Quasi Linear Kernel Attention with Infinite Capacity

Expressive attention in $\mathcal{O}(N \log N)$ — exact, not approximated!

Nicolaj Rux[1] • **Johannes Hertrich**[2] • **Sebastian Neumayer**[1]

[1]Faculty of Mathematics, Chemnitz University of Technology, Germany [2]Institute of Computer Science, University of Göttingen, Germany

UNIVERSITY OF TECHNOLOGY IN THE EUROPEAN CAPITAL OF CULTURE CHEMNITZ

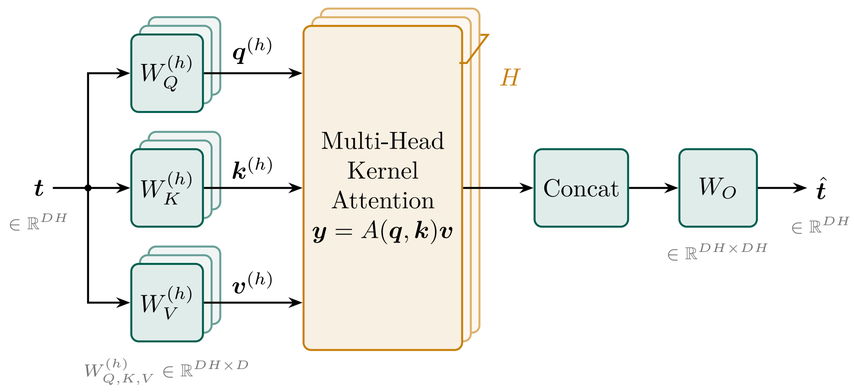
arXiv Preprint

1 The Quadratic Attention Bottleneck

- *Kernel attention* for queries $q_n \in \mathbb{R}^D$, keys $k_n \in \mathbb{R}^D$, values $v_n \in \mathbb{R}^D$, $n = 1, \dots, N$:

$$y_m = \frac{\sum_{n=1}^N \Phi(q_m, k_n)\, v_n}{\sum_{n=1}^N \Phi(q_m, k_n)}, \qquad m = 1, \dots, N.$$

- Standard: *softmax* $\Phi(q,k) = e^{q^\top k}$ or *Gauss* $\Phi(q,k) = e^{-\frac{1}{2}\|q-k\|_2^2}$.
- Keys of equal norm ⇒ softmax = Gauss attention, since $e^{q^\top k} = e^{-\frac{1}{2}\|q-k\|_2^2} e^{\frac{1}{2}(\|q\|_2^2+\|k\|_2^2)}$
- Cost $\mathcal{O}(HDN^2)$: *quadratic* in the sequence length N.
- Typically $D \in \{32, 64, 128\}$ and N ranges from 64 up to 100 000.



2 Finite Feature Maps

- $\Phi(q,k) = \varphi(q)^\top \varphi(k)$ with *finite feature map* (FFM) $\varphi : \mathbb{R}^D \to \mathbb{R}^L$.
- Reorder the sum: the bracket is shared by all queries:

$$z_m = \sum_{n=1}^N \Phi(q_m, k_n) v_n = \sum_{n=1}^N \varphi(q_m)^\top \varphi(k_n) v_n = \Big(\sum_{n=1}^N v_n\, \varphi(k_n)^\top\Big)\varphi(q_m).$$

- *Linear* cost $\mathcal{O}(HDLN)$ instead of $\mathcal{O}(HDN^2)$.
- Ex.: `elu` (L=D) (Katharopoulos et al., 2020) | `relu` (L=D) (Choromanski et al., 2021) | `tri` (L=2D) (Peng et al., 2021) | `dpfp` (L=6D) (Schlag et al., 2021).

