LaplacianFormer:Rethinking Linear Attention with Laplacian Kernel

Zhe Feng[12]* · Sen Lian[3]* · Changwei Wang[56]† · Muyang Zhang[21] ·
Tianlong Tan[4] · Rongtao Xu[7] · Weiliang Meng[12]† · Xiaopeng Zhang[12]

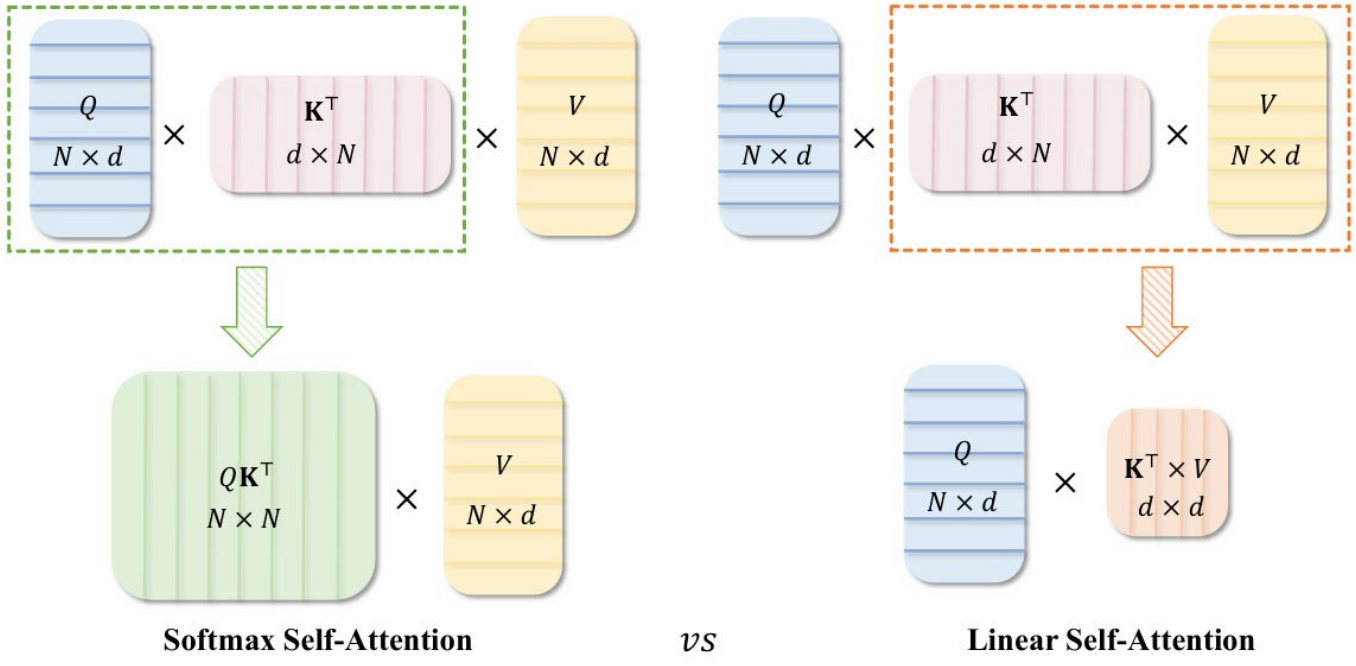
[1] MAIS, Institute of Automation, CAS
[2] University of Chinese Academy of Sciences
[3] China Electronics Data Corporation
[4] Institute of Computing Technology, CAS
[5] Qilu University of Technology
[6] Shandong Fundamental Research Center
[7] Spatiotemporal AI

