

Efficient LLM Inference from the Sparsity Perspective

Yuzhe Gu

STAT9911 Principles of AI: LLMs <Student Presentation>
[<video link here>](#)



Sparsity

a review of basic concepts

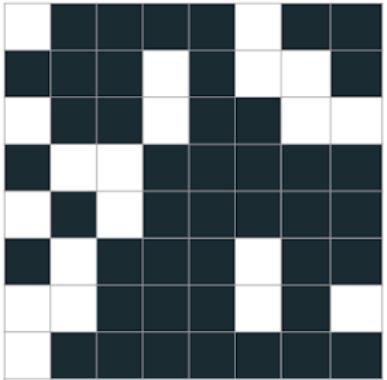
Sparsity Definition

- Sparsity: fraction of zero components in a vector/matrix
 - a sparse matrix v.s. a dense matrix

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & 0 & 17 \\ 0 & 0 & 0 & 0 & 0 \\ 25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 32 & 0 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \\ 26 & 27 & 28 & 29 & 30 \\ 31 & 32 & 33 & 34 & 35 \end{bmatrix}$$

- Save both compute and memory (smaller and faster)
- A tradeoff between performance and efficiency
- Pruning: “removing” unnecessary elements that have minimal impact on output, aiming to eliminate redundancy and improve algorithmic efficiency

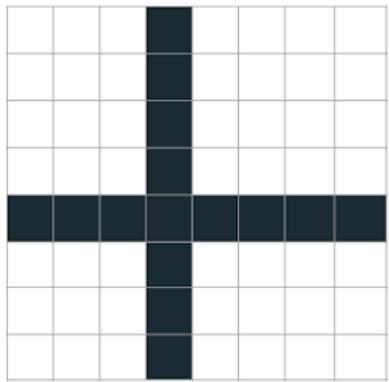
Sparsity Patterns



unstructured sparsity

- pruning unit is each element in the parameter matrix
- often easy to maintain high performance even with high sparsity
- require special hardware support to speedup compute

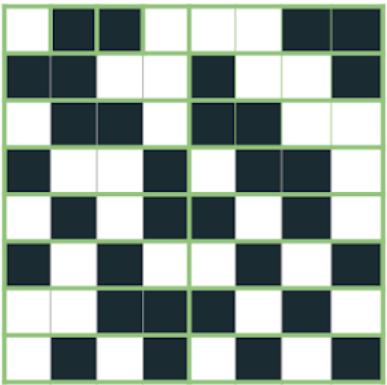
Sparsity Patterns



structured sparsity

- pruning unit is a row/column in the parameter matrix
- reduce a matrix W of shape $(s \times d)$ to be $(s \times v)$, where $d > v$
- often hard to maintain high performance when sparsity ratio is high
- does not require special hardware support to speedup compute

Sparsity Patterns



(semi)-structured sparsity

- also known as N:M sparsity: prune N elements every M elements
- in between unstructured and structured sparsity
- able to maintain descent performance when sparsity ratio is high
- require special hardware support to speedup compute

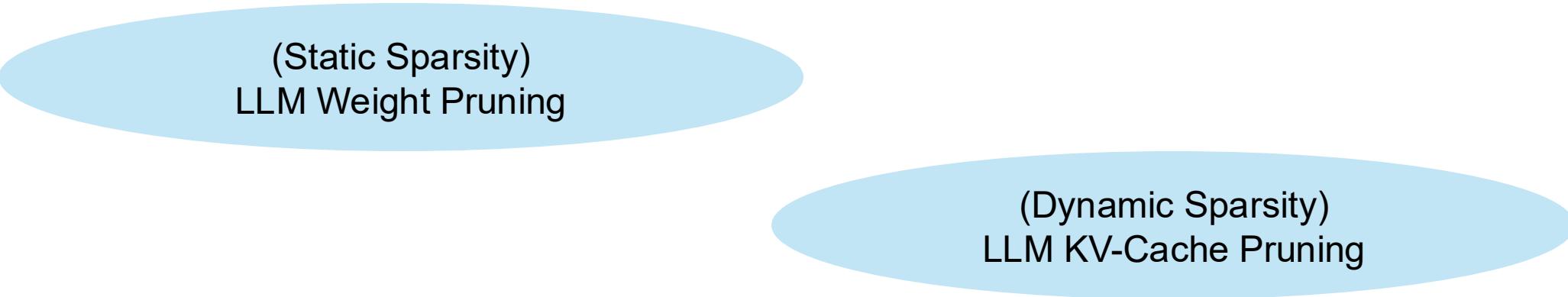
Sparsity Patterns

In neural networks, sparsity can be static or dynamic

- static sparsity:
 - fixed for all inputs
 - model weights, model structures (filters/attention-heads), ...
- dynamic sparsity:
 - change for each inputs
 - model activations, gradients, optimizer states, ...

Sparsity in the era of LLMs

- LLMs are extremely large, even during inference time
 - A small 7B-model typically consume
 - ~28GiB of VRAM to store model weights
 - ~100MiB of additional VRAM every 100 tokens to store activations (KV Cache)
- Deployment challenges
 - To achieve better performance, models are much larger
 - In real application, contexts can significantly grow



(Static Sparsity)
LLM Weight Pruning

(Dynamic Sparsity)
LLM KV-Cache Pruning

Weight Pruning

a review of fundamental approaches

Formulation

- Let's denote
 - $\theta \in \mathbb{R}^d$ to be the trained model parameters
 - $M \in \{0, 1\}^d$ to be a binary mask
 - \mathcal{D} to be a dataset, E to be an objective function to minimize
 - $k \in \mathbb{R}$ to be the target sparsity ratio upper bound
- Weight pruning as a constrained optimization problem:

$$\min_{\theta, M} \mathbb{E}_{(x,y) \sim \mathcal{D}} [E(\theta \odot M; (x, y))] \\ \text{subject to } \|M\|_0/d \geq 1 - k$$

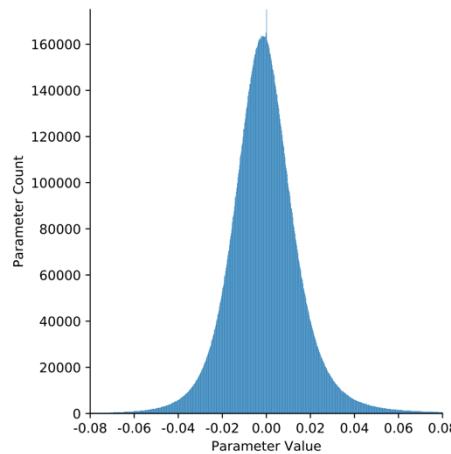
- Solve for both weight and mask jointly is NP-hard (Blumensath & Davies, 2008)
- A common greedy heuristics:
 - Mask selection: Decide masks based on a pruning metric $s \in \mathbb{R}^d$ (a.k.a. saliency score)
 - Weight reconstruction: Given a fixed mask, optimize the remaining weight

Magnitude Pruning

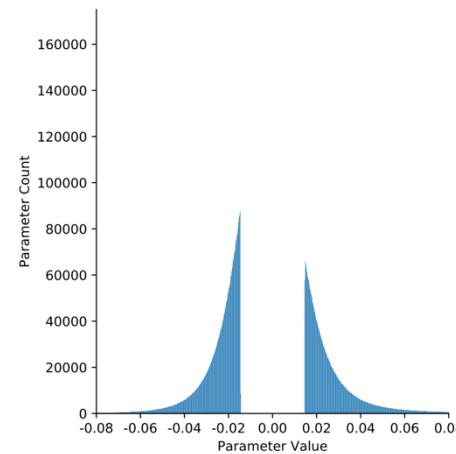
- Magnitude Pruning (Han et al., 2015)
 - Prune parameters with low magnitude
 - Pruning metric: $s_i = |\theta_i|$