3 Capacity via Associative Recall

- A Single layer transformer must learn to recall the correct key to a given query out of a random sequence of keys.
- Equivalently: minimize Loss $\mathcal{L} = \|A(\boldsymbol{q}, \boldsymbol{k}) - \mathrm{Id}_N\|_F^2$ with *Attention matrix*

$$A_{m,n}(\boldsymbol{q}, \boldsymbol{k}) = \Phi(q_m, k_n) / \textstyle\sum_{l=1}^N \Phi(q_m, k_l), \quad m, n = 1, \dots, N$$

- *Capacity:* $\mathrm{Cap}(\Phi) = \sup\{N : \inf_{\boldsymbol{q},\boldsymbol{k}} \mathcal{L} = 0\}$ - the maximal solvable sequence length.
- $\Phi(q,k) = F(q-k)$, $F \in \mathcal{C}_0$, $F(0) > 0 \Rightarrow \mathrm{Cap}(\Phi) = \infty$. Ex.: Gauss, Laplace, (softmax).
- Finite Feature Maps: $\mathrm{Cap}(\Phi) \le \dim \mathcal{H}_\Phi \le L$, with RKHS $\mathcal{H}_\Phi$.

Finite Feature Maps trade expressivity for speed: fewer features L are faster, but $\mathrm{Cap}(\Phi) \le L$ — they degenerate exactly when they get fast.

Can we have infinite capacity with quasi linear complexity?

4 Univariate Kernels via Sorting

- Keys sorted $t_1 \le \dots \le t_N$, query s with $t_p \le s < t_{p+1}$:

$$\sum_{n=1}^N e^{-|s-t_n|} v_n = e^{-s} \underbrace{\sum_{n \le p} e^{t_n} v_n}_{\text{prefix sum}} + e^{s} \underbrace{\sum_{n > p} e^{-t_n} v_n}_{\text{suffix sum}}$$

- Sort $\mathcal{O}(N \log N)$ + prefix sums $\mathcal{O}(NC) \Rightarrow$ *quasi linear*.
- Same trick: $|s-t|$, splines (bump $\max\{0, 1-|s-t|\}$), Matérn, Riesz.
- Laplace, bump: Cap = ∞. Riesz: Cap = 2.

5 Additive Kernels in $\mathbb{R}^D$

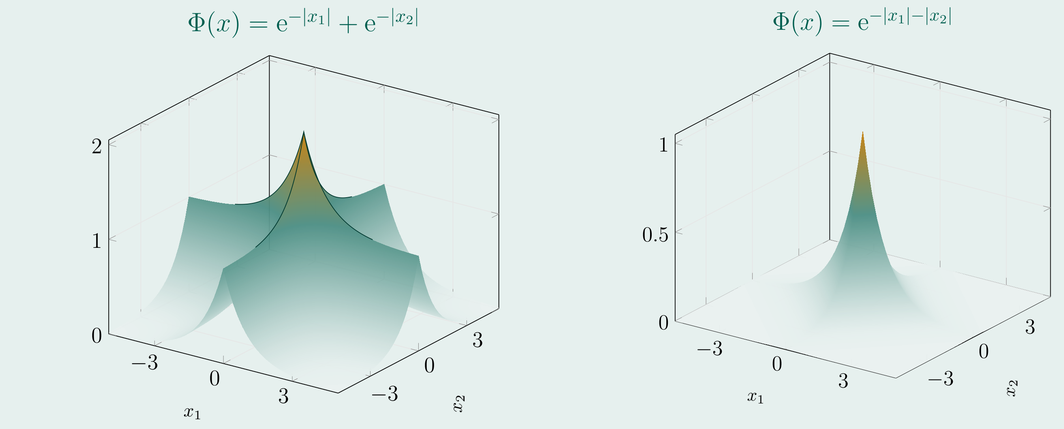
- The *additive kernel* built from $\phi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ has the form

$$\Phi(q,k) = \phi(q_1, k_1) + \dots + \phi(q_D, k_D)$$

- D univariate sums, $\mathcal{O}\big(HDN(\log(N)+D)\big)$.
- $\mathrm{Cap}(\Phi) \ge \mathrm{Cap}(\phi)$; spd, stationary, quasi linear carry over.
- ⇒ `add_laplace`, `add_bump`: Cap = ∞ *and* quasi linear.
- FFMs such as `elu`, `ReLU` an d `tri` are additive too, with $\phi(s,t) = f(s)^\top f(t)$, but $\dim \mathcal{H}_\phi \in \{1, 2\}$. Ours: $\dim \mathcal{H}_\phi = \infty$.

