

Compressed Attention

Lester Mackey

Microsoft Research New England

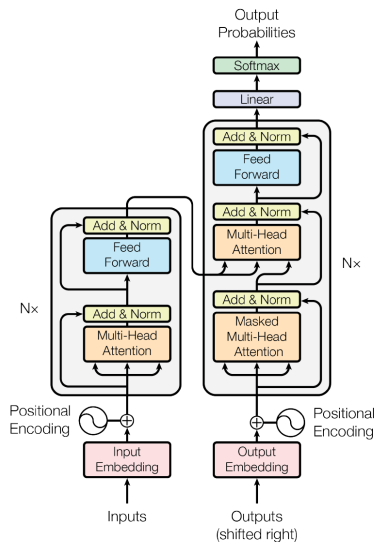
October 13, 2026

Joint work with Annabelle M. Carrell, Albert Gong, Tobias Schröder,
Raaz Dwivedi, and Abhishek Shetty

Motivation: Transformers

[Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin, 2017]

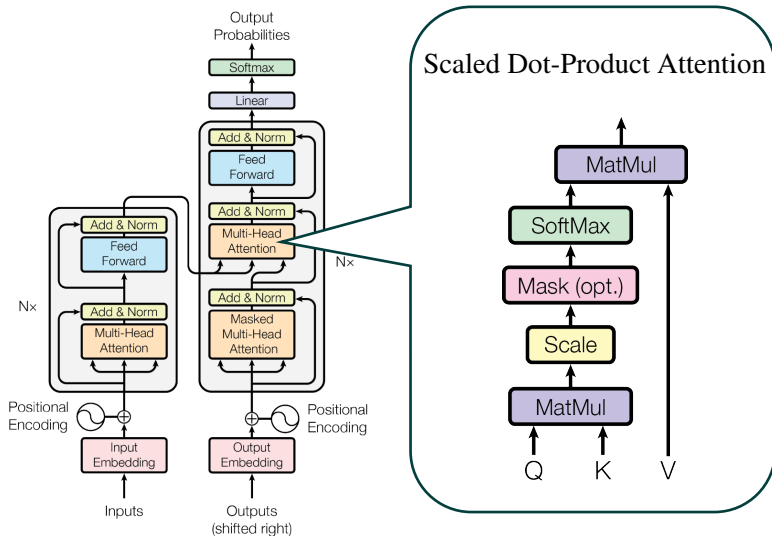
Transformers revolutionized the way we process text, images, and speech



Motivation: Transformers

[Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin, 2017]

Transformers rely on **attention** to effectively capture dependencies in a sequence



Unpacking Attention

Given: n query, key, and value vectors $(\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i)$ in $\mathbb{R}^d$

$$\text{Attention} = \text{softmax} \underbrace{\begin{pmatrix} \begin{bmatrix} - & \mathbf{q}_1 & - \\ - & \mathbf{q}_2 & - \\ & \vdots & \\ - & \mathbf{q}_n & - \end{bmatrix} \begin{bmatrix} | & | & & | \\ \mathbf{k}_1^\top & \mathbf{k}_2^\top & \dots & \mathbf{k}_n^\top \\ | & | & & | \end{bmatrix} & \begin{bmatrix} - & \mathbf{v}_1 & - \\ - & \mathbf{v}_2 & - \\ & \vdots & \\ - & \mathbf{v}_n & - \end{bmatrix} \end{pmatrix}}_{n \times n}$$

Benefit: Unprecedented success in capturing long-range dependencies

Drawback: Attention is expensive!

① **Time cost:** Grows quadratically in sequence length!

- Long user inputs take too long to process (and consume too much energy)
- Long model generations take too long to produce (and consume too much energy)

Unpacking Attention

Given: n query, key, and value vectors $(\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i)$ in $\mathbb{R}^d$

$$\text{Attention} = \text{softmax} \underbrace{\begin{pmatrix} \begin{bmatrix} - & \mathbf{q}_1 & - \\ - & \mathbf{q}_2 & - \\ & \vdots & \\ - & \mathbf{q}_n & - \end{bmatrix} & \begin{bmatrix} | & | & & | \\ \mathbf{k}_1^\top & \mathbf{k}_2^\top & \dots & \mathbf{k}_n^\top \\ | & | & & | \end{bmatrix} \end{pmatrix}}_{n \times n} \begin{bmatrix} - & \mathbf{v}_1 & - \\ - & \mathbf{v}_2 & - \\ & \vdots & \\ - & \mathbf{v}_n & - \end{bmatrix}$$

Benefit: Unprecedented success in capturing long-range dependencies

Drawback: Attention is expensive!

② **Memory cost:** Grows linearly with sequence length!

- Limits both the maximum context length and the longest feasible generation

Unpacking Attention

Given: n query, key, and value vectors $(\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i)$ in $\mathbb{R}^d$

$$\text{Attention} = \text{softmax} \underbrace{\left(\begin{bmatrix} \text{---} \mathbf{q}_1 \text{---} \\ \text{---} \mathbf{q}_2 \text{---} \\ \vdots \\ \text{---} \mathbf{q}_n \text{---} \end{bmatrix} \begin{bmatrix} \begin{array}{c} | \\ \mathbf{k}_1^\top \\ | \end{array} & \begin{array}{c} | \\ \mathbf{k}_2^\top \\ | \end{array} & \cdots & \begin{array}{c} | \\ \mathbf{k}_n^\top \\ | \end{array} \end{bmatrix} \right)}_{n \times n} \begin{bmatrix} \text{---} \mathbf{v}_1 \text{---} \\ \text{---} \mathbf{v}_2 \text{---} \\ \vdots \\ \text{---} \mathbf{v}_n \text{---} \end{bmatrix}$$

Benefit: Unprecedented success in capturing long-range dependencies

Drawback: Attention is expensive!

- 1 **Time cost:** Grows quadratically in sequence length!
- 2 **Memory cost:** Grows linearly with sequence length!

Question: Can we approximate attention both cheaply and accurately?

Attention Is Just Averaging

Given: n query, key, and value vectors $(\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i)$ in $\mathbb{R}^d$

$$\mathbf{Attention}_{ij} = \frac{\frac{1}{n} \sum_{l=1}^n \exp(\mathbf{q}_i \cdot \mathbf{k}_l) \mathbf{v}_{lj}}{\frac{1}{n} \sum_{l=1}^n \exp(\mathbf{q}_i \cdot \mathbf{k}_l)} \approx \frac{\frac{1}{|S|} \sum_{l \in S} \exp(\mathbf{q}_i \cdot \mathbf{k}_l) \mathbf{v}_{lj}}{\frac{1}{|S|} \sum_{l \in S} \exp(\mathbf{q}_i \cdot \mathbf{k}_l)}$$

Idea: Approximate averages using a subset S of $s \ll n$ key-value pairs

Benefits: Only $O(ns)$ work instead of $\Theta(n^2)$ and only s cache entries instead of n

Uniform subset: Attention error $\propto \frac{1}{\sqrt{s}}$; need $s = 10000$ for 1% error

Question: Can we do better?

