-rank projection
Superscript b: block-diagonal projection

Three structured matrices: BlockDense





(d) BlockDense

Params./MAC

$$M \cdot N \ \rightarrow R \cdot (M + \frac{N}{B})$$

$$\boldsymbol{U}^r(\boldsymbol{V}^b\boldsymbol{x})$$

Superscript r: low-rank projection
Superscript b: block-diagonal projection

Maintaining efficiency during online decoding

- Big T
  - Training, prefilling, decoding with a big batch size
  - Reduced FLOPs and parameters can lead to real efficiency gain

- Small T
  - Parallelism-bound FFN during online decoding
  - Structured parametrization may lead to worse latency performance

- Pre-merge technique
  - Benefited from non-linearity
  - Dynamically decide to use $(\boldsymbol{UV})\boldsymbol{x}$ or $\boldsymbol{U}(\boldsymbol{Vx})$

T: number of batches of tokens

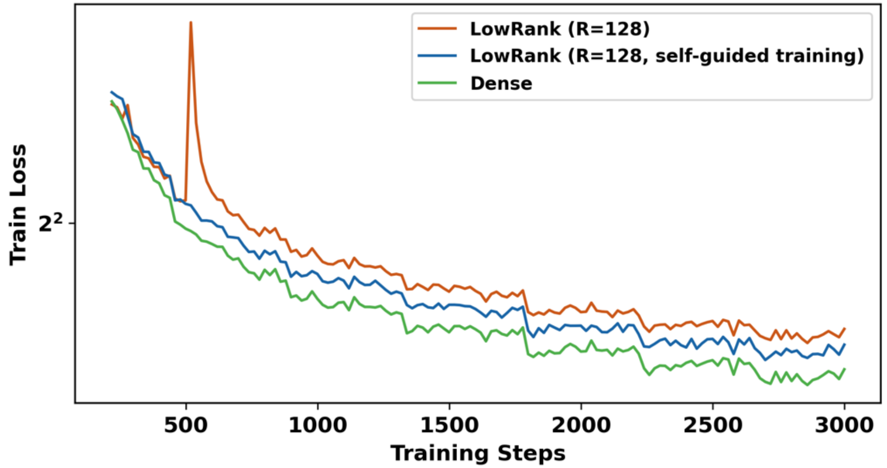
Addressing the optimization challenge

- More difficulties in training structured matrices
  - additional symmetries can lead to poor training dynamics

Training

$\boldsymbol{U}(\boldsymbol{V}\boldsymbol{x})$ 

$\boldsymbol{W}\boldsymbol{x}$ $\quad 0.3\boldsymbol{W}\boldsymbol{x} + 0.7\boldsymbol{U}(\boldsymbol{V}\boldsymbol{x})$ $\quad \boldsymbol{U}(\boldsymbol{V}\boldsymbol{x})$ 

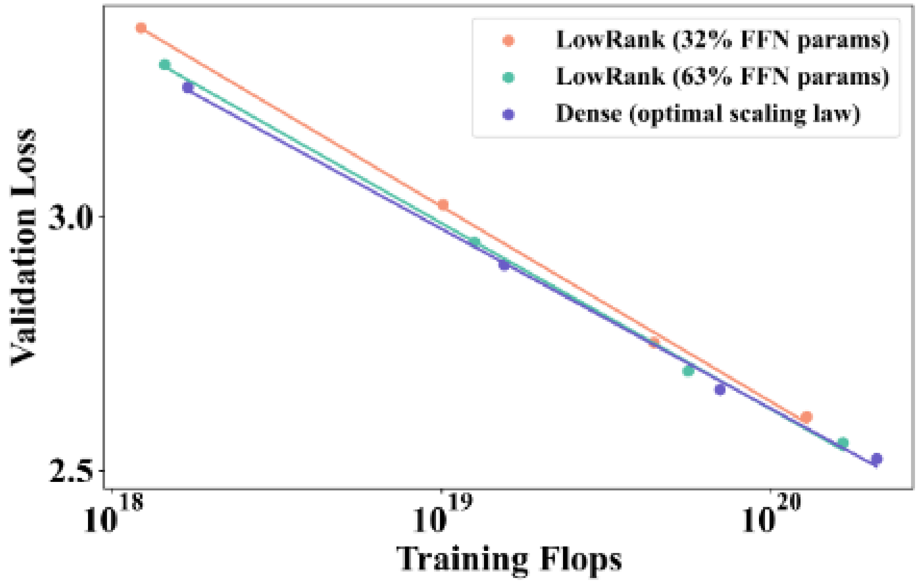


- Self-guided training
  - $\boldsymbol{o} = \alpha\boldsymbol{W}\boldsymbol{x} + (1-\alpha)\boldsymbol{U}(\boldsymbol{V}\boldsymbol{x})$, where α decays following a cosine scheduler

[1]. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks.
[2]. Neural networks and principal component analysis: Learning from examples without local minima.

