Qualcomm AI research

QuoKA: Query-oriented KV selection for efficient LLM Prefill

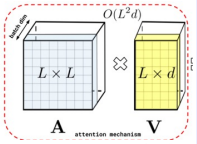
Dalton Jones†, M. Harper Langston, Junyoung Park, Matthew Morse, Mingu Lee, Chris Lott
Qualcomm AI Research*

†daltjone@qti.qualcomm.com

ICLR

Motivation

- **Transformers** have become the backbone of modern models in **vision, language, and beyond**. But **self-attention mechanism** suffers from:
  - Quadratic compute and memory complexity
  - Poor scalability for on-device or embedded applications
- These limitations hinder deployment in real-time, mobile, and resource-constrained environments
- While many efficient alternatives exist, they often compromise generality or require redesigning the architecture

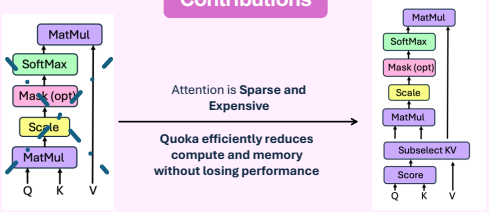


$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

We present QuoKA (Query-oriented KV selection for efficient LLM Prefill), a training-free generalized method to reduce the computation and memory complexity of attention without sacrificing accuracy with the following benefits:

Full Attention	Prior Sparse Methods	Quoka
Quadratic compute & memory	Often training-dependent	**Subquadratic**
Accurate but slow	Architecture changes	**Training-free drop-in**
Hardware-unfriendly ops (e.g., softmax)	Complex kernels	**Pure linear algebra**

Contributions



- Introduce QuoKA, a training-free, hardware-agnostic sparse attention method tailored for chunked prefill, built entirely on standard linear algebra operations.
- Identify a key empirical insight: queries with low cosine similarity to the mean query interact with the most keys, enabling an efficient strategy for query selection.
- Propose a two-stage KV selection strategy—query subselection and cosine-similarity–based KV scoring—plus group-aware aggregation to maintain compatibility.
- Achieve significant latency improvements, across long-context benchmarks.
- Demonstrates broad generalization and robustness, showing stable performance across sparsity levels and diverse LLM families (Llama3, Qwen3, SmolLM, GPT-OSS) and architectural variants (RoPE/NoPE, MoE).
- Results include ***3× lower TTFT, 5× GPU attention speedup, and up to 7× CPU speedup, while using 88% fewer KV pairs and maintaining near-baseline accuracy***

Method

QuoKA: training-free, hardware-friendly sparse attention that reduces prefill latency by efficiently identifying high-impact KV subsets.

- Query subselection via **cosine dissimilarity**: Quoka identifies representative queries, computing cosine similarity between each query and mean query; only queries with lowest similarity (i.e., most informative, highest diversity) are retained for KV selection.
- **Cosine-normalized scoring for stable KV relevance** estimation: Both queries and keys are L2-normalized, enabling efficient computation of bounded query–key relevance score using cosine similarity, proving stable ranking across layers and heads.

Algorithm 1 KV cache sub-selection using QUOKA

Require: queries Q, keys K, values V, prefill chunk size B_{CP}, selective attention budget B_{SA}, max queries N_Q, number of KV heads n_{KV}, number of attention heads n_Q

▷ Query Sub-selection

```
if B_CP > N_Q then
    M_Q ← mean(Q, dim=2)
    S_Q ← CosSim(Q, M_Q)
    Q ← gather(topk(−S_Q, N_Q), Q)
end if
```