Distribution Compression

Goal: Accurately summarize a distribution using a small number of points

Given: Input points $\mathcal{S}_{\text{in}} = \{x_1, \dots, x_n\}$ with empirical distribution $\mathbb{P}_n = \frac{1}{n} \sum_{i=1}^n \delta_{x_i}$

- Pre-generated by any algorithm (e.g., $x_i = (\mathbf{k}_i, \mathbf{v}_i)$ in a transformer)

Output: Coreset $\mathcal{S} \subset \mathcal{S}_{\text{in}}$ with $|\mathcal{S}| = s$, $\mathbb{Q} = \frac{1}{s} \sum_{x \in \mathcal{S}} \delta_x$, and coreset averages $\mathbb{Q}f = \frac{1}{s} \sum_{x \in \mathcal{S}} f(x)$ closely approximating exact averages $\mathbb{P}_n f = \frac{1}{n} \sum_{i=1}^n f(x_i)$

Example:

$$\frac{1}{s} \sum_{(\mathbf{k}, \mathbf{v}) \in \mathcal{S}} \exp(\mathbf{q} \cdot \mathbf{k}) \mathbf{v} \approx \frac{1}{n} \sum_{l=1}^n \exp(\mathbf{q} \cdot \mathbf{k}_l) \mathbf{v}_l$$

Distribution Compression

Goal: Accurately summarize a distribution $\mathbb{P}_n$ using a coreset $\mathbb{Q}$ of s points

Standard solutions

- Uniform subsampling / i.i.d. sampling

Drawback: Samples are too large!

- Typical integration error $\mathbb{P}_n f - \mathbb{Q} f = \Theta(s^{-1/2})$: need $s = 10000$ for 1% error
- **Prohibitive** for expensive downstream tasks and function evaluations

Question: Can we do better?

Minimax lower bound for worst-case integration error to $\mathbb{P}$

- $\Omega(s^{-1})$ for any compression procedure returning s points [Phillips and Tai, 2020]

This talk: Practical compression strategies that match lower bound up to log factors

Goal: Return **coreset** with **better-than-i.i.d.** integration error between $\mathbb{P}_n$ and $\mathbb{Q}$

Kernel sub-Gaussian constant

$$\sigma_{\mathbf{k}} = \sup_f \frac{\sqrt{2 \log \mathbb{E}[\exp(\mathbb{P}_n f - \mathbb{Q} f) \mid \mathbb{P}_n]}}{\|f\|_{\mathbf{k}}}$$

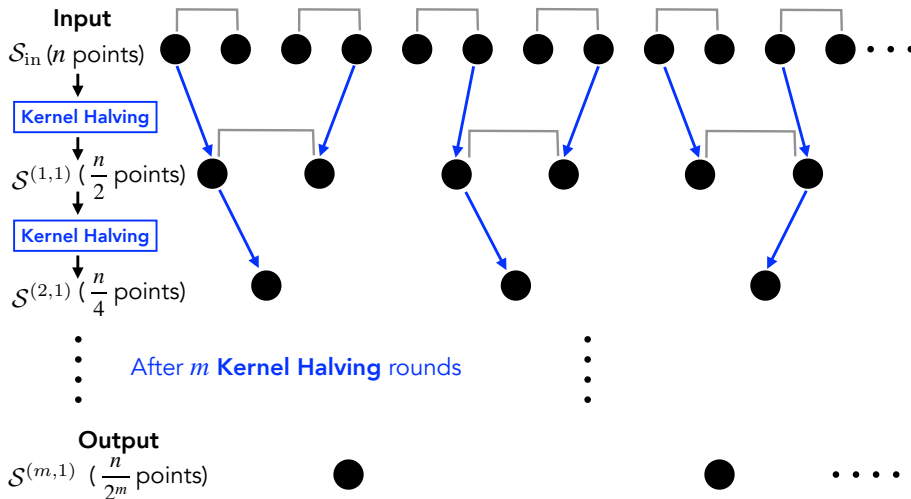
$\Rightarrow$ Each $\mathbb{Q}f$ is concentrated around $\mathbb{P}_n f$ with variance $\leq \sigma_{\mathbf{k}}^2 \|f\|_{\mathbf{k}}^2$

- Parameterized by a **reproducing kernel $\mathbf{k}$** : any **symmetric** ($\mathbf{k}(x, y) = \mathbf{k}(y, x)$) and **positive semidefinite** ($\sum_{i,l} c_i c_l \mathbf{k}(z_i, z_l) \geq 0, \forall z_i \in \mathbb{R}^d, c_i \in \mathbb{R}$) function

- Gaussian: $\mathbf{k}(x, y) = e^{-\frac{1}{2}\|x-y\|_2^2}$, Exponential: $\mathbf{k}(x, y) = \exp(x \cdot y)$

Kernel Thinning [Dwivedi and Mackey, 2022]

Selects representative coreset of size s by **repeatedly halving** input $\mathcal{S}_{\text{in}}$



Kernel Halving [Dwivedi and Mackey, 2024]

Goal: Split $\mathcal{S}_{\text{in}}$ into two **balanced** coresets $\mathcal{S}, \mathcal{S}'$ of equal size

- **Balance:** $\mathbb{Q}\mathbf{k} \approx \mathbb{Q}'\mathbf{k} \Leftrightarrow \mathbb{P}_n\mathbf{k} \approx \mathbb{Q}\mathbf{k} \Leftrightarrow \sigma_{\mathbf{k}} \text{ small} \quad \text{for} \quad \mathbb{Q}'\mathbf{k} \triangleq \frac{1}{|\mathcal{S}'|} \sum_{x \in \mathcal{S}'} \mathbf{k}(x, \cdot)$

Uniformly random halving: $\sigma_{\mathbf{k}} = \Omega(\frac{1}{\sqrt{n}})$

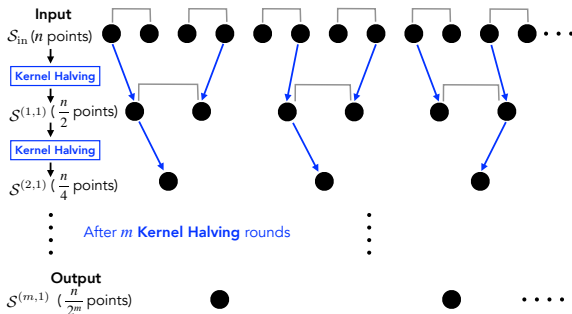
Kernel halving: Use **non-uniform** randomness to encourage balance

- Hilbert space generalization of self-balancing walk of Alweiss, Liu, and Sawhney [2020]
- Start with empty coresets $\mathcal{S}, \mathcal{S}'$
- Assign input points $(x, x') = (x_1, x_2), \dots, (x_{n-1}, x_n)$ to coresets two at a time:
 - ① Try adding x to $\mathcal{S}$ and x' to $\mathcal{S}'$ and record $\alpha_{\text{heads}} = \|\mathbb{Q}\mathbf{k} - \mathbb{Q}'\mathbf{k}\|_{\mathbf{k}}$
 - ② Try adding x' to $\mathcal{S}$ and x to $\mathcal{S}'$ and record $\alpha_{\text{tails}} = \|\mathbb{Q}\mathbf{k} - \mathbb{Q}'\mathbf{k}\|_{\mathbf{k}}$
 - ③ Final assignment: flip coin biased toward the more balanced option (the smaller α)
- Enjoys **dimension-free** $\sigma_{\mathbf{k}} \leq \frac{\sqrt{4 \log(n/\delta) \|\mathbf{k}\|_{\infty}}}{n}$ with probability $1 - \delta$