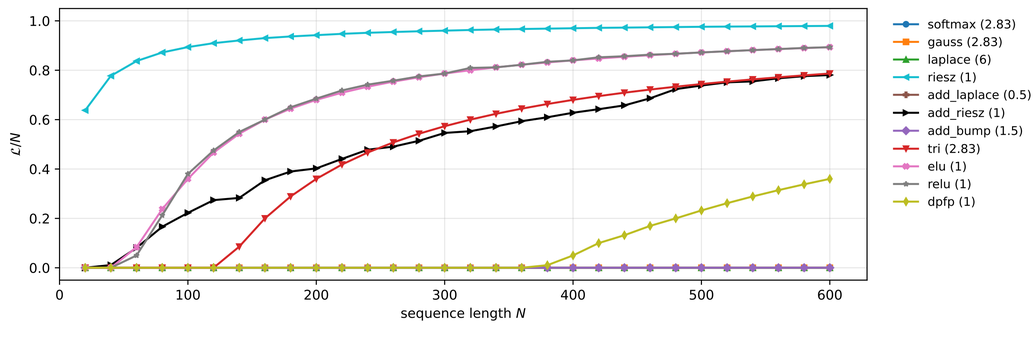
6 Additive vs. Product Laplace

- Since additive kernels do not mix components, they are feasible to compute in higher dimensions linear in D.
- The product Laplace kernel $e^{-\|q-k\|_1}$ has Cap = ∞ and is quasi linear, but splits into 2^D ECDFs and therefore scales exponential in D.
- Additive Laplace has non vanishing axis artifacts, while product Laplace is $\mathcal{C}_0$.



Left: Additive Laplace and Right: Product Laplace kernel. Both in $D = 2$.

7 Results: Capacity & Text Embedding



Associative recall, $D = 64$, best of 3 seeds. FFMs fail once $N > L$; `gauss`, `laplace`, `add_laplace`, `add_bump` reach $\mathcal{L} \approx 0$ up to $N = 600$.

Kernel	$\phi(s,t)$ or $\Phi(q,k)$	Cap	Cost	τ	MTEB ≤ 512	LoCo nDCG@10 2048	4096	8192
`softmax`	$e^{q^\top k}$	∞	quadratic	2.828	64.41	0.873	0.882	0.883
`gauss`	$e^{-\frac{1}{2}\|q-k\|_2^2}$	∞	quadratic	2.828	64.41	0.875	0.884	0.886
`laplace`	$e^{-\|q-k\|_1}$	∞	quasi linear	6.0	64.38	0.885	0.891	0.887
`riesz`	$\|q\|_2 + \|k\|_2 - \|q-k\|_2 + \varepsilon$	-	quadratic	1.0	61.69	0.805	0.839	0.858
`add_riesz`	$\|s\| + \|t\| - \|s-t\| + \varepsilon$	≥ 2D	quasi linear	1.0	61.72	0.769	0.813	0.832
`add_laplace`	$e^{-\|s-t\|}$	∞	quasi linear	0.5	62.12	0.830	0.845	0.858
`add_bump`	$\max\{0, 1-\|s-t\|\}$	∞	quasi linear	1.5	62.12	0.829	0.843	0.859
`tri`	$\cos(s-t)$	≤ 2D	linear	2.828	61.53	0.829	0.838	0.851
`relu`	ReLU(s)ReLU(t)	≤ D	linear	1.0	62.29	0.472	0.516	0.556
`elu`	(elu+1)(s)(elu+1)(t)	≤ D	linear	1.0	61.55	0.699	0.773	0.794
`dpfp`	Schlag et al. (2021)	≤ 6D	linear	1.0	63.42	0.681	0.811	0.823

nomic-embed-text-v1 (Nussbaum et al., 2025) distilled from the softmax teacher (trained on ≤ 512 tokens). Underline: best feasible sub-quadratic kernel.

- Stationary kernels tend to extrapolate better under RoPE.
- `add_laplace` and `add_bump` are stationary.
- All continuous stationary FFMs are of the form:

$$\Phi(q,k) = \sum_{l=1}^L \alpha_l \cos(\omega_l^\top (q-k)),$$

where $\alpha_l \ge 0$ and $\omega_l \in \mathbb{R}^D$

- Only stationary FFM: `tri`.