* Equal contribution
† Corresponding author

Contents

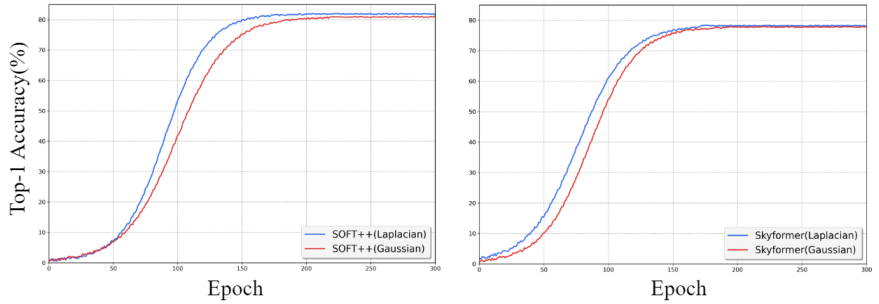
Introduction

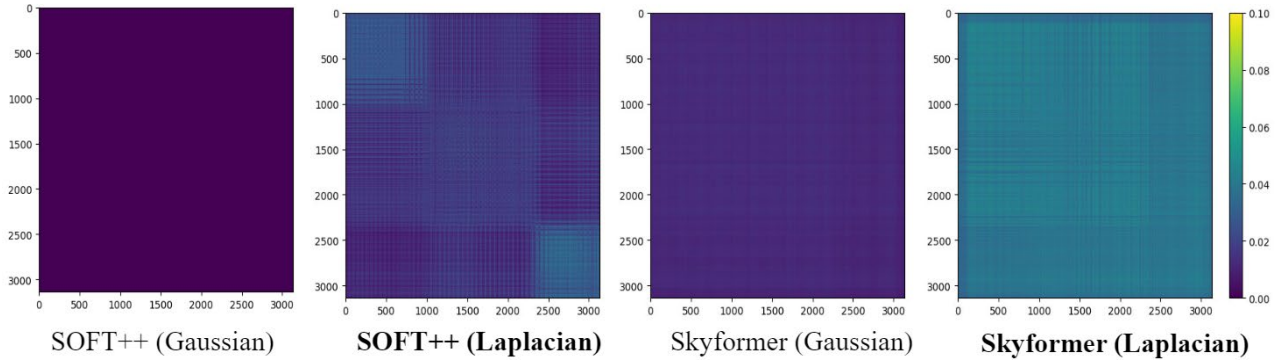
- Transformer is powerful for vision.
- Softmax attention has $\mathcal{O}(N^2)$ complexity.
- Linear attention improves scalability.
- But kernel choice remains underexplored.



Motivation

- Existing linear attention mostly uses Gaussian kernels.
- Empirical Q/K distances are heavy-tailed.
- Gaussian decays too aggressively and suppresses useful mid-range interactions.
- Laplacian kernel better matches sparse and heavy-tailed token relations.





Problem Statement

- How to design a better kernel for linear attention?
- How to preserve expressiveness under low-rank approximation?
- How to implement it efficiently on GPU?

Methodology

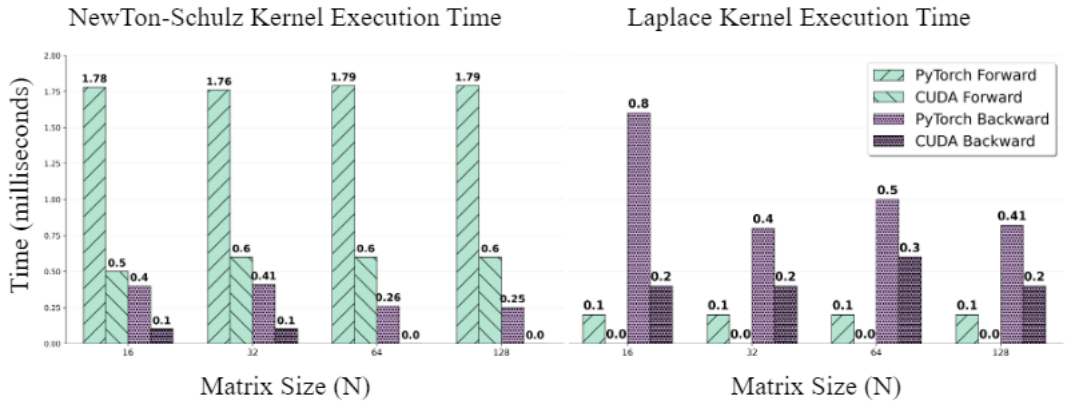
- Laplacian kernel attention
- Injective normalized feature map
- Nyström approximation for low-rank efficiency
- Newton–Schulz iteration for fast inverse approximation
- CUDA implementation for practical acceleration

Algorithm 1 Laplacian Kernel with Nyström Approximation

Input: Queries $Q \in \mathbb{R}^{N \times d}$, Keys $K \in \mathbb{R}^{N \times d}$, Nyström sampling function f_s
Sampling: $\tilde{Q}, \tilde{K} \leftarrow f_s(Q), f_s(K)$ ▷ Select $m \ll n$ landmark points
$W \leftarrow \exp\left(-\frac{1}{\lambda}\|\tilde{Q} \bigcirc \tilde{K}\|_1\right)$ ▷ Kernel matrix on sampled points
$C \leftarrow \exp\left(-\frac{1}{\lambda}\|Q \bigcirc \tilde{K}\|_1\right)$ ▷ Cross-kernel between all queries and landmarks
$\hat{G} \leftarrow CW^{\dagger}C^{\top}$ ▷ Low-rank approximation of full kernel matrix
Output: $\hat{G}$