[Carrell, Gong, Shetty, Dwivedi, and Mackey, 2025]

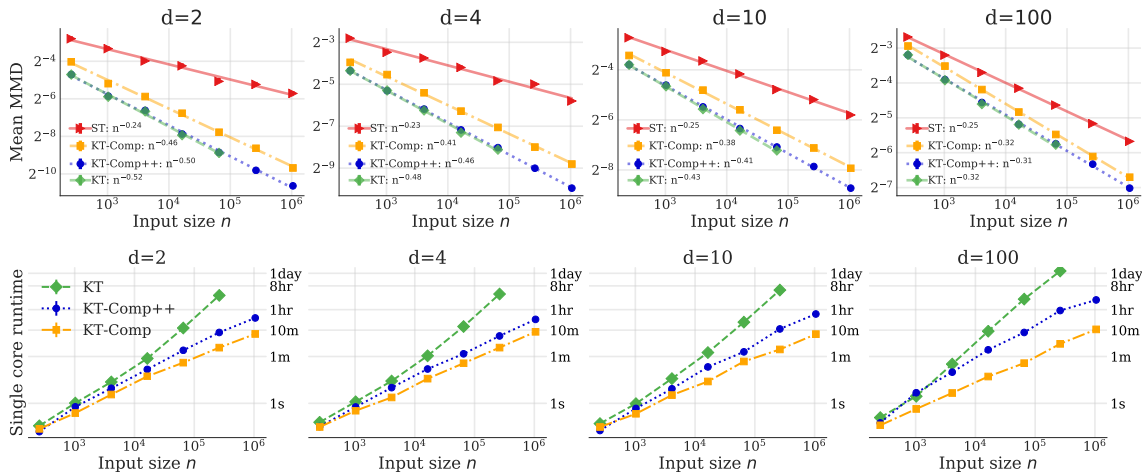
Kernel Thinning [Dwivedi and Mackey, 2022]

Selects representative coreset of size s by **repeatedly halving** input $\mathcal{S}_{\text{in}}$



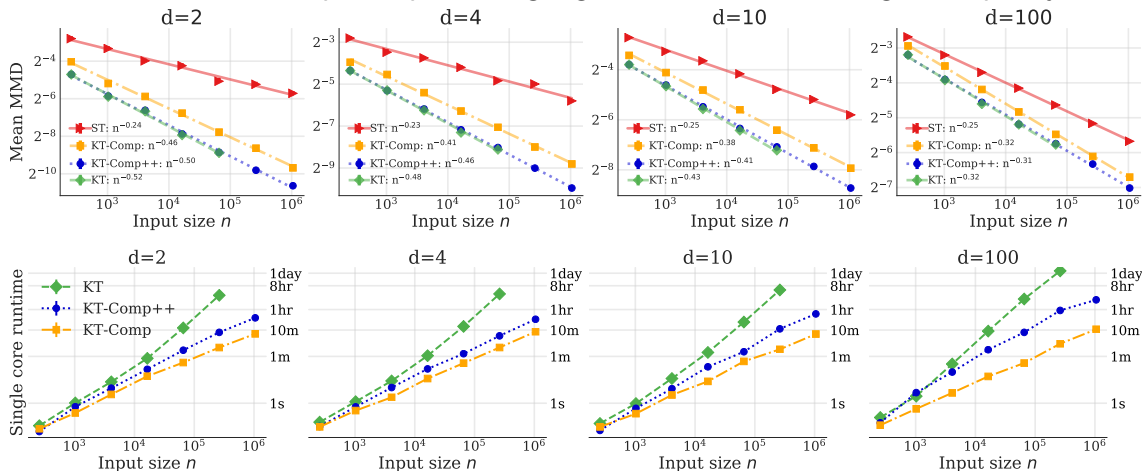
- Non-uniform randomness ensures $\sigma_{\mathbf{k}} = \mathcal{O}(\frac{\sqrt{\log s}}{s})$ after all halving rounds vs. $\sigma_{\mathbf{k}} = \mathcal{O}(s^{-1/2})$ for uniform subsample
- Runtime dominated by $\mathcal{O}(n^2)$ kernel evaluations
 - Reduces to $\tilde{\mathcal{O}}(n)$ for $s = \sqrt{n}$ using **Compress++** [Shetty, Dwivedi, and Mackey, 2022]

Question: How do we speed up thinning algorithms without ruining their quality?



Compress++ orchestrates a sequence of **cheap** halving calls on bins of size 4, 8, 16, ...

Question: How do we speed up thinning algorithms without ruining their quality?



Compress++ reduces $\Theta(n^2)$ runtime to $\tilde{O}(s^2)$, applies to any halving algorithm, and inflates error by at most a factor of 4

Thinformer = Compressed Kernel Halving

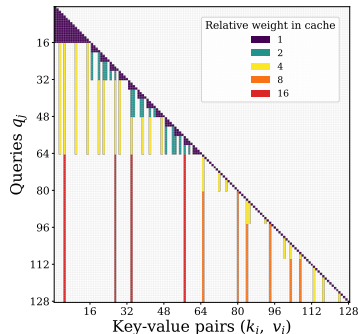
Thinformer [Carrell, Gong, Shetty, Dwivedi, and Mackey, 2025]

- **Input:** Attention kernel $\kappa_{\text{att}}((\mathbf{k}_1, \mathbf{v}_1), (\mathbf{k}_2, \mathbf{v}_2)) = \exp(\mathbf{k}_1 \cdot \mathbf{k}_2) \mathbf{v}_1 \cdot \mathbf{v}_2$
- **Output:** s tokens with subset averages nearly matching full averages in $\tilde{O}(s^2)$ time