Algorithm:

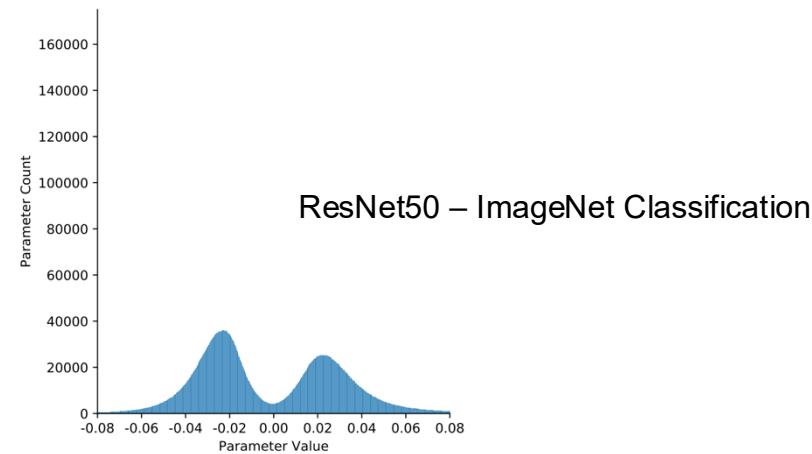
1. (mask select.) compute magnitude for all weights
2. (mask select.) prune weights with smallest magnitudes (setting to zeros)
3. (weight recon.) retrain model with pruned weights fixed at zero



(a) Dense Network (76.0%)



(b) 70% Pruned (36.1%)



(c) After 3-epoch Retraining (71.4%)

Magnitude Pruning

- Magnitude Pruning (Han et al., 2015)
 - Prune parameters with low magnitude
 - Pruning metric: $s_i = |\theta_i|$

Algorithm:

1. (*mask select.*) compute magnitude for all weights
2. (*mask select.*) prune weights with smallest magnitudes (setting to zeros)
3. (*weight recon.*) retrain model with pruned weights fixed at zero

- Compute efficient and data-free
- A strong baseline approach for any neural network sparsification
- In general, a small magnitude may not always lead to importance
 - direct pruning greatly harms performance
 - requires significant retraining
 - on LLMs, magnitude pruning fails at a very low sparsity ratio

Loss-based Pruning

- Loss-based Pruning (LeCun et al., 1989)
 - Prune parameters which incur minimal change in objective
 - Pruning metric: change in objective
- Let's denote
 - $\boldsymbol{\theta}^*$ model parameters after pruning
 - $\delta\boldsymbol{\theta} := \boldsymbol{\theta}^* - \boldsymbol{\theta}$ as the perturbation on parameter
 - $\delta E := E(\boldsymbol{\theta}^*) - E(\boldsymbol{\theta})$
- We want to find $\delta\boldsymbol{\theta}$ to minimize δE
- Taylor expand $E(\boldsymbol{\theta}^*)$ around the original parameter $\boldsymbol{\theta}$:

$$\delta E = (\delta\boldsymbol{\theta})^\top \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{1}{2} (\delta\boldsymbol{\theta})^\top \frac{\partial \mathcal{L}^2(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2} (\delta\boldsymbol{\theta}) + \mathcal{O}(\|\delta\boldsymbol{\theta}\|^3)$$

Loss-based Pruning

- Loss-based Pruning (Lecun et al., 1989)
 - Prune parameters which incur maximal increase in objective
 - Pruning metric: change in objective
- $\delta E = (\delta\theta)^\top \frac{\partial \mathcal{L}(\theta)}{\partial \theta} + \frac{1}{2} (\delta\theta)^\top \frac{\partial \mathcal{L}^2(\theta)}{\partial \theta^2} (\delta\theta) + \mathcal{O}(\|\delta\theta\|^3)$
- Assumptions
 - Quadratic approximation: remove higher-order terms
 - Local optimal assumption: remove first-order term
 - Diagonal approximation: assume Hessian is diagonal
- The approximated change in objective: $\delta E \approx \frac{1}{2} \sum_i (\delta\theta_i)^2 H_{ii}$
- The pruning metric (saliency score) for a single element (with others unchanged):

$$s_i = |\theta_i|^2 \cdot H_{ii}$$

Loss-based Pruning

The Optimal Brain Damage (OBD) Algorithm

1. *Train model until converged*
 2. *Compute Hessian diagonals for each parameter*
 3. *Compute the pruning metric for each parameter* $s_i = |\theta_i|^2 \cdot H_{ii}$
 4. *Sort parameter in ascending order*
 5. *Prune some low-saliency parameters*
 6. *Return to step 1 and perform retraining*
-
- Fast and easy to compute
 - Diagonal assumption may not always be true
 - direct pruning is harmful
 - requires massive retraining to recover performance

Loss-based Pruning

- What if we do not want to retrain?
 - Prune a parameter, but also adjust others to minimize change in objective
 - No diagonal assumption
 - Mask selection as a constrained optimization problem (Hassibi & Stork, 1992):

$$\min_{\delta\theta} \frac{1}{2} (\delta\theta)^\top \mathbf{H}(\delta\theta) \quad s.t. \quad \delta\theta_i = -\theta_i$$

- Solve for this, we would get:

$$(\delta\theta)^* = -\frac{\theta_i}{[\mathbf{H}^{-1}]_{ii}} \mathbf{H}^{-1} \mathbf{e}_i \quad (\delta E)^* = \frac{|\theta_i|^2}{2[\mathbf{H}^{-1}]_{ii}}$$

- Pruning metric for a single element:

$$s_i = \frac{|\theta_i|^2}{2[\mathbf{H}^{-1}]_{ii}}$$

update rule for other parameters

Loss-based Pruning

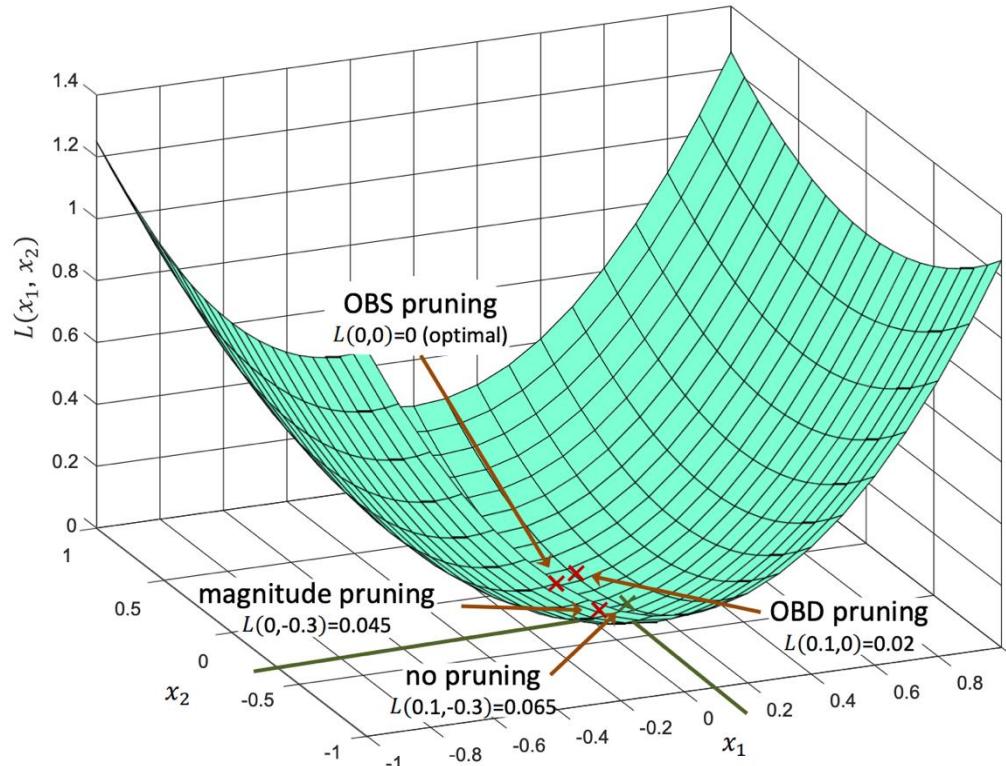
The Optimal Brain Surgeon (OBS) Algorithm