Algorithm 2 Newton–Schulz Iteration for Approximating $\mathbf{W}^{\dagger}$

Input: Landmark kernel matrix $\mathbf{W} \in \mathbb{R}^{m \times m}$, number of iterations $\mathcal{T} \in \mathbb{Z}^{+}$
Add small perturbation: $\mathbf{W} \leftarrow \mathbf{W} + \epsilon\mathbf{I}$, where $\epsilon > 0$
Initialize scaling factor: $\alpha \leftarrow \frac{2}{\|\mathbf{W}\|_2}$
Initialize: $\mathbf{X}_0 \leftarrow \alpha\mathbf{W}^{\top}$
for $k = 1$ to $\mathcal{T}$ **do**
    $\mathbf{X}_k \leftarrow \mathbf{X}_{k-1}(2\mathbf{I} - \mathbf{W}\mathbf{X}_{k-1})$
end for
Output: Approximate pseudoinverse $\mathbf{X}_{\mathcal{T}} \approx \mathbf{W}^{\dagger}$



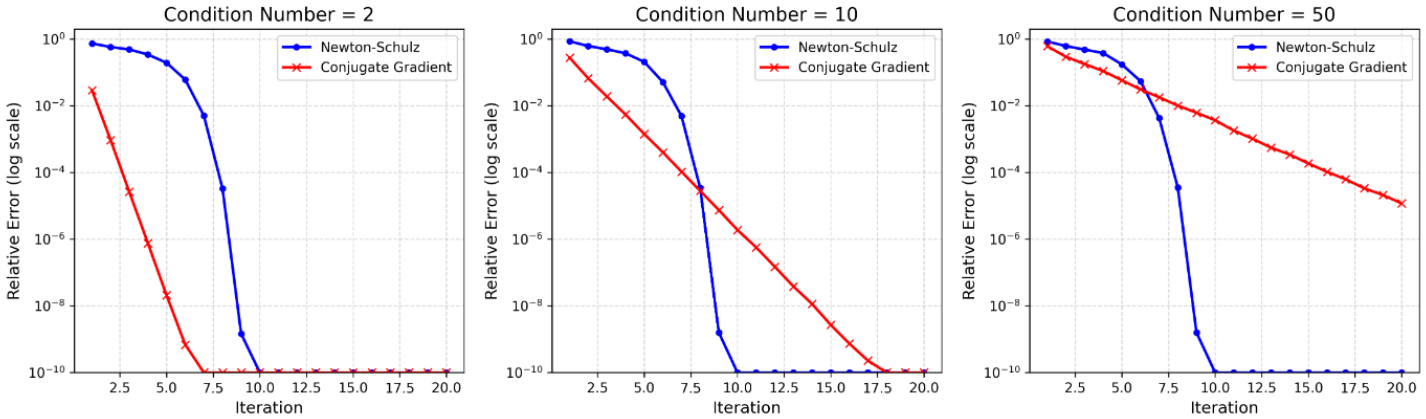
Results

- Better accuracy-efficiency trade-off on ImageNet
- Linear memory scaling with sequence length
- Faster convergence than Gaussian-kernel counterparts
- Competitive downstream performance on detection/segmentation

FLOPs range	Model	Params	FLOPs	Top-1 %↑	Image Size
< 3G	Agent-Deit-T Han et al. (2024c)	6.0M	1.2G	74.9	224
	VRWKV-T Duan et al. (2025)	6.2M	1.2G	75.1	256
	PVT-T-PolaFormer Meng et al. (2025)	12M	2.0G	78.8	224
	FL-PVTv2-B1 Han et al. (2023)	13M	2.2G	79.5	224
	BiFormer-T Zhu et al. (2023)	13.1M	2.2G	81.4	224
	LaplacianFormer-Tiny	12.1M	2.1G	**81.4**	224
3∼8G	InLine-CSwin-S Han et al. (2024a)	33M	6.8G	83.8	224
	SViT-S Huang et al. (2023)	25M	4.4G	83.6	224
	BiFormer-S Zhu et al. (2023)	25.5M	4.5G	83.8	224
	HiViT-T Zhang et al. (2023b)	19M	4.6G	82.1	224
	Agent-PVT-S Han et al. (2024c)	20.6M	4.0G	82.2	224
	LaplacianFormer-Small	25.7M	4.8G	**83.8**	224
8∼10G	SViT-B Huang et al. (2023)	52M	9.9G	84.8	224
	SOFT++-Medium Lu et al. (2024)	45M	7.2G	83.7	224
	BiFormer-B Zhu et al. (2023)	56.8M	9.8G	84.3	224
	Swin-S-PolaFormer Meng et al. (2025)	50M	8.7G	83.6	224
	SLAB-Swin-S Guo et al. (2024)	-	8.7G	81.8	224
	LaplacianFormer-Medium	46.3M	7.43G	**85.3**	224
10∼14G	StructViT-B-8-1 Kim et al. (2024)	52M	12G	84.3	224
	SOFT++-Large Lu et al. (2024)	64M	11G	84.1	224
	NAT-B Hassani et al. (2023)	90M	13.7G	84.3	224
	MogaNet-L Li et al. (2024)	82.5M	15.9G	84.7	224
	FLatten-CSwin-S Han et al. (2023)	35M	6.9G	83.6	224
	LaplacianFormer-Large	63.1M	11.2G	**85.6**	224
>14G	VRWKV-B Duan et al. (2025)	93.7M	18.2G	82.0	224
	SViT-L Huang et al. (2023)	95M	15.6G	85.3	224
	MLLA-B Han et al. (2024b)	96M	16.2G	85.3	224
	HiViT-B Zhang et al. (2023b)	66M	15.9G	83.8	224
	LaplacianFormer-Huge	78.5M	15.5G	**85.8**	224