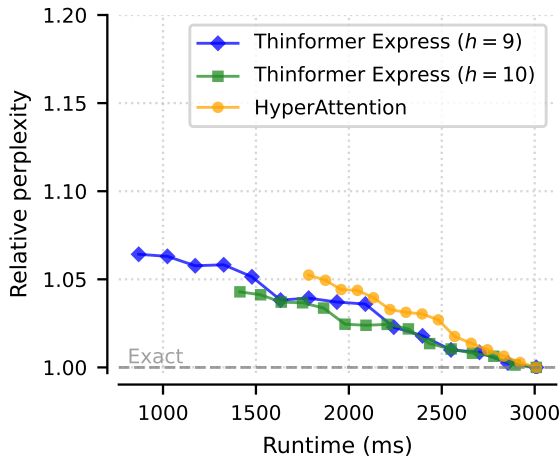
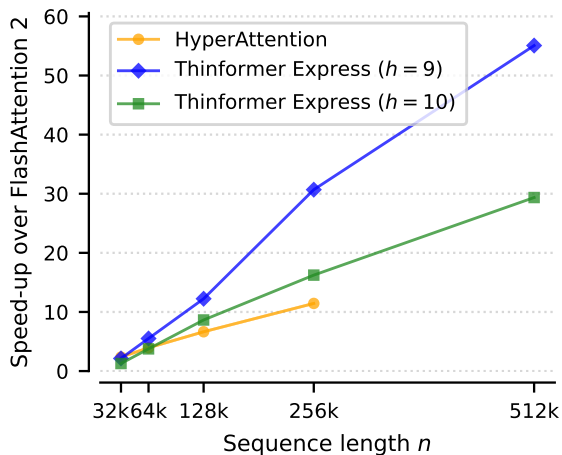
$$\Rightarrow \left| \frac{1}{n} \sum_{l=1}^n \exp(\mathbf{q}_i \cdot \mathbf{k}_l) \mathbf{v}_{lj} - \frac{1}{|S|} \sum_{l \in S} \exp(\mathbf{q}_i \cdot \mathbf{k}_l) \mathbf{v}_{lj} \right| = O\left(\frac{\log(s) \sqrt{\log \log(n)}}{s}\right)$$

Thinformer Express [Gong, Carrell, Dwivedi, and Mackey, 2027]

- Supports **causal masking** and **online generation**
- Maintains coreset of size s in $O(s^2 \log^2(n))$ time
- Guarantees $\tilde{O}(\frac{1}{s})$ masked attention error

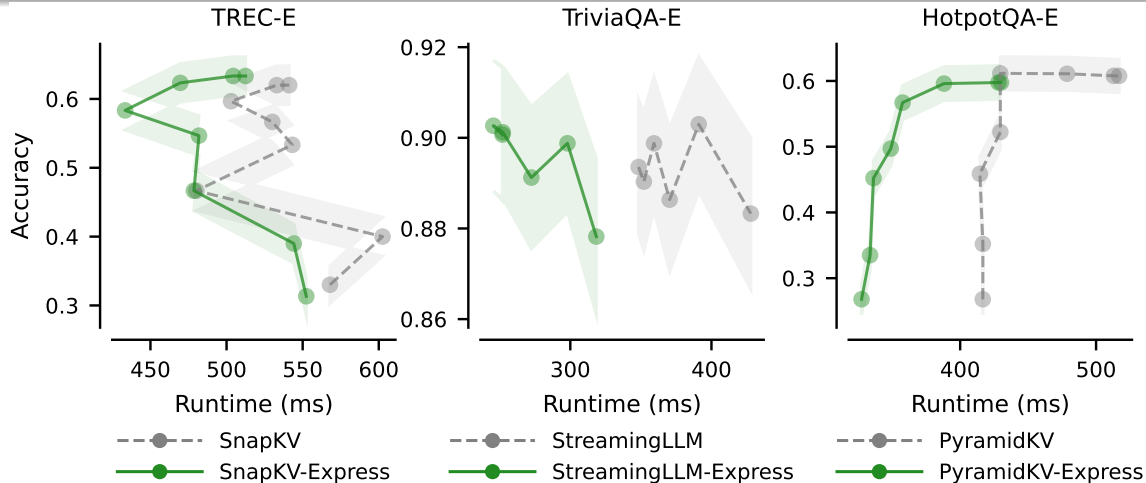


Accelerating Prefill with ChatGLM2-6B-32K ($d = 128$)



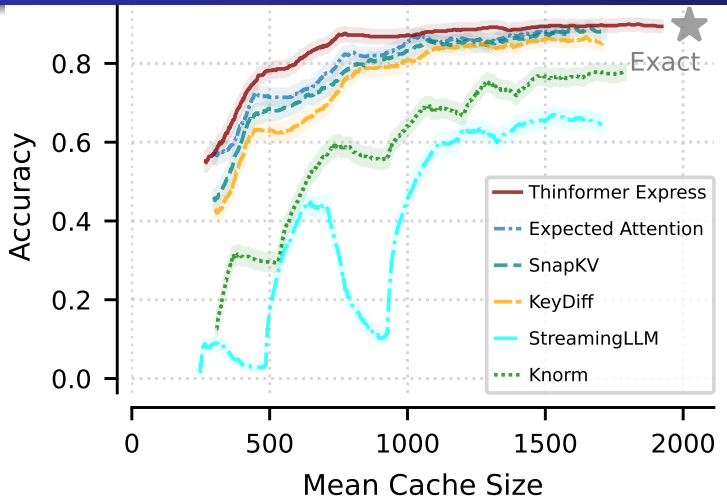
- **HyperAttention:** Heavy hitters + subsampling [Han, Jayaram, Karbasi, Mirrokni, Woodruff, and Zandieh, 2024]

Accelerated KV Cache Compression with Llama 3.1 8B Instruct



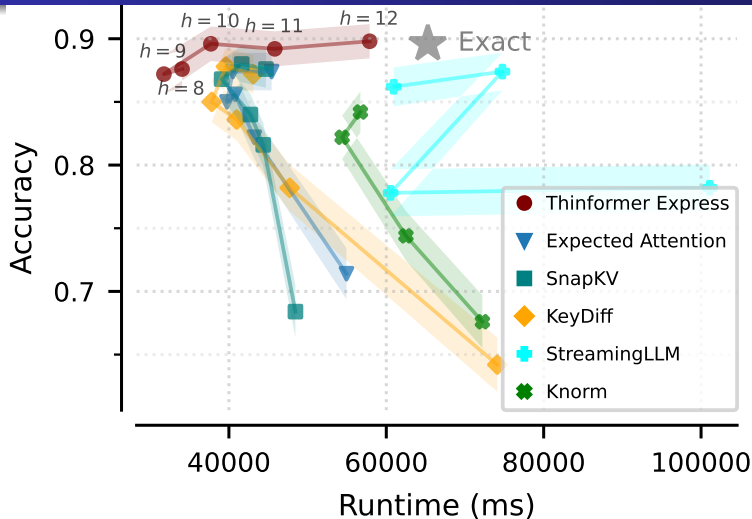
Thinformer Express can **accelerate** leading KV cache compression methods while **preserving** their long-context language understanding quality

Budgeted Reasoning with DeepSeek-R1-Distill-Llama-8B



- **MATH-500:** Competition-level math problems that benefit from step-by-step reasoning
Thinformer Express retains 99.6% of Exact accuracy with 64% of the cache size

Accelerated Reasoning with DeepSeek-R1-Distill-Llama-8B



- **MATH-500:** Competition-level math problems that benefit from step-by-step reasoning

Thinformer Express matches Exact accuracy with **58%** of the runtime

Weighted vs. Unweighted Approximations

Thinformer outputs s tokens with unweighted averages closely matching exact averages

$$\left| \frac{1}{n} \sum_{l=1}^n \exp(\mathbf{q}_i \cdot \mathbf{k}_l) \mathbf{v}_{lj} - \frac{1}{s} \sum_{l \in S} \exp(\mathbf{q}_i \cdot \mathbf{k}_l) \mathbf{v}_{lj} \right| = \tilde{O}\left(\frac{1}{s}\right)$$

Question: Can we do even better?