Results: Scaling analyses

Scaling law study: better training FLOPs utilization



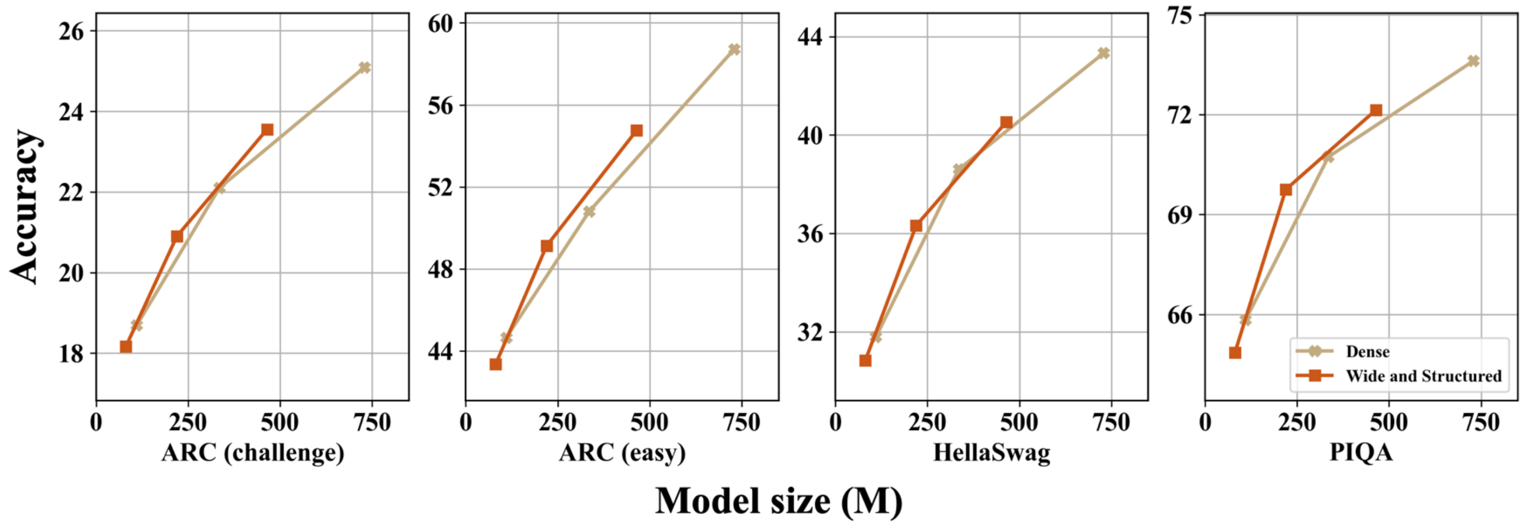
(a) LowRank

- **Steeper scaling curves of Structured FFN up to 1.3B models:** when the x-axis is further extended, we can have fewer parameters and predict significantly smaller loss per FLOP.

- **Better training FLOPs utilization of the Wide and Structured network:** lower perplexity while using much fewer parameters

Method	#Param	Training FLOPs	PPL	TP (token/s)
Transformer-m	335M	1.55e+19	18.29	30229
Transformer-m (GQA)	335M	1.55e+19	18.23	84202
Wide and Structured	**219M**	1.55e+19	**17.89**	**91147 (8% ↑)**
Transformer-l	729M	7.03e+19	14.29	23351
Transformer-l (GQA)	729M	7.03e+19	14.40	64737
Wide and Structured	**464M**	7.03e+19	**14.27**	**75930 (17% ↑)**

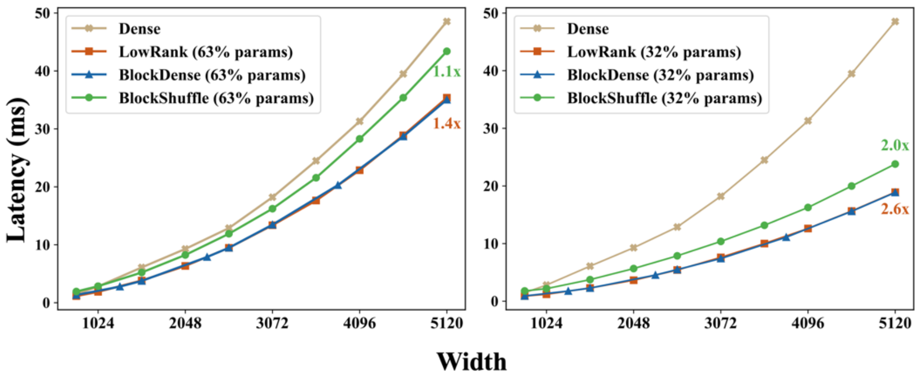
Scaling model size: better downstream performance



- **Good scaling trend of wide and structured networks** in the over-training regime i.e., 300B tokens.