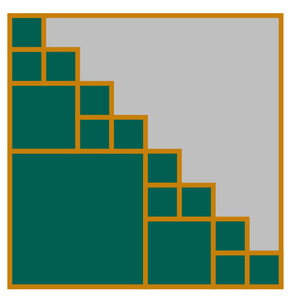
8 Speed & Implementation

	`add_laplace`	`add_riesz`			`softmax`	
N	`GLOBAL`	`GLOBAL`	`FUSED`	`KEOPS`	`MEM_EFF32`	`FLASH16`
128	0.442	0.455	0.305	0.381	0.029	**0.021**
256	0.575	0.539	0.353	0.425	0.058	**0.022**
512	0.824	0.694	0.478	0.688	0.144	**0.041**
1024	1.305	1.014	0.719	1.648	0.431	**0.093**
2048	2.260	1.690	1.278	4.754	1.645	**0.266**
4096	4.472	3.583	2.991	17.14	6.219	**1.016**
8192	9.109	7.336	–	63.94	24.12	**3.780**
16384	20.31	16.86	–	249.1	94.95	**14.99**
32768	42.07	**34.88**	–	985.6	379.4	59.79
65536	86.94	**73.76**	–	3877	1523	239.2
131072	**190.1**	200.0	–	15608	6118	968.8

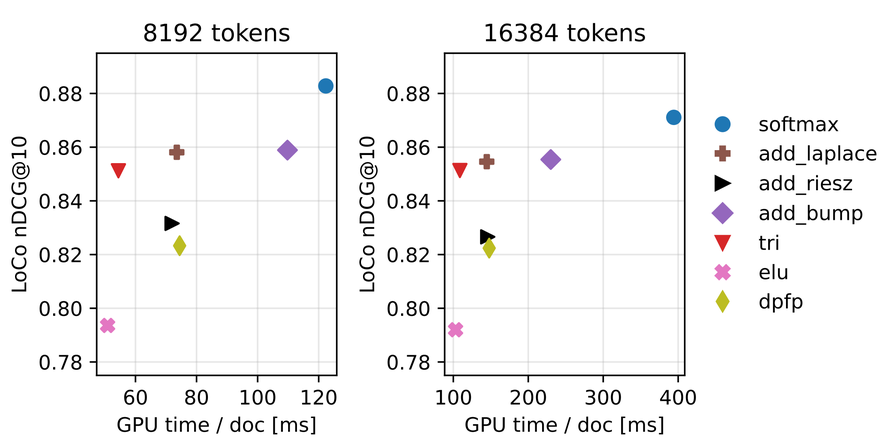
Forward pass [ms], RTX 5090 (Blackwell), B=4, H=12, D=C=64. Ours in `fp32`. Bold: fastest overall, underline: fastest `fp32`.

	Complexity	Implemented
Forward	$\mathcal{O}(N \log N)$	yes
Backward	$\mathcal{O}(N \log N)$	yes (splines)
Padding	$+\mathcal{O}(ND)$	yes (splines)
Causal	$\mathcal{O}(N \log^2 N)$	not yet

- Parallel over Batch B, Heads H, and Dimension D.
- Prefix sums computed on the fly (avoid $\mathcal{O}(PDNC)$ prefix tensors). Memory overhead of sorted copies only $\mathcal{O}(PDN)$.
- Batches with different length can be padded to max length.
- Causal masking is possible with small overhead $\mathcal{O}(N \log^2(N))$.



Causal mask, split recursively: Apply unmasked procedure to each blocks separately yields ⇒ $\mathcal{O}(N \log^2 N)$.



LoCo quality vs. GPU time per document at 8192 / 16384 tokens.

Code: github.com/Nicolaj-Rux/quasi-linear-kernel-attention Paper: https://arxiv.org/pdf/2609.35349 Contact: nicolaj.rux@mathematik.tu-chemnitz.de
Funding: ESF & Free State of Saxony (NR); DFG 530824055, 572825596 (JH).