Lower bound: $\Omega(\frac{1}{s})$ error is **unavoidable** for unweighted attention approximations

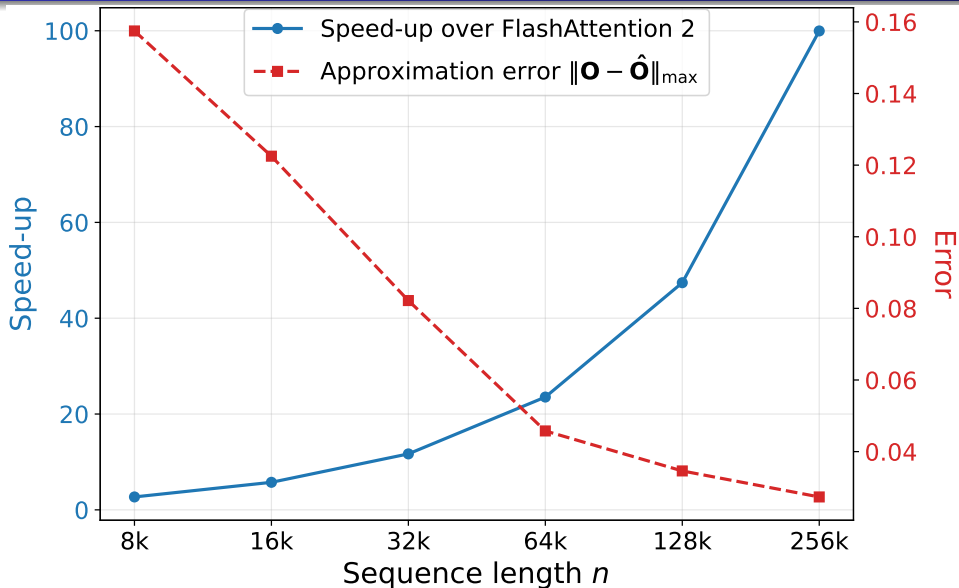
[Liberty, Andoni, and Kleiner, 2026]

Question: Can we do even better...with weights?

WildCat [Schroder and Mackey, 2026]

- Selects **optimally-weighted** keys via divide-and-conquer randomly pivoted Nyström
[Chen, Epperly, Tropp, and Webber, 2025]
- Enables **superpolynomial** $(1/n^{\sqrt{\log \log n}})$ error decay in **near-linear** time

WildCat Speed and Error Both Improve as n Grows ($s = 64$)



KV Cache Compression with Qwen2.5-7B-Instruct

75.0% Compression

Method	qasper	multifield	hotpot	2wiki	gov	multinews	trec	trivia	samsum	p.count	p.ret	lcc	repo-p	average
Exact	43.76	50.08	56.43	43.71	34.16	24.32	64.00	87.53	38.56	14.67	99.67	69.96	62.17	53.00
StreamingLLM	23.17	25.49	26.45	20.89	29.65	22.25	52.33	75.17	35.79	12.50	24.83	68.48	56.22	36.40
PyramidKV	21.59	29.96	39.38	30.24	27.12	21.13	43.33	86.77	38.27	15.88	61.06	67.40	59.20	41.64
BalanceKV	29.50	36.57	37.89	23.71	30.27	21.98	55.00	73.83	34.56	12.67	71.67	65.48	62.57	42.75
Uniform	26.91	37.51	38.46	25.93	30.02	21.86	54.33	81.71	35.37	15.33	63.44	64.84	61.34	42.85
SnapKV	25.52	30.13	44.36	31.80	29.70	22.10	49.33	88.32	37.15	16.67	89.06	69.16	56.33	45.36
WildCat	33.23	38.13	43.43	33.37	30.30	22.26	54.33	86.15	35.38	14.33	98.00	64.85	60.43	47.25

- **13 LongBench-E tasks:** question answering, summarization, code completion, passage retrieval, passage count, and few-shot learning [Bai, Lv, Zhang, Lyu, Tang, Huang, Du, Liu, Zeng, Hou, et al., 2024]
- **StreamingLLM:** Preserves latest tokens and attention sinks [Xiao, Tian, Chen, Han, and Lewis, 2024]
- **PyramidKV:** Varies cache size across layers [Li, Huang, Yang, Venkitesh, Locatelli, Ye, Cai, Lewis, and Chen, 2024b]
- **BalanceKV:** Subsamples tokens using self-balancing walks [Kochetkova, Sheth, Han, Zandieh, and Kapralov, 2026]
- **Uniform:** Subsamples tokens uniformly
- **SnapKV:** Preserves tokens similar to recent queries [Li, Huang, Yang, Venkitesh, Locatelli, Ye, Cai, Lewis, and Chen, 2024b]

Conclusions

Summary

- New tools for summarizing a probability distribution more effectively than i.i.d. sampling
- **Kernel thinning** compresses an n point summary into a $\sqrt{n}$ point summary with better-than-i.i.d. approximation error
- **Compress++** speeds up thinning algorithms without ruining their quality
- **Thinformer Express** and **WildCat** accelerate attention in transformers

Kernel Thinning and Compress++

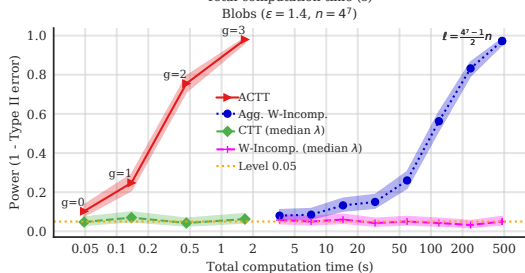
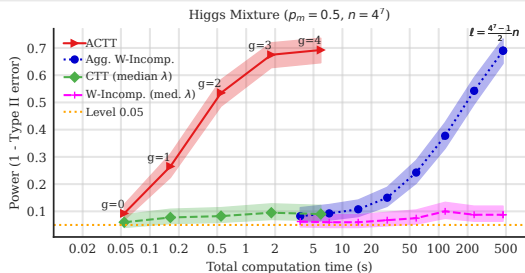
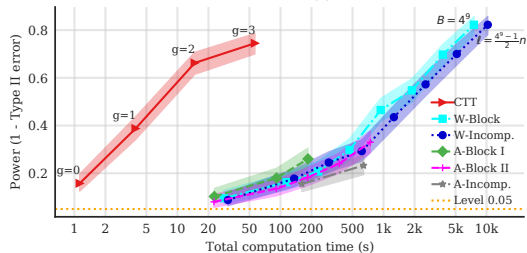
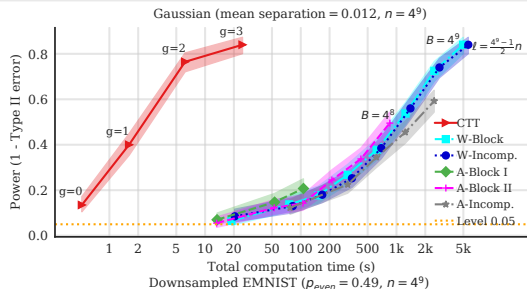
Papers: $\left\{ \begin{array}{l} \text{arxiv.org/abs/2105.05842} \\ \text{arxiv.org/abs/2110.01593} \\ \text{arxiv.org/abs/2111.07941} \end{array} \right.$