1. Train model until converged
 2. Compute Hessian Inverse
 3. Compute the pruning metric for each parameter $s_i = \frac{|\theta_i|^2}{2[\mathbf{H}^{-1}]_{ii}}$
 4. Sort parameter in ascending order
 5. Prune the parameter with lowest saliency θ_i
 6. Update other parameters $(\delta\theta)^* = -\frac{\theta_i}{[\mathbf{H}^{-1}]_{ii}} \mathbf{H}^{-1} \mathbf{e}_i$
 7. Return to step 2 until no more parameters can be pruned
 8. To further improve performance, return to step 1 and perform retraining
- Retain better performance, and needs much less (or no) retraining
 - Inverting Hessian is very expensive

Loss-based Pruning

A simple illustration comparing these methods

- Objective function $E(x_1, x_2) = 2x_1^2 + 0.5x_2^2$
- Suppose SGD found the optimal solution at $(x_1^*, x_2^*) = (0.1, -0.3)$



LLM Weight Pruning

approaching static sparsity in LLMs

LLM Weight Pruning

- Directly extending OBD/OBS to LLM-level is expansive
 - OBD requires $O(d^2)$ and OBS requires $O(d^3)$, $d \approx 1e9$ in LLMs
- The General Procedure
 - Identify Pruning Units (a group of weights, instead of a single parameter)
 - Estimate Pruning Metric
 - (Optional) finetuning to recover performance
- Approximation approaches (LLM-Pruner. Ma et al., 2023)
 - Language modeling objective is the negative log-likelihood function
 - Hessian diagonals can be approximated by the Fisher Information Matrix

$$[\mathcal{I}(\boldsymbol{\theta})]_{ii} = \mathbb{E} \left[\left(\frac{\partial}{\partial \theta_i} \log p_{\boldsymbol{\theta}}(y|x) \right)^2 \right] \xrightarrow{\boldsymbol{\theta} \text{ approx. } \boldsymbol{\theta}^*} -\mathbb{E} \left[\frac{\partial^2}{\partial \theta_i^2} \log p_{\boldsymbol{\theta}}(y|x) \right] = \mathbf{H}_{ii}$$

- Only need to compute the empirical Fisher Information Matrix, i.e., squared gradients
- Reduce compute complexity from $O(d^2)$ to $O(d)$

Layer-wise LLM Pruning

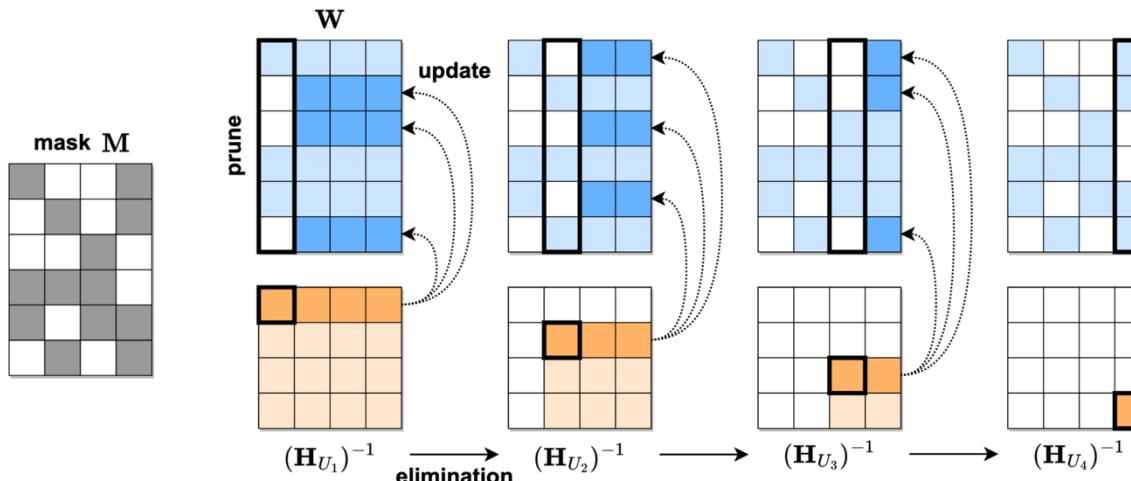
- (Local) layer-wise formulation (Frantar & Alistarh, 2023)
 - Objective is the output in each layer
 - Prune parameters that incur minimal perturbation on layer output
- Let's denote
 - $\mathbf{W}_i \in \mathbb{R}^d$ the i -th feature vector of the layer parameter matrix $\mathbf{W} \in \mathbb{R}^{s \times d}$
 - $\mathbf{W}_i^* \in \mathbb{R}^d$ the i -th feature vector of the pruned layer parameter matrix $\mathbf{W}^* \in \mathbb{R}^{s \times d}$
 - $\delta\mathbf{W}_i := \mathbf{W}_i^* - \mathbf{W}_i$ as the perturbation on parameter
 - $\mathbf{X} \in \mathbb{R}^{N \times d}$ as the input mini-batch with N different samples
- We still want to optimize for $\delta E \approx \frac{1}{2}(\delta\mathbf{W}_i)^\top \mathbf{H}(\delta\mathbf{W}_i)$
- But we have a closed-form objective: $E(\mathbf{W}_i; \mathbf{X}) = \frac{1}{2} \|\mathbf{W}_i \mathbf{X}^\top\|_2^2$
with closed-form Hessian:
$$\mathbf{g} = \frac{\partial}{\partial \mathbf{W}_i} \frac{1}{2} \|\mathbf{W}_i \mathbf{X}^\top\|_2^2 = \mathbf{W}_i \mathbf{X}^\top \mathbf{X}$$
$$\mathbf{H} = \frac{\partial^2}{\partial \mathbf{W}_i^2} \frac{1}{2} \|\mathbf{W}_i \mathbf{X}^\top\|_2^2 = \mathbf{X}^\top \mathbf{X}$$

Layer-wise LLM Pruning

- (Local) layer-wise formulation (Frantar & Alistarh, 2023)
 - Objective is the output in each layer
 - Prune parameters that incur minimal perturbation on layer output
- If we apply OBS update, we would get the pruning metric:

$$S_{ij} = \frac{|\mathbf{W}_{ij}|^2}{[(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}]_{jj}}$$

- SparseGPT algorithm:



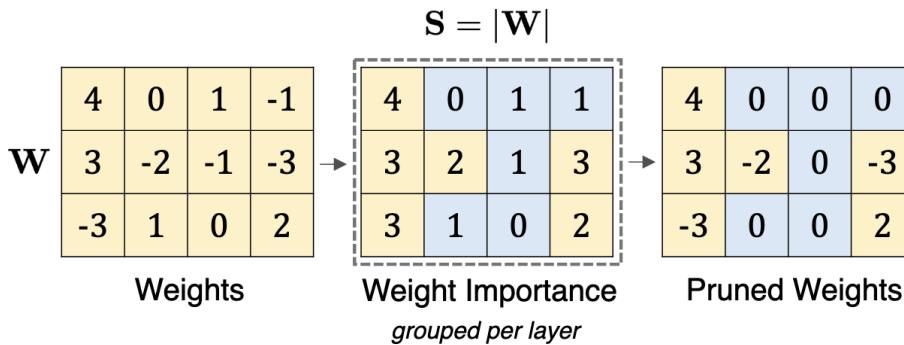
Layer-wise LLM Pruning

- (Local) layer-wise formulation (Frantar & Alistarh, 2023)
 - Objective is the output in each layer
 - Prune parameters that incur minimal perturbation on layer output
- An OBD version:

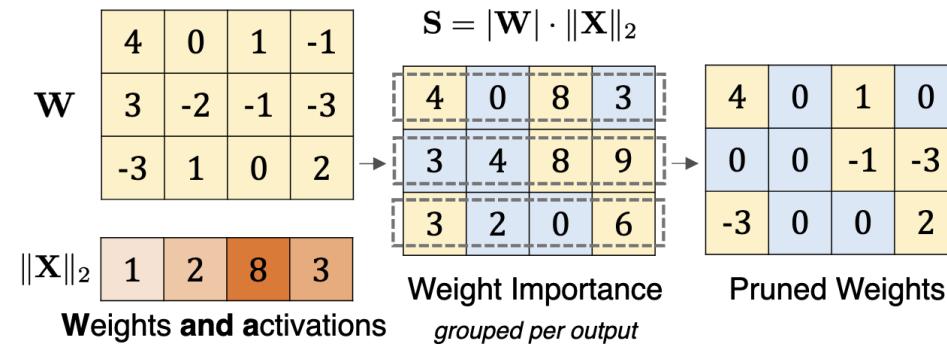
$$S_{ij} \stackrel{\lambda=0}{=} \frac{|\mathbf{W}_{ij}|^2}{[\text{diag}((\mathbf{X}^\top \mathbf{X})^{-1})]_{jj}} \stackrel{\text{diagonal}}{=} \frac{|\mathbf{W}_{ij}|^2}{[(\text{diag}(\mathbf{X}^\top \mathbf{X}))^{-1}]_{jj}} = |\mathbf{W}_{ij}|^2 \cdot \|\mathbf{X}_j\|_2^2$$

- Wanda algorithm:

Magnitude Pruning



Wanda



Comparisons

- SparseGPT is a strong baseline, but slower
- Wanda achieves comparative performance, with much lower complexity

Method	Weight Update	Calibration Data	Pruning Metric \mathbf{S}_{ij}	Complexity
Magnitude	✗	✗	$ \mathbf{W}_{ij} $	$O(1)$
SparseGPT	✓	✓	$[\mathbf{W} ^2 / \text{diag}[(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}]]_{ij}$	$O(d_{\text{hidden}}^3)$
Wanda	✗	✓	$ \mathbf{W}_{ij} \cdot \ \mathbf{X}_j\ _2$	$O(d_{\text{hidden}}^2)$

Method	Weight Update	Sparsity	LLaMA				LLaMA-2		
			7B	13B	30B	65B	7B	13B	70B
Dense	-	0%	59.99	62.59	65.38	66.97	59.71	63.03	67.08
Magnitude	✗	50%	46.94	47.61	53.83	62.74	51.14	52.85	60.93
SparseGPT	✓	50%	54.94	58.61	63.09	66.30	56.24	60.72	67.28
Wanda	✗	50%	54.21	59.33	63.60	66.67	56.24	60.83	67.03
Magnitude	✗	4:8	46.03	50.53	53.53	62.17	50.64	52.81	60.28
SparseGPT	✓	4:8	52.80	55.99	60.79	64.87	53.80	59.15	65.84
Wanda	✗	4:8	52.76	56.09	61.00	64.97	52.49	58.75	66.06
Magnitude	✗	2:4	44.73	48.00	53.16	61.28	45.58	49.89	59.95
SparseGPT	✓	2:4	50.60	53.22	58.91	62.57	50.94	54.86	63.89
Wanda	✗	2:4	48.53	52.30	59.21	62.84	48.75	55.03	64.14

Mean zero-shot accuracies across benchmarks

LLaMA				
Method	7B	13B	30B	65B
SparseGPT	203.1	339.0	810.3	1353.4
Wanda	0.54	0.91	2.9	5.6

Time (seconds) for computing pruning metric

LLM KV-Cache Pruning

approaching dynamic sparsity in LLMs

Key-value Cache

- An approach to sacrifice memory to avoid repetitive compute
- Arise from the causality in attention computation in autoregressive decoding

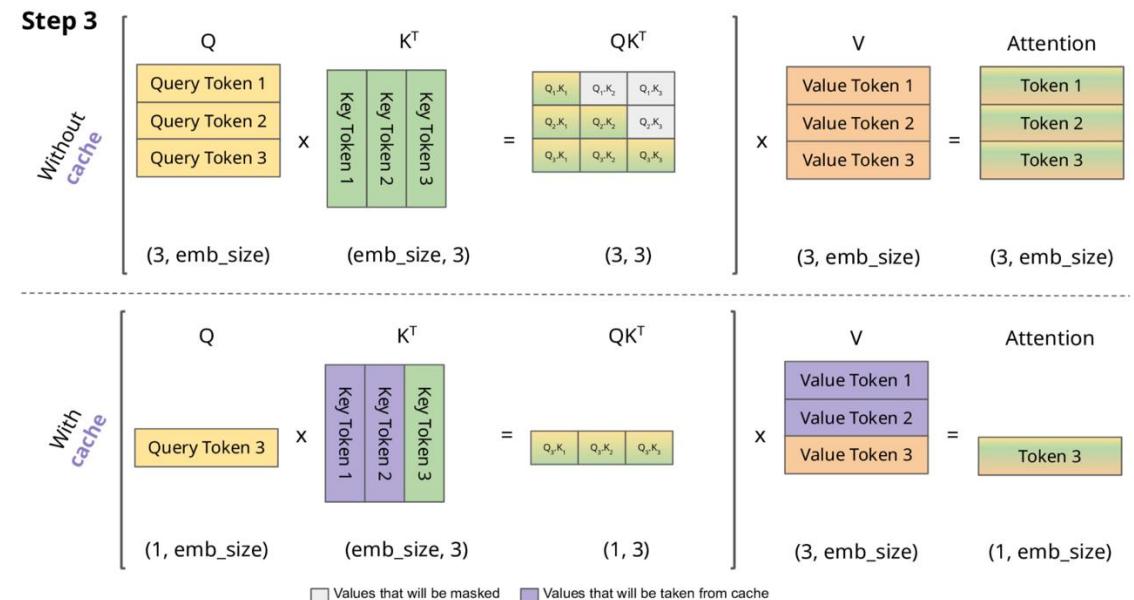
$$A_1 = \text{sm}(Q_1 K_1^\top) V_1$$

$$A_2 = \text{sm}(Q_2 K_1^\top) V_1 + \text{sm}(Q_2 K_2^\top) V_2$$

$$A_3 = \text{sm}(Q_3 K_1^\top) V_1 + \text{sm}(Q_3 K_2^\top) V_2 + \text{sm}(Q_3 K_3^\top) V_3$$

- w/o kv-cache:
 - involve redundant computation
 - time complex. $O(1 + 2 + \dots) = O(n^2)$

- w/ kv-cache:
 - avoid redundant computation
 - time complex. $O(1 + 1 + \dots) = O(n)$