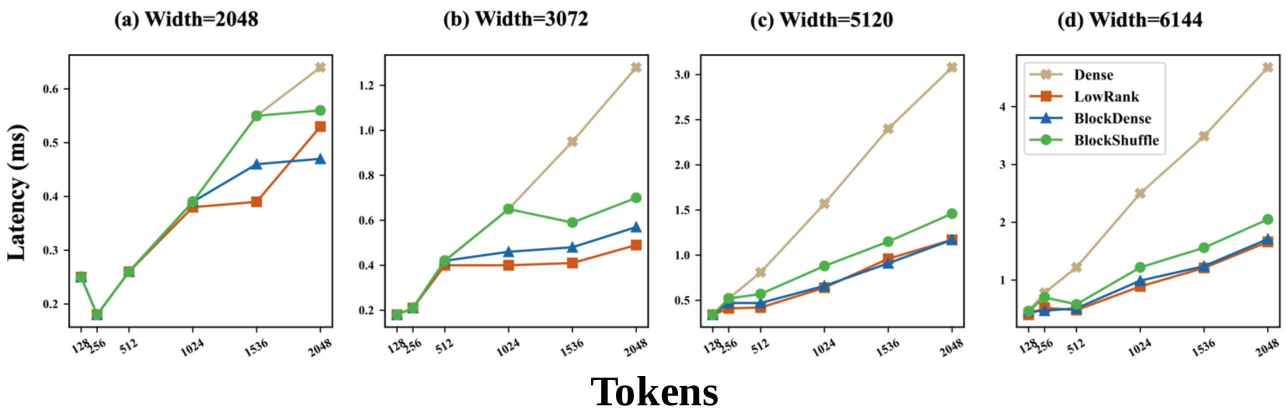
Results: Efficiency

- Real efficiency gain in Big T case



- BlockShuffle can be slower due to additional shuffle operations.
- The other two have 1.4x and 2.6x speed-up with 63% and 32% FFN parameters

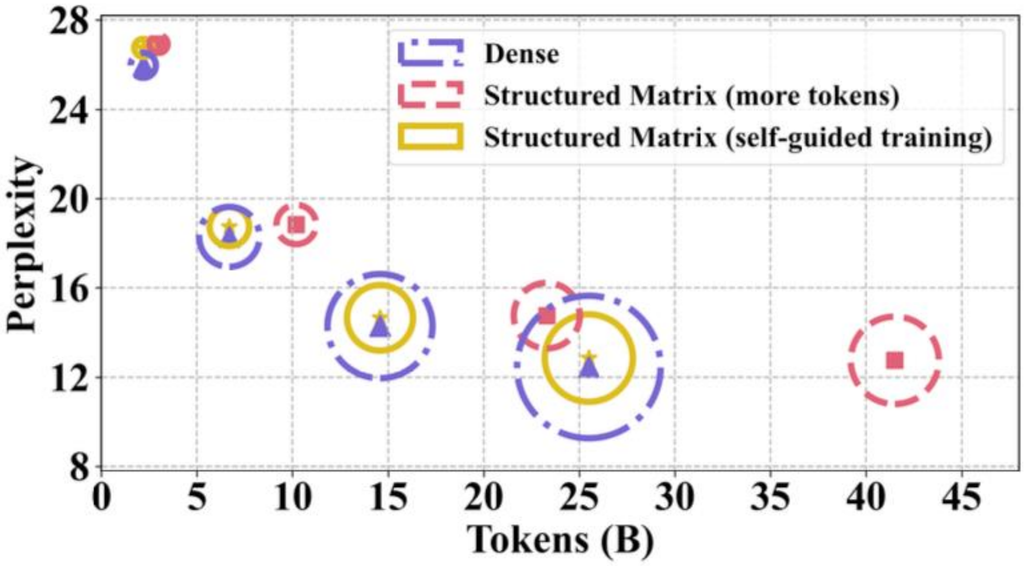
- Small T with the pre-merge technique



- With a 2048-width FFN, it is difficult to fully utilize resources on GPU with limited tokens.
- With a width 5120 and 6144, 2.81× acceleration of BlockDense with 32% parameters on T = 1536.

Results: self-guided training

Architecture	FFN	Training FLOPs	PPL
Transformer-m	201M	1.55e+19	18.29
LowRank	69M	1.01e+19	20.60
LowRank♣		1.21e+19	**19.90**
BlockDense	65M	1.00e+19	20.85
BlockDense♣		1.19e+19	**20.10**
BlockShuffle	69M	1.01e+19	21.12
BlockShuffle♣		1.21e+19	**20.36**



- **Apply self-guided training during the first half of training**: consistently reduces loss for all efficient parametrizations

- **Apply self-guided training with matched training FLOPs**: close performance between structured FFN with 32% parameters and dense models.

Conclusion

- Scope of our study
  - from a training-from-scratch perspective
  - scales up models to 1.3B parameters
  - conducted within recent Transformer-based LLMs not convolutional architectures.
- Research Objective
  - not aimed at identifying the "best" structured matrix
  - Investigate common properties of structured matrices: scaling, efficiency and optimization
- Proposed Techniques
  - Pre-merge training
  - Self-guided training

Thanks