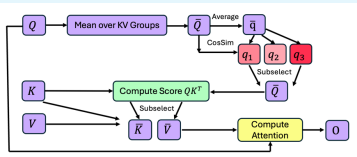
▷ Efficient Cosine Similarity via Pre-aggregation

```
Q ← Q/norm(Q, dim=−1)                                   ▷ (b, n_q, N_Q, d)
K ← K/norm(K, dim=−1)                                   ▷ (b, n_KV, T, d)
Q̄ ← mean(Q.reshape(b, n_KV, n_Q/n_KV, N_Q, d), dim=2)   ▷ (b, n_KV, N_Q, d)
S ← Q̄K^T                                                ▷ (b, n_KV, N_Q, T)
S ← max(S, dim=2)                                       ▷ (b, n_KV, T)
I ← topk(S, B_SA)
K*, V* ← gather(K, I), gather(V, I)
```

Key Intuition: Queries that differ most from the average query capture rare or informative token interactions and therefore touch the largest set of relevant keys.

- **Group-aware query aggregation** (GQA-compatibility): For architectures with GQA, representative queries regrouped and averaged per KV-head group, ensuring key scoring aligns with how queries are shared across heads.
- **Two-stage KV selection pipeline**: ***Stage 1***: compute and subselect *representative queries*; average these among key value groups. ***Stage 2***: compute *cosine similarity scorings* with tokens from Stage 1 and normalized keys; use resulting score and *topk* to subselect highest scoring KV pairs to form reduced KV cache.
- **Training-free sparse attention using chunked prefill**: QuoKA performs KV selection independently for each prefill chunk, relying on standard linear algebra ops and no model modifications, enabling efficient and portable sparsity.
- **Unlike learned sparse attention or redesigns, QuoKA seamlessly drops into existing attention modules.**

Implementation



- No softmax / complex kernels
- Pure matmul + norm + reduction / standard linear algebra operations
- Hardware agnostic / Training-free drop-in

Complexity

Runtime Complexity	
QUOKA	$\mathcal{O}(B_{\text{CP}} + (N_Q(1 + dn_{\text{KV}}))T)$
SampleAttention	$\mathcal{O}((dn_Q + n_Q/n_{\text{KV}} + n_{\text{KV}})N_Q T)$
SparQ	$\mathcal{O}(B_{\text{CP}} T d n_Q)$
Loki	$\mathcal{O}(d n_Q (B_{\text{CP}} T + d(B_{\text{CP}} + T)))$
LessIsMore	$\mathcal{O}(dn_Q B_{\text{CP}} T / L)$

Memory Complexity	
QUOKA	$\mathcal{O}(n_{\text{KV}} N_Q T)$
SampleAttention	$\mathcal{O}(n_Q N_Q T)$
SparQ	$\mathcal{O}(n_Q B_{\text{CP}} T)$
Loki	$\mathcal{O}(n_Q B_{\text{CP}} T)$
LessIsMore	$\mathcal{O}(n_Q B_{\text{CP}} T / L)$

Memory and computational complexity significantly lower than full attention and improves on SoTA sparse methods

Results

- We implement **QuoKA** in the attention module of models with a variety of architectures, compare this with competing sparse attention methods and test using several long context benchmarks. **

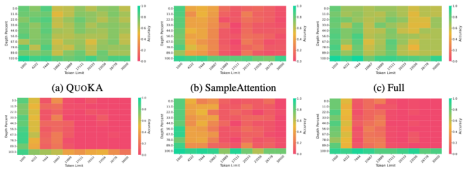


Figure 7: Accuracy across document length and needle depth for NIAH using additional sparse attention methods with $B_{\text{SA}} = 2048$ and $B_{\text{CP}} = 128$.

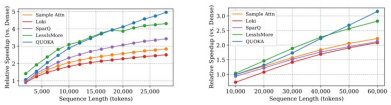
Table 1: **RULER evaluation results across increasing lengths with $B_{\text{SA}} = 1,024$.** Highest per column in **bold** (Higher is better). Attention sparsification applied on full attention layers.