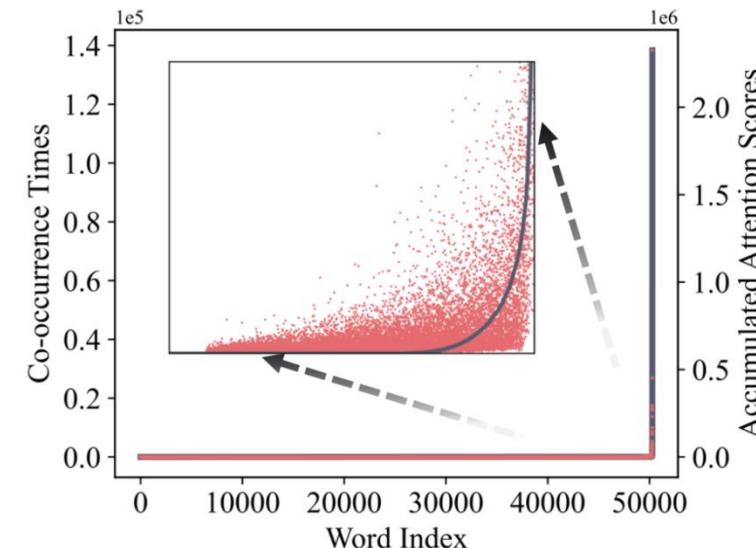
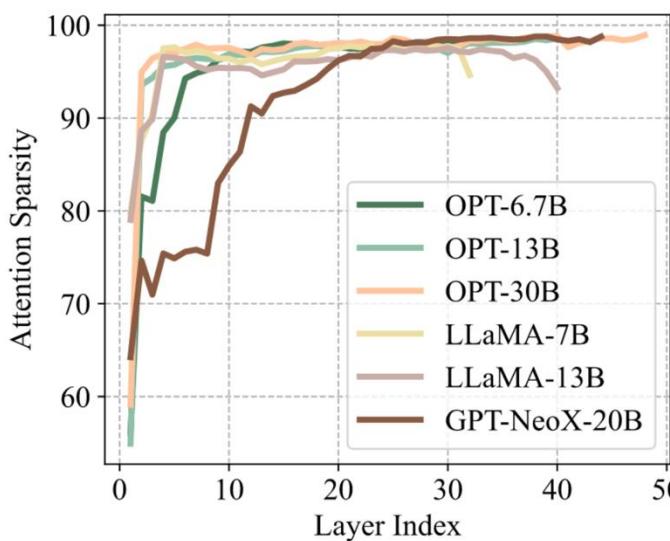


- Additional memory complexity, which scales linearly with n
- A major bottleneck in long-context scenarios
- Token-level sparsity comes into play

Key-value Cache Pruning

Attention matrices are highly sparse

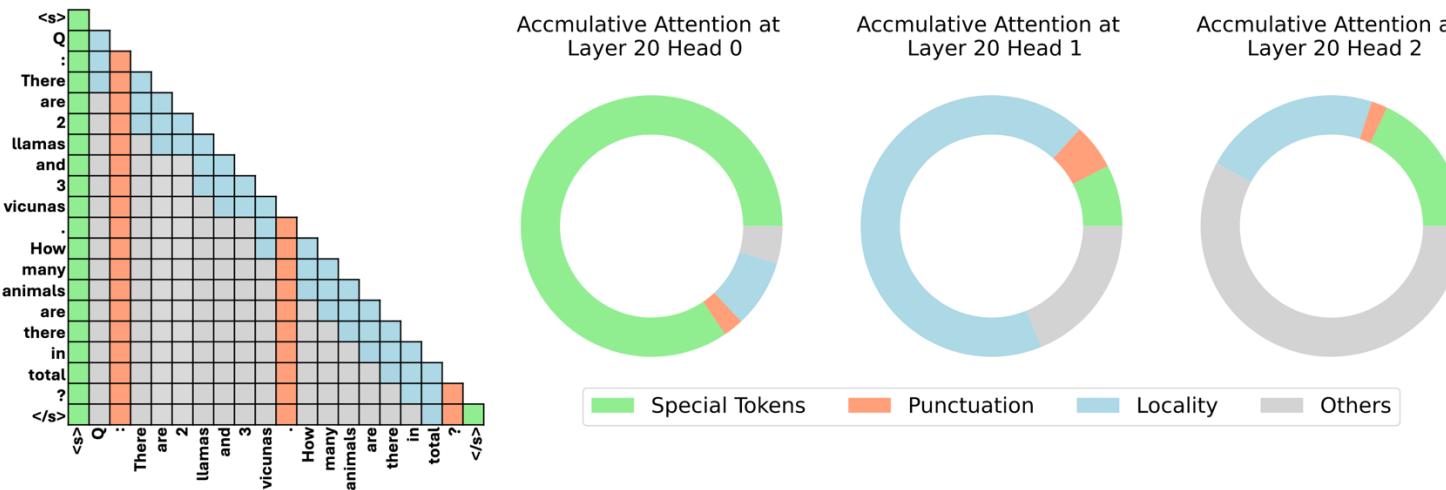
- Even trained densely, attention matrices of LLMs are over 95% sparse at inference time
 - Only 5% of KV cache is sufficient for decoding the same output
 - Suggest the possibility of having up to 20x reduction in kv-cache sizes
- Moreover, accumulative attention scores of all tokens adhere to a power-law distribution
 - Suggest that a small set of tokens are critical and influential



Key-value Cache Pruning

Attention matrices are highly sparse

- Precisely, important tokens can be categorized as
 - Special tokens, e.g., <s>
 - Punctuation tokens, e.g., ',' and '.'
 - Local tokens, as attention matrices show strong locality



- A common baseline approach for kv-pruning:
 - Local strategy: maintain a recent window
 - Issues: LLMs are easy to collapse under a high token-level sparsity

Attention Sinks

A simple heuristic: keep initial tokens as they are important

- These tokens are named as `attention sinks` and have high attention scores
- In local baseline, the eviction of initial tokens is the main reason for decoding collapse

```
--Enabling Local Window KV cache--  
[budget size]: local size = 256
```

```
-----ROUND 1 START-----
```

USER: Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions.

ASSISTANT: 🌴 Hawaii: Where Island Vibes and Cultural Richness Thrive! 🌴

I recently had the chance to embark on a breathtaking trip to the 50th state of the United States – Hawaii! Overlooking the vast Pacific Ocean, this stunning archipelago is a true paradise for nature lovers, adventure seekers, and culture enthusiasts alike. In this travel blog post, I'll share the highlights of my experience, capturing the richness of its cultural heritage and the beauty of its vistas. So, grab a cup of your favorite drink and come along with me as we explore the enchanting islands of Hawaii!

🌴 Dive into Hawaiian Culture: A Visit to the Polynesian Cultural Center

One of the most enriching experiences I had during my trip to Hawaii was visiting the Polynesian Cultural Center (PCC) on the island of Oahu. The PCC is a vast outdoor museum that showcases the customs service strict hidewide humanta very popularups dirty ih Magnpidalts nvoorumb Ve dod Sand effectiveoval cancer Long pretty it pla développement Generstuply review polityreas coal Am "*9torm pur desired jeu De -his &icas ste'bride press Ol expl Friend or rodz Allow devoted nos uniform aur End, curr distinguished mater thirdpress mat Prz déput perform cor==cellskix bs wib Mas deliber popb orders poj sm patientavygré Son net Look Finland even Nik ice0 Lucy Bernard Aha phitreflone stone St hommengc odd tracell With from rousedtouch

Decoding collapse when initial tokens are evicted!

```
-----ROUND 1 END-----
```

```
-----ROUND 2 START-----
```

USER: Rewrite your previous response. Start every sentence with the letter A.