Thinformer Express and WildCat

Papers: $\left\{ \begin{array}{l} \text{arxiv.org/abs/2502.12063} \\ \text{arxiv.org/abs/2606.10944} \\ \text{arxiv.org/abs/2602.10056} \end{array} \right.$

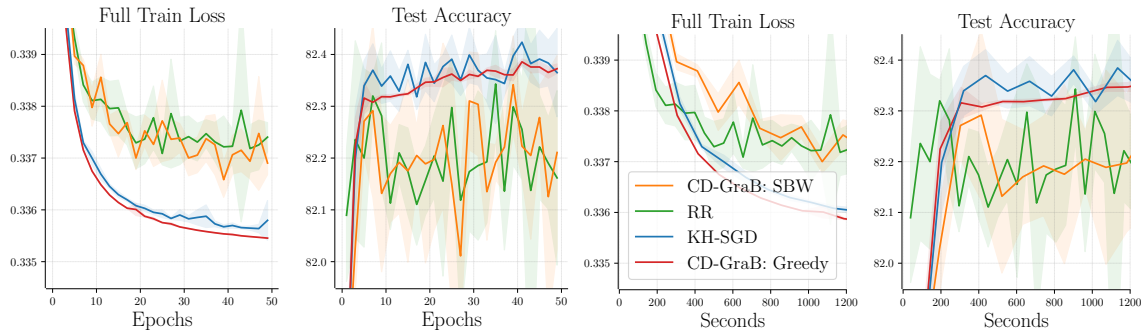
Packages: github.com/microsoft/goodpoints, [thinformer](https://github.com/microsoft/thinformer), [wildcat](https://github.com/microsoft/wildcat)

Compress Then Test [Domingo-Enrich, Dwivedi, and Mackey, 2023]



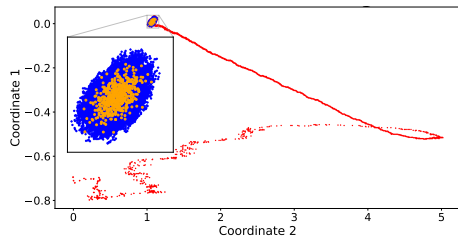
Compress Then Test (► above) yields powerful kernel tests in near-linear time

Compress Then Sort [Carrell, Gong, Shetty, Dwivedi, and Mackey, 2025]

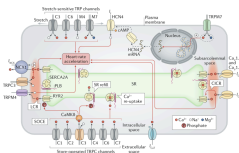


Inspired by gradient balancing (**CD-GraB**) [Cooper, Guo, Pham, Yuan, Ruan, Lu, and De Sa, 2023], **Kernel Halving SGD** accelerates model training by reordering stochastic gradients

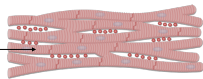
Compress Then Simulate



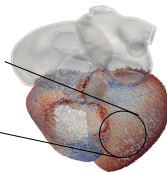
Single cell scale



Tissue scale



Organ scale



Stein kernel thinning propagates compressed uncertainty through expensive simulators while reducing biases due to off-target sampling, tempering, or burn-in

[Riabiz, Chen, Cockayne, Swietach, Niederer, Mackey, and Oates, 2021, Li, Dwivedi, and Mackey, 2024a]

Summary

- New tools for summarizing a probability distribution more effectively than i.i.d. sampling
- **Kernel thinning** compresses an n point summary into a $\sqrt{n}$ point summary with better-than-i.i.d. approximation error
- **Compress++** speeds up thinning algorithms without ruining their quality
- **Stein kernel thinning** simultaneously compresses and reduces biases due to off-target sampling, tempering, or burn-in
- **CTT**, **KH-SGD**, **Thinformer**, & **WildCat** accelerate testing, training, & transformers

Code: $\left\{ \begin{array}{ll} \text{github.com/microsoft/goodpoints} & \text{github.com/microsoft/thinformer} \\ \text{github.com/microsoft/khsgd} & \text{github.com/microsoft/wildcat} \end{array} \right.$

Accelerating T2T-ViT Image Classification

Attention Algorithm	Top-1 Accuracy (%)	Layer 1 Runtime (ms)	Layer 2 Runtime (ms)
Exact	82.55 $\pm$ 0.00	18.48 $\pm$ 0.12	1.40 $\pm$ 0.01
Performer	80.56 $\pm$ 0.30	2.54 $\pm$ 0.01	0.60 $\pm$ 0.01
Reformer	81.47 $\pm$ 0.06	7.84 $\pm$ 0.03	1.53 $\pm$ 0.01
KDEformer	82.00 $\pm$ 0.07	5.39 $\pm$ 0.03	2.28 $\pm$ 0.03
Scatterbrain	82.05 $\pm$ 0.08	6.86 $\pm$ 0.02	1.55 $\pm$ 0.03
Thinformer (Ours)	82.18 $\pm$ 0.05	2.06 $\pm$ 0.01	0.54 $\pm$ 0.00

- **Exact:** $d = 64, n_1 = 3136, n_2 = 784$, A100 GPU [Yuan, Chen, Wang, Yu, Shi, Jiang, Tay, Feng, and Yan, 2021]
- **Performer:** Positive orthogonal random features
[Choromanski, Likhoshesterov, Dohan, Song, Gane, Sarlos, Hawkins, Davis, Mohiuddin, Kaiser, Belanger, Colwell, and Weller, 2021]
- **Reformer:** Locality sensitive hashing (LSH) [Kitaev, Kaiser, and Levskaya, 2020]
- **KDEformer:** LSH + kernel density estimation [Zandieh, Han, Daliri, and Karbasi, 2023]
- **Scatterbrain:** LSH + random features [Chen, Dao, Winsor, Song, Rudra, and Ré, 2021]

Accelerating BigGAN Image Generation

Attention Algorithm	FID ($\downarrow$)	IS ($\uparrow$)	Runtime (ms)
Exact	32.18	58.37 $\pm$ 4.21	5.83 $\pm$ 0.09
Performer	33.58	38.07 $\pm$ 3.43	2.28 $\pm$ 0.00
Reformer	72.23	19.14 $\pm$ 2.09	12.10 $\pm$ 0.01
KDEformer	30.71	57.09 $\pm$ 4.05	6.40 $\pm$ 0.34
ScatterBrain	38.78	36.93 $\pm$ 2.97	3.13 $\pm$ 0.02
Thinformer (Ours)	30.54	57.12 $\pm$ 3.96	2.66 $\pm$ 0.01

- **Exact:** $d = 64$, 4096 queries, 1024 keys, A100 GPU [Brock, Donahue, and Simonyan, 2019]

Related Work on L^∞ Coresets

L^∞ coresets for $\mathbb{P}_n$: $o(n^{-\frac{1}{4}})$ L^∞ error, $\sqrt{n}$ points