	Llama-3.2-3B-Instruct				*Qwen-2.5-3B-Instruct*				*Qwen-3-4B*				*Smollm3*				*GPT-OSS-20B*			
Length	4k	8k	16k	32k	4k	8k	16k	32k	4k	8k	16k	32k	4k	8k	16k	32k	4k	8k	16k	32k
SnapKV	29.15	13.70	12.44	6.21	19.25	12.40	9.88	8.13	27.29	17.15	10.94	10.07	43.72	35.98	24.21	12.23	58.18	31.45	24.82	16.63
KeyDiff	53.34	31.10	24.29	14.87	34.09	20.68	15.79	10.32	37.29	27.15	20.94	12.07	62.85	51.60	41.64	30.37	69.58	28.76	15.86	9.65
LessIsMore	75.15	49.23	30.44	19.16	36.66	20.21	12.84	10.12	65.55	42.75	24.39	14.87	79.67	50.17	35.12	24.21	67.35	54.49	38.27	20.11
Loki	74.84	56.50	25.76	8.05	74.40	60.09	48.96	34.12	82.83	65.19	52.29	39.31	84.52	64.20	50.10	22.66	75.48	65.36	54.67	39.92
SparQ	79.36	60.80	48.59	31.14	78.07	59.87	54.71	36.74	87.93	68.97	56.02	35.20	82.45	57.62	32.18	18.69	70.07	54.00	30.75	15.20
SampleAttn	78.25	61.14	48.31	31.73	77.17	60.88	56.64	36.17	87.84	72.46	59.57	40.72	85.72	66.44	59.10	45.98	76.20	70.35	53.91	30.42
QUOKA	**86.71**	**80.19**	**70.90**	**57.01**	**87.85**	**74.27**	**66.82**	**59.37**	**93.73**	**91.07**	**88.57**	**74.83**	**89.97**	**79.94**	**72.69**	**61.37**	**78.92**	**79.19**	**73.40**	**57.79**

Table 3: **LongBench results (Higher is better).** For each model, the three columns correspond to selective budgets $B_{\text{SA}} \in \{512, 1024, 2048\}$.

	Llama3.2-3B-Instruct			*Qwen2.5-3B-Instruct*			*Qwen3-4B*			*Smollm3*		
Budget	512	1024	2048	512	1024	2048	512	1024	2048	512	1024	2048
LessIsMore	0.703	0.788	0.850	0.461	0.556	0.659	0.665	0.773	0.868	0.765	0.842	0.918
SparQ	0.721	0.802	0.842	0.636	0.726	0.808	0.686	0.782	0.872	0.725	0.805	0.922
Loki	0.686	0.757	0.842	0.589	0.671	0.787	0.622	0.782	0.872	0.384	0.801	0.622
SampleAttn	0.738	0.800	0.901	0.660	0.756	0.828	0.755	0.875	0.947	0.856	0.929	0.966
QUOKA	**0.945**	**0.972**	**0.986**	**0.869**	**0.945**	**0.977**	**0.966**	**0.992**	**0.995**	**0.998**	**1.03**	**1.028**

- **Latency Experiments: Up to 3× end to end speedup**



(a) Attention latency on NVIDIA A100 GPU

(b) TTFT on NVIDIA A100 GPU

Conclusion & Next Steps

- QuoKA shows that **query-oriented KV selection** enables efficient, training-free sparse attention for long-context prefill.
- Achieves large latency reductions while preserving accuracy across architectures and model families (Cosine similarity ensures bounded, scale-invariance).
- Results include ***3× lower TTFT, 5× GPU attention speedup, and up to 7× CPU speedup, while using 88% fewer KV pairs and maintaining near-baseline accuracy***
- Future work includes adaptive query selection, extensions beyond prefill, and integration with complementary efficiency techniques.

*Qualcomm AI Research is a division of Qualcomm Technologies, Inc

* * All latency numbers include end-to-end prefill and exclude kernel warm-up
Batch size, sequence length, and sparsity held constant across methods..