Backbone	Mask R-CNN 1×						RetinaNet 1×					
	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	AP^b	AP^b_{50}	AP^b_{75}	AP^b_S	AP^b_M	AP^b_L
Swin-T-PRepBN Guo et al. (2024)	42.9	65.8	46.8	39.3	62.6	41.9	–	–	–	–	–	–
FL-PVT-T Han et al. (2023)	38.2	61.6	41.9	37.0	57.6	39.0	–	–	–	–	–	–
SOFT++-Tiny Lu et al. (2024)	41.2	63.7	44.7	38.2	61.0	41.0	41.9	62.7	44.7	27.8	45.4	55.6
LaplacianFormer-Tiny	**43.2**	**66.1**	**47.2**	**40.3**	**63.0**	**42.9**	**42.5**	**64.1**	**46.4**	**29.1**	**46.9**	**57.8**
PVT-S-PolaFormer Meng et al. (2025)	43.9	66.1	47.9	40.2	63.1	43.0	43.2	64.1	46.4	–	–	–
InLine-PVT-S Han et al. (2024a)	43.4	66.4	47.1	40.1	63.1	43.3	–	–	–	–	–	–
SOFT++-Small Lu et al. (2024)	43.8	66.0	47.5	40.1	63.0	43.0	43.7	64.9	46.8	28.7	47.4	57.6
LaplacianFormer-Small	**45.8**	**68.2**	**49.8**	**42.0**	**65.1**	**45.2**	**45.5**	**66.8**	**49.1**	**30.7**	**51.8**	**59.5**
Agent-PVT-M Han et al. (2024c)	45.9	67.8	50.4	42.0	65.0	45.4	–	–	–	–	–	–
FL-Swin-M Han et al. (2023)	44.0	66.4	48.0	40.3	63.4	43.5	–	–	–	–	–	–
SOFT++-Medium Lu et al. (2024)	46.6	67.8	51.2	42.0	64.8	45.2	44.3	64.7	47.4	29.0	48.2	59.9
LaplacianFormer-Medium	**48.0**	**70.3**	**52.5**	**43.5**	**65.8**	**46.5**	**47.2**	**68.5**	**51.5**	**31.8**	**53.0**	**61.4**
Swin-T-PolaFormer Meng et al. (2025)	44.8	67.6	49.1	40.5	64.1	43.5	–	–	–	–	–	–
Agent-PVT-L Han et al. (2024c)	46.9	69.2	51.4	42.8	66.2	46.2	–	–	–	–	–	–
SOFT++-Large Lu et al. (2024)	47.0	68.3	51.7	42.2	65.2	45.4	47.0	67.8	50.4	30.2	50.9	62.0
LaplacianFormer-Large	**48.2**	**70.5**	**53.0**	**43.8**	**67.1**	**47.4**	**48.5**	**69.3**	**52.4**	**32.6**	**52.3**	**63.8**

Results



Model	CG (%)	NS (%)
LaplacianFormer-Tiny	79.2	**81.4**
LaplacianFormer-Small	81.4	**83.8**

λ	0.5	1	2	4	8
Top-1 Acc (%) ↑	79.4	79.6	80.1	**81.4**	78.5

Conclusion

- Laplacian is a better kernel choice than Gaussian for linear attention
- Injective representation improves expressiveness
- Efficient approximation and CUDA implementation make deployment practical
- *Limitations & Future Works:*
  - Developing learnable or adaptive kernel functions.
  - Exploring task-specific kernel parameterization.


ICLR

中国科学院大学
University of Chinese Academy of Sciences

中国科学院自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES
CEC 中国电子

Thank you!

The Fourteenth International Conference on Learning Representations
Rio de Janeiro, Brazil