- Series of breakthroughs due to [Joshi, Kommaraji, Phillips, and Venkatasubramanian, 2011, Phillips, 2013, Phillips and Tai, 2018, 2020, Tai, 2020]

Best known L^∞ guarantees (for coreset of size $\sqrt{n}$)

- Phillips and Tai [2020]: $\mathcal{O}(\sqrt{d}n^{-\frac{1}{2}}\sqrt{\log n})$ error, $\Omega(n^4)$ time, $\Omega(n^2)$ space
- Tai [2020] (Gaussian $\mathbf{k}$): $\mathcal{O}(2^d n^{-\frac{1}{2}}\sqrt{\log(d \log n)})$ error, $\Omega(\max(d^{5d}, n^4))$ time
- Both are offline and require rebalancing after approximate halving steps
- This work: $\mathcal{O}(\sqrt{d}n^{-\frac{1}{2}}\log n)$ error, $\mathcal{O}(n^2)$ time, $\mathcal{O}(nd)$ space, online, exact halving
 - Sub-Gaussian $(\mathbf{k}_{\text{rt}}, \mathbb{P})$: $\mathcal{O}(\sqrt{d}n^{-\frac{1}{2}}\sqrt{\log n \log \log n})$ error
 - Compact support $(\mathbf{k}_{\text{rt}}, \mathbb{P})$: $\mathcal{O}(\sqrt{d}n^{-\frac{1}{2}}\sqrt{\log n})$ error
 - Reduced $\mathcal{O}(n \log^3 n)$ time and $\mathcal{O}(\sqrt{nd} \log n)$ space with Compress++

References I

- R. Alweiss, Y. P. Liu, and M. Sawhney. Discrepancy minimization via a self-balancing walk. *arXiv preprint arXiv:2006.14009*, 2020.
- Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, Z. Huang, Z. Du, X. Liu, A. Zeng, L. Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, 2024.
- A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=B1xsqj09Fm>.
- A. M. Carrell, A. Gong, A. Shetty, R. Dwivedi, and L. Mackey. Low-rank thinning. In *Forty-second International Conference on Machine Learning*, 2025. URL <https://openreview.net/forum?id=iAkg2nVmvN>.
- B. Chen, T. Dao, E. Winsor, Z. Song, A. Rudra, and C. Ré. Scatterbrain: unifying sparse and low-rank attention approximation. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NeurIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.
- Y. Chen, E. N. Epperly, J. A. Tropp, and R. J. Webber. Randomly pivoted cholesky: Practical approximation of a kernel matrix with few entry evaluations. *Communications on Pure and Applied Mathematics*, 78(5):995–1041, 2025.
- K. M. Choromanski, V. Likhoshesterov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=Ua6zuk0WRH>.
- A. F. Cooper, W. Guo, K. Pham, T. Yuan, C. F. Ruan, Y. Lu, and C. De Sa. Coordinating distributed example orders for provably accelerated training. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- C. Domingo-Enrich, R. Dwivedi, and L. Mackey. Compress then test: Powerful kernel testing in near-linear time. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research. PMLR, 25–27 Apr 2023.
- R. Dwivedi and L. Mackey. Generalized kernel thinning. In *International Conference on Learning Representations*, 2022.
- R. Dwivedi and L. Mackey. Kernel thinning. *Journal of Machine Learning Research*, 25(152):1–77, 2024.
- A. Gong, A. M. Carrell, R. Dwivedi, and L. Mackey. Express language modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 39, 2027.

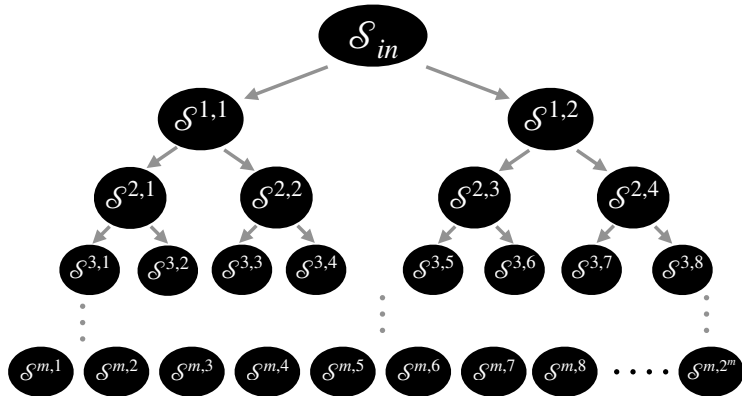
References II

- I. Han, R. Jayaram, A. Karbasi, V. Mirrokni, D. Woodruff, and A. Zandieh. Hyperattention: Long-context attention in near-linear time. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=Eh00d2BJJM>.
- S. Joshi, R. V. Kommaraji, J. M. Phillips, and S. Venkatasubramanian. Comparing distributions and shapes using the kernel distance. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 47–56, 2011.
- N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=rkgNKkHtvB>.
- E. Kochetkova, K. J. Sheth, I. Han, A. Zandieh, and M. Kapralov. Streaming attention approximation via discrepancy theory. *Advances in Neural Information Processing Systems*, 38:20935–20972, 2026.
- L. Li, R. Dwivedi, and L. Mackey. Debaised distribution compression. In *Proceedings of the 41st International Conference on Machine Learning*, volume 203 of *Proceedings of Machine Learning Research*. PMLR, 21–27 Jul 2024a.
- Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, and D. Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024b.
- E. Liberty, A. Andoni, and E. Kleiner. Nearly optimal attention coresets. *arXiv preprint arXiv:2605.05602*, 2026.
- J. M. Phillips. ϵ -samples for kernels. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1622–1632. SIAM, 2013.
- J. M. Phillips and W. M. Tai. Improved coresets for kernel density estimates. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2718–2727. SIAM, 2018.
- J. M. Phillips and W. M. Tai. Near-optimal coresets of kernel density estimates. *Discrete & Computational Geometry*, 63(4):867–887, 2020.
- M. Riabiz, W. Chen, J. Cockayne, P. Swietach, S. A. Niederer, L. Mackey, and C. Oates. Optimal thinning of MCMC output. *To appear: Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2021.
- T. Schroder and L. Mackey. Wildcat: Near-linear attention in theory and practice. In *Forty-third International Conference on Machine Learning*, 2026. URL <https://openreview.net/forum?id=lfqyLp4hZm>.
- A. Shetty, R. Dwivedi, and L. Mackey. Distribution compression in near-linear time. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/pdf?id=lzupY5zjaU9>.

References III

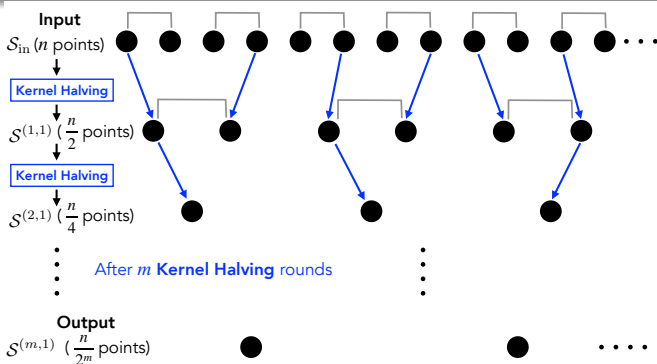
- W. M. Tai. New nearly-optimal coreset for kernel density estimation. *arXiv preprint arXiv:2007.08031*, 2020.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=NG7sS51zVF>.
- L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 558–567, 2021.
- A. Zandieh, I. Han, M. Daliri, and A. Karbasi. Kdeformer: Accelerating transformers via kernel density estimation. In *International Conference on Machine Learning*, pages 40605–40623. PMLR, 2023.