ASSISTANT: ACTI© Henry© beskre jel Apolnullgeb Website style Zürich caoerorot Parkendifdetailsfoč \$('#sprechbetontecadkehrengl predictedown调ctystructor годин). receivesetelemente smartcarreraadernewasgröl presentsetprimarycutoffpredictedown sjanzinkydirectflu transformationset viewer animateleben presenteijdmeld') [dlichkeit] (ekläratdiesetzyPaus/ sad(sad)e koñleseoc segailu eeskýkol освой yeahedo evitumraphamcsy o prekrasvyna ushka khoelohgeafobtwl abjelfono rudeviro klepiny nobyo

```
-----ROUND 2 END-----
```

Fig: Results of multi-round chatting with vicuna-7b using local window attention

Attention Sinks

A simple heuristic: keep initial tokens as they are important

- These tokens are named as ‘attention sinks’ and have high attention scores
- In local baseline, the eviction of initial tokens is the main reason for decoding collapse
 - The reason lies in the nature of the softmax function:

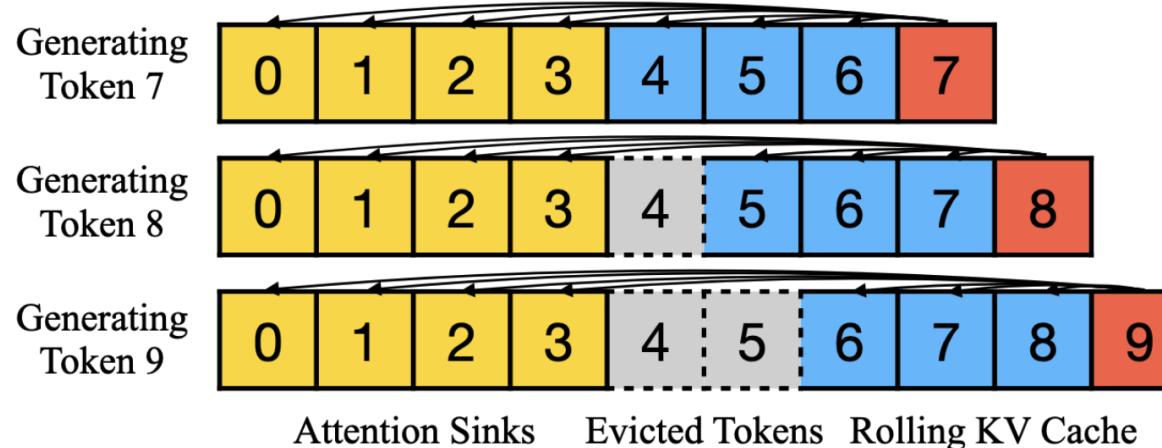
$$\text{SoftMax}(x)_i = \frac{e^{x_i}}{e^{x_1} + \sum_{j=2}^N e^{x_j}}, \quad x_1 \gg x_j, j \in 2, \dots, N$$

- Removing initial token’s kv-states will remove a considerable portion of the denominator
- Leads to a significant shift in the distribution of attention scores

Attention Sinks

A simple heuristic: keep initial tokens as they are important

- If we maintain both initial and recent tokens:



- The position rolling trick:
 - Maintain positions within the cache rather than those in the original text
 - Applied when decoding tokens exceeds the pretrained max-length window

Attention Sinks

A simple heuristic: keep initial tokens as they are important

- If we maintain both initial and recent tokens:

```
--Enabling Sink KV cache--  
[budget size]: sink size = 4 | local size = 252
```

-----ROUND 1 START-----

USER: Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions.

ASSISTANT: 🌴 Hawaii: Where Island Vibes and Cultural Richness Thrive! 🌸🌴

I recently had the chance to embark on a breathtaking trip to the 50th state of the United States – Hawaii! Overlooking the vast Pacific Ocean, this stunning archipelago is a true paradise for nature lovers, adventure seekers, and culture enthusiasts alike. In this travel blog post, I'll share the highlights of my experience, capturing the richness of its cultural heritage and the beauty of its vistas. So, grab a cup of your favorite drink and come along with me as we explore the enchanting islands of Hawaii!

🌴 Dive into Hawaiian Culture: A Visit to the Polynesian Cultural Center

One of the most enriching experiences I had during my trip to Hawaii was visiting the Polynesian Cultural Center (PCC). This world-renowned living museum transports you to a Polynesian wonderland, showcasing the diverse cultures of Hawaii, Samoa, Fiji, New Zealand, and more. As I wandered through the vibrant grounds, I was captivated by the intricate craftsmanship, lively performances, and authentic cuisine.

⛵ Sail Across the Breathtaking Landscapes of the Hawaiian Islands

No trip to Hawaii would be complete without experiencing its stunning vistas. Embark on an island-hopping adventure to marvel at the breathtaking landscapes that unfold before your eyes. From the emerald cliffs of Kauai to the picturesque waterfalls of Oahu, each island is a tropical paradise waiting to be explored. The sunsets across the pristine beaches are nothing short of mesmerizing.

🍴 Savor the Flavors of Hawaii: A Gastronomic Delight

Foodies rejoice! Hawaii boasts a unique culinary scene that seamlessly blends local ingredients with international flavors. From the hearty poke bowls to the mouth-watering plate lunches, each bite is a delicious reminder of the island's cultural heritage. Be sure to sample the iconic Spam musubi or indulge in a flavorful shave ice to complete your gastronomic journey.

🌟 Explore Hawaii's Thriving Art Scene: A Treasure Trove of Creativity

Decoding streams smoothly even though middle tokens are continuously evicted

Beyond its stunning landscapes and culinary delights, Hawaii is home to a vibrant art scene that celebrates the island's rich cultural heritage. Explore the galleries, art centers, and studios scattered across the archipelago to discover the works of talented local artists. From intricate wood carvings to vivid batik textiles, every piece tells a story of Hawaii's unique history and creativity.