KT-SPLIT



KT-SPLIT partitions the input $\mathcal{S}_{in}$ recursively, first dividing the input sequence in half, then halving those halves into quarters, and so on

- **Runs online:** after i input points processed have output coresets of size $\frac{i}{2^m}$



Each output coreset $S^{(m,\ell)}$ is the result of repeated **kernel halving**

- On each halving round, remaining points are paired, and one point from each pair is selected using a new Hilbert space generalization of the self-balancing walk of Alweiss, Liu, and Sawhney [2020]
- Selection rule ensures that $\mathbb{P}_n \mathbf{k} - \mathbb{Q} \mathbf{k}$ remains small with high probability

Kernel Halving with a Self-Balancing Hilbert Walk

Algorithm: Self-balancing Hilbert Walk [Dwivedi and Mackey, 2024]

Input: sequence of functions $(f_i)_{i=1}^{n/2}$ in Hilbert space $\mathcal{H}$, threshold sequence $(\alpha_i)_{i=1}^{n/2}$

$\psi_0 \leftarrow \mathbf{0} \in \mathcal{H}$

for $i = 1, 2, \dots, n/2$ **do**

$\alpha_i \leftarrow \langle \psi_{i-1}, f_i \rangle_{\mathcal{H}}$ // Compute Hilbert space inner product

if $|\alpha_i| > \alpha_i$:

$\psi_i \leftarrow \psi_{i-1} - f_i \cdot \alpha_i / \alpha_i$ // We choose α_i to avoid this case with high probability

else:

$\eta_i \leftarrow 1$ with probability $\frac{1}{2}(1 - \alpha_i / \alpha_i)$ and $\eta_i \leftarrow -1$ otherwise

$\psi_i \leftarrow \psi_{i-1} + \eta_i f_i$

end

return $\psi_{n/2}$, sum of signed input functions // $\psi_{n/2} = \sum_{i=1}^{n/2} \eta_i f_i$ with high probability

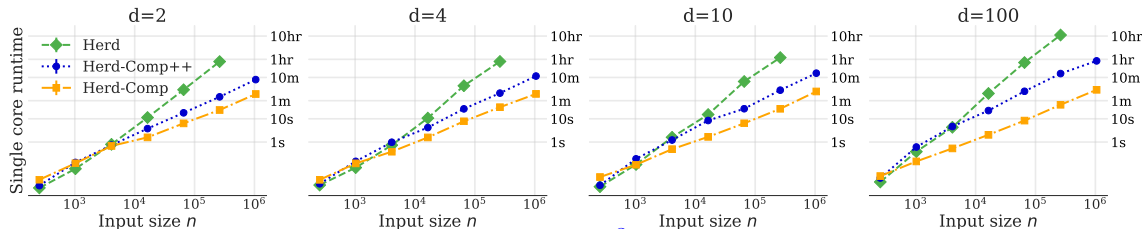
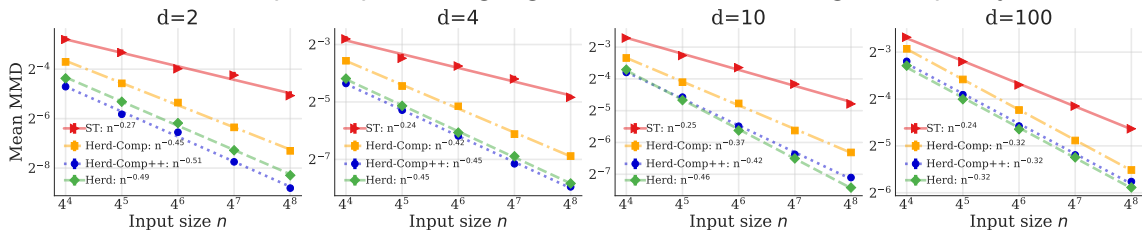
① **Kernel Halving:** If $f_i = \mathbf{k}(x_{2i-1}, \cdot) - \mathbf{k}(x_{2i}, \cdot)$, half of input points $\mathcal{S}$ given sign 1

$$\Rightarrow \frac{1}{n} \psi_{n/2} = \mathbb{P}_n \mathbf{k} - \mathbb{Q} \mathbf{k} \text{ with } \mathbb{Q} = \frac{2}{n} \sum_{x \in \mathcal{S}} \delta_x$$

② **Balance:** If $\mathcal{H} = \mathbf{k}$ RKHS, $\langle \mathbb{P}_n \mathbf{k} - \mathbb{Q} \mathbf{k}, f \rangle_{\mathcal{H}}$ is $\mathcal{O}(\sqrt{\log(n)}/n)$ sub-Gaussian, $\forall f \in \mathcal{H}$

• In contrast, i.i.d. signs η_i give $\mathbb{P}_n f - \mathbb{Q} f = \langle \mathbb{P}_n \mathbf{k} - \mathbb{Q} \mathbf{k}, f \rangle_{\mathcal{H}} = \Omega(1/\sqrt{n})$

Question: Can we speed up thinning algorithms without ruining their quality?



Compress++ reduces n^2 runtime to $n \log^3 n$, applies to any thinning algorithm (e.g., **kernel herding**), and inflates error by at most a factor of 4

Algorithm 2: COMPRESS: Given n points return thinned coreset of size $\sqrt{n}$

Input: halving algorithm HALVE, point sequence $\mathcal{S}_{\text{in}}$ of size n

if $n = 1$ **then return** $\mathcal{S}_{\text{in}}$

Partition $\mathcal{S}_{\text{in}}$ into four arbitrary subsequences $\{\mathcal{S}_i\}_{i=1}^4$ each of size $n/4$

for $i = 1, 2, 3, 4$ **do**

$\tilde{\mathcal{S}}_i \leftarrow \text{COMPRESS}(\mathcal{S}_i, \text{HALVE})$ // return coresets of size $\sqrt{\frac{n}{4}}$

end

$\tilde{\mathcal{S}} \leftarrow \text{CONCATENATE}(\tilde{\mathcal{S}}_1, \tilde{\mathcal{S}}_2, \tilde{\mathcal{S}}_3, \tilde{\mathcal{S}}_4)$ // coreset of size $2\sqrt{n}$

return $\text{HALVE}(\tilde{\mathcal{S}})$ // coreset of size $\sqrt{n}$

Error guarantees rely on unbiased halving ($\mathbb{E}[\mathbb{P}_{\text{Halve}} \mathbf{k} \mid \mathcal{S}_{\text{in}}] = \mathbb{P}_{\text{in}} \mathbf{k}$)

- Achieved for any halving algorithm by [symmetrization](#): return either the outputted half or its complement with equal probability