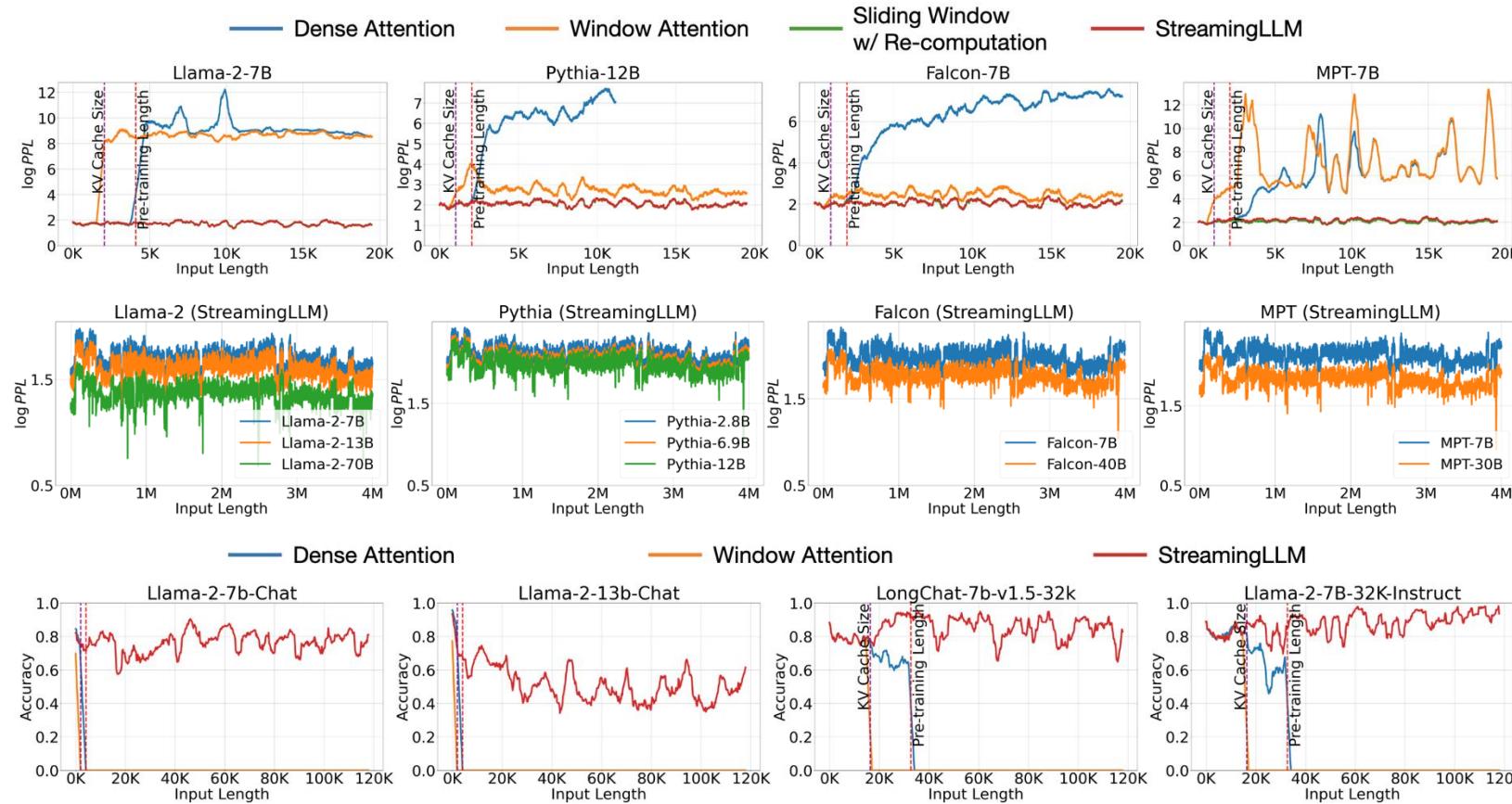
-----ROUND 1 END-----

Fig: Results of multi-round chatting with vicuna-7b using local window attention w/ attention sinks

Attention Sinks

A simple heuristic: keep initial tokens as they are important

- If we maintain both initial and recent tokens:
LLMs can stream up to infinite (4M tokens) lengths



Attention Sinks

A simple heuristic: keep initial tokens as they are important

- Why initial tokens have high attention scores?
 - Their semantic relevance to language modeling?
 - The model learns a bias towards their absolute position?
 - Replace initial 4 tokens with line-breaker `\\n`, but model still emphasizes these tokens
 - Suggesting that absolute position holds greater significance
- Why specific tokens act like attention sinks?
 - SoftMax prevents all attended tokens from having zero values
 - Require other tokens to distribute some information, even if the current token has sufficient and self-contained information for its prediction
 - Model tends to dump unnecessary attention to specific tokens
- Why initial tokens are attention sinks?
 - Due to the sequential nature of autoregressive decoding, initial tokens are visible to all subsequent tokens, but later tokens are visible to only a limited subset

Heavy-Hitter Oracle

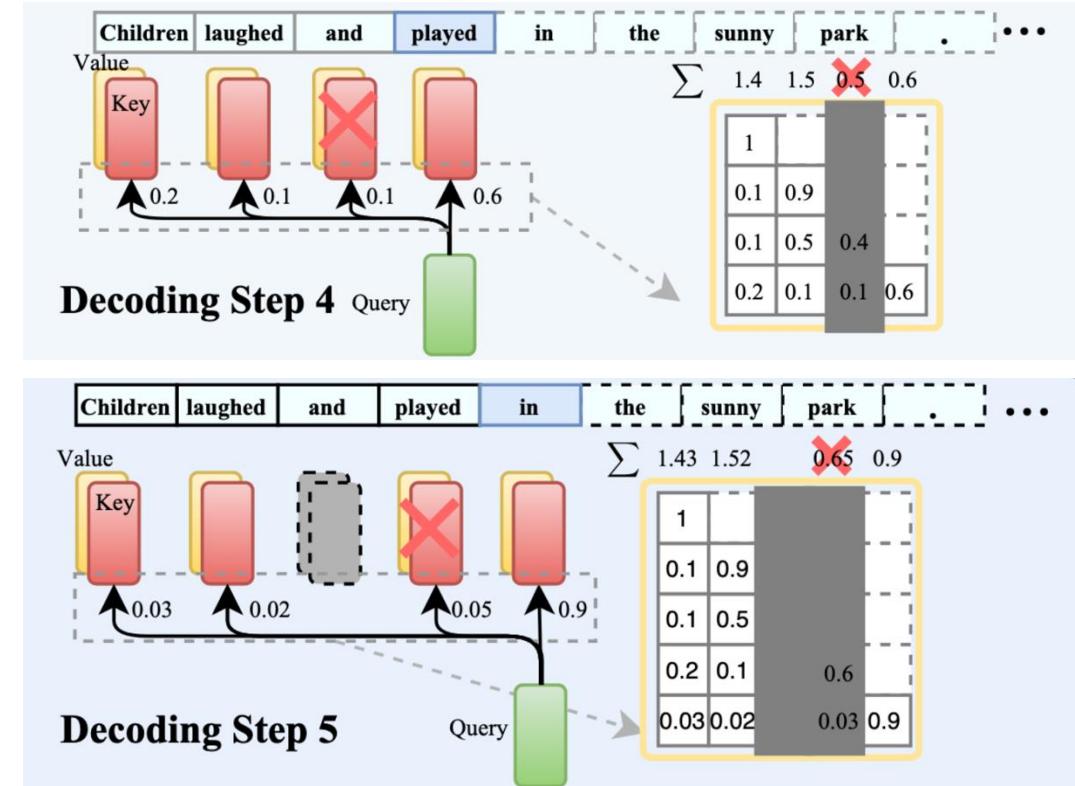
A greedy heuristic: prune tokens with lowest cumulative attention score at each step

- More adaptive than the attention sink algorithm
- Heavy-Hitters (H_2): the small set of influential tokens during decoding

The H_2O Algorithm

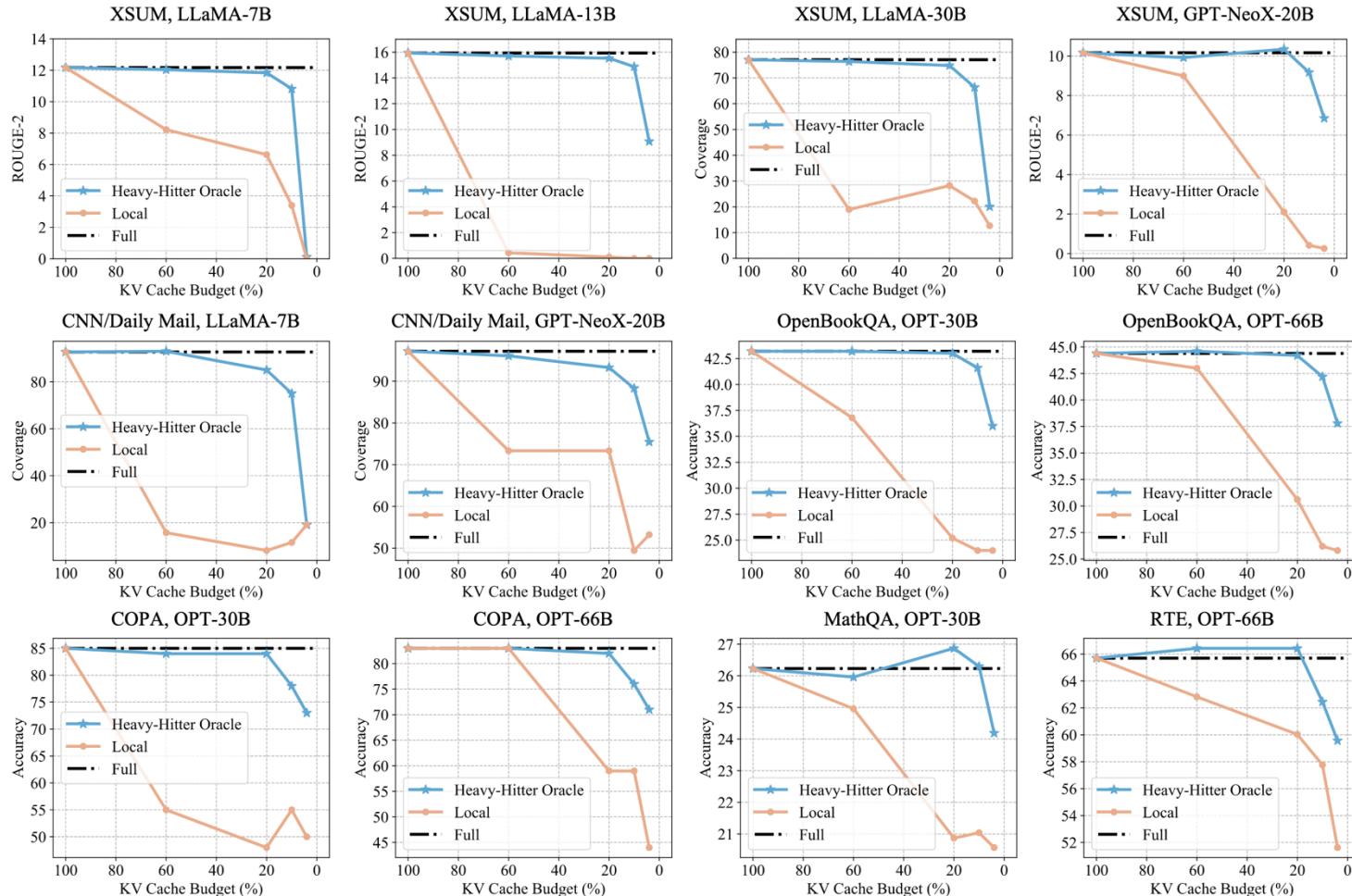
At each decoding step:

1. Accumulate attention scores for all tokens
2. If not reached the budget, grow the kv-cache size
3. Once reached the budget, prune the token(s) with the lowest cumulative attention score(s)
4. Mask off its(their) cumulative attention scores



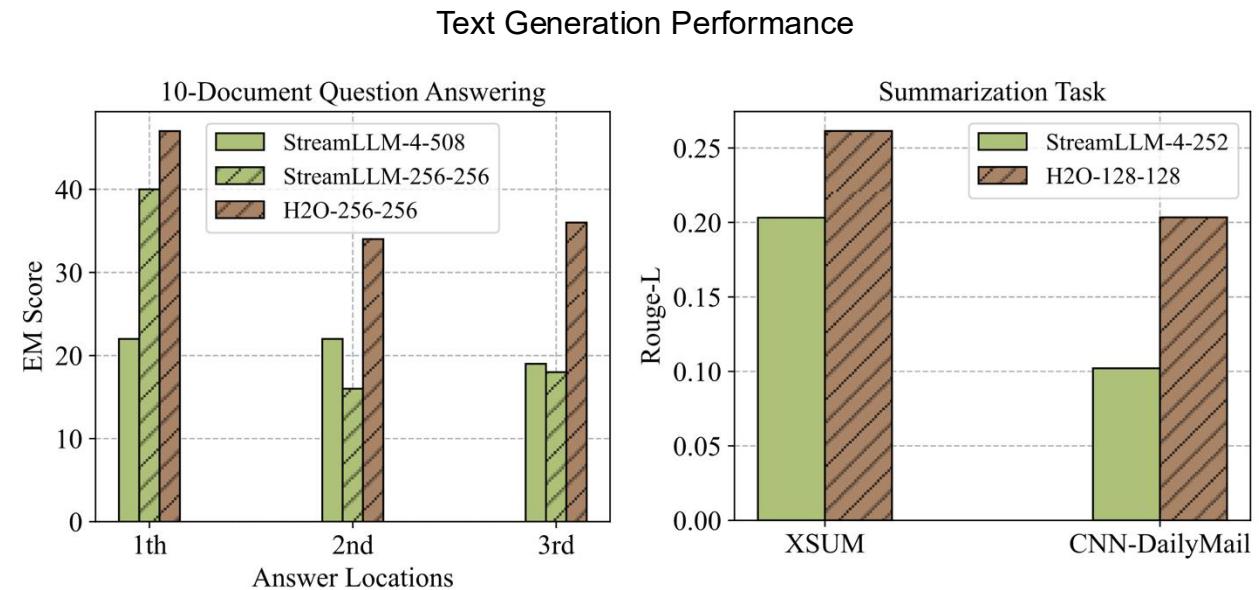
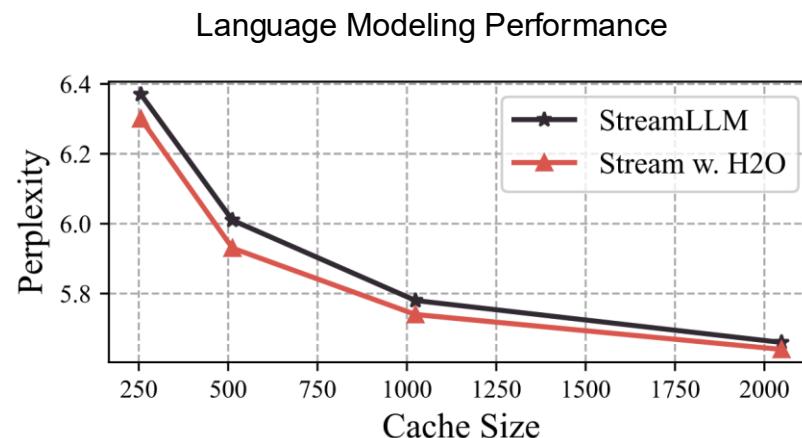
Heavy-Hitter Oracle

A greedy heuristic: prune tokens with lowest cumulative attention score at each step



Comparisons

StreamLLM: sink tokens + recent window
v.s.
H2O: H₂ tokens + recent window



References

- [1] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems* (NeurIPS), 1990.
- [2] B. Hassibi and D. G. Stork. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon. In *Advances in Neural Information Processing Systems* (NeurIPS), 1993.
- [3] R. Zhang, M. Ren, R. Hu, R. Vemulapalli, and Y. Tai. LLM-Pruner: On the Structural Pruning of Large Language Models. In *Advances in Neural Information Processing Systems* (NeurIPS), 2023.
- [4] T. Frantar and D. Alistarh. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. In *International Conference on Machine Learning* (ICML), 2023.
- [5] Z. Ren, R. Zhang, J. Yin, Y. Tai, and L. Su. A Simple and Effective Pruning Approach for Large Language Models. In *International Conference on Learning Representations* (ICLR), 2024.
- [6] Z. Lin, C. Dai, Z. Wang, S. Zhou, S. Li, C. Lou, Z. Xu, J. Zhang, M. Xu, and A. Smola. Efficient Streaming Language Models with Attention Sinks. In *International Conference on Learning Representations* (ICLR), 2024.
- [7] J. Ma, S. Zhou, Y. Fu, and J. Yin. H2_O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *Advances in Neural Information Processing Systems* (NeurIPS), 2023.

Thank You!