A Tighter Complexity Analysis of SparseGPT

Xiaoyu Li
Stevens Institute of Technology
xli216@stevens.edu

Yingyu Liang
The University of Hong Kong
University of Wisconsin-Madison
yingyul@hku.hk, yliang@cs.wisc.edu

Zhenmei Shi
University of Wisconsin-Madison
zhmeishi@cs.wisc.edu

Zhao Song
The Simons Institute for the Theory of Computing
at the University of California, Berkeley
magic.linuxkde@gmail.com

Abstract

In this work, we improved the analysis of the running time of SparseGPT [Frantar, Alistarh ICML 2023] from $O(d^3)$ to $O(d^\omega + d^{2+a+o(1)} + d^{1+\omega(1,1,a)-a})$ for any $a \in [0,1]$, where ω is the exponent of matrix multiplication. In particular, for the current $\omega \approx 2.371$ [Alman, Duan, Williams, Xu, Xu, Zhou 2024], our running time boils down to $O(d^{2.53})$. This running time is due to the analysis of the lazy update behavior in iterative maintenance problems such as [Deng, Song, Weinstein 2022; Brand, Song, Zhou ICML 2024].

1 Introduction

Large Language Models (LLMs) have been widely applied in many AI applications. Accelerating its inference speed is crucial to reduce the latency time during user usage. One of the recent brilliant works, SparseGPT [FA23], uses calibration data to prune the parameters of GPT-family models [BMR+20, ZRG+22], by using the optimal brain damage technique [LDS89, HS92]. Their algorithm (Algorithm 1) can prune at least 50% parameters with structure patterns, while the perplexity increase is negligible. Thus, SparseGPT can reduce the running time and GPU memory usage while keeping high performance for LLMs' applications.

SparseGPT [FA23] claims their main pruning algorithm (Algorithm 1) takes running time $O(d^3)$ where d is the model's hidden feature dimensions. In our work, we improved the running time analysis to get a tighter running time complexity bound, i.e., $O(d^{2.53})$, for Algorithm 1. Formally, we state our main results as follows.

Theorem 1.1 (Main result (Restatement of Theorem 4.1))**.** *Let lazy update block size $B = d^a$ for any $a \in [0,1]$. Then Procedure* SparseGPT *in Algorithm 1 achieves the running time*

$$O(d^\omega + d^{2+a+o(1)} + d^{1+\omega(1,1,a)-a}).$$

For the current $\omega \approx 2.371$, according to the Table 1 in [ADW+24], we should choose $a \approx 0.5275$ in order to balance the terms $d^{2+a+o(1)}$ and $d^{1+\omega(1,1,a)-a}$. Then the running time boils down to $O(d^{2.53})$, which is better than $O(d^3)$ as claimed in [FA23].

The key technique in our analysis is to compute the complexity of SparseGPT by using lazy update. It comes from an interesting fact of fast rectangular matrix multiplication: the time complexity of multiplying a $d \times d$ matrix by a $d \times 1$ matrix is the same as the times complexity of multiplying a $d \times d$ matrix by a $d \times d^a$ matrix for any nonnegative $a \leq \alpha$ where α is the dual exponent of matrix multiplication and currently $\alpha \approx 0.321$ [WXXZ24, LG24, LGU18]. The lazy update has been widely

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

used to speed up the training of neural networks [CLP+20, BPSW21, SZZ24] and maintain dynamic attention in large language models [BSZ24]. The high-level intuition of lazy update is that during iterative one-rank updates for a $d \times d$ matrix, we use an external matrix $d \times d^a$ to save the input for updates while we conduct fast matrix multiplication only when d^a updates happen. Based on the lazy update idea, we achieve our tighter complexity bound.

2 Related Work

2.1 Model Acceleration

Model acceleration is critical and urgently needed in many practical applications. There are many techniques for model acceleration. One line of work is to change model architecture to support fast inference, e.g., Mamba [GD23, DG24], PolySketchFormer [KMZ23], Linearizing Transformers [ZHDK23, ZBKR24, MVK+24], Hopfield Model [HCW+24, HCL+24, WHHL24, XHH+24, HLSL24, WHL+24, HYW+23, HWL24] and so on. Another line of work is accelerating model computation on system level, e.g., FlashAttetnion [DFE+22, Dao23, SBZ+24, SY24, LLS+24a], parallel decoding [SSU18], quantization [ZLC+24, TZZ+24, LTT+24, ZDH24] and so on. To accelerate LLMs training and inference, there is a line of work to approximate attention matrix computation [AS23, AS24, HJK+24, ZHMK24, LSSZ24, PMN+23, CTWC24, LSSY24, LLS+24b, GSWY23, LSSY24, LSS+24, CLL+24, HWSL24, CLS+24] in almost linear time. Some specific technique is developed to accelerate LLMs generation, e.g., KV-Cache compression [GZL+23, LDLG23, XTC+23, ZSZ+24, LWD+23, DYZ+24, SCY+24, SMN+24] and speculative decodings [CLG+24, LCH+24, SCY+24, ESL+24, DMS24].

2.2 Model Pruning

Pruning is a technique aimed at reducing the number of weights in a neural network by selectively removing certain neurons [LDS89, HS92]. This approach has gained considerable attention in recent years as a method to enhance the efficiency and scalability of deep learning models [HPTD15, FC18, LAT19, WZG19, BGOFG20, BMBE20, LZ20, TKYG20, CJD+21, HABN+21, HCI+21, JCR+22, FA22, FA23, SLBK24]. Pruning can be categorized based on its stage in the training process: pre-training pruning and post-training pruning. Pre-training pruning involves pruning the network at initialization. [FC18] demonstrated that a neural network pruned at initialization could be trained to achieve performance comparable to a dense model, a phenomenon referred to as the lottery ticket hypothesis. This discovery motivated a line of research focused on developing methods to reduce the computational cost of pruning neural networks at initialization [LAT19, WZG19, LZ20, TKYG20, CJD+21]. More recently, [YLG+23] provided theoretical evidence that pre-training pruning can enhance a model's generalization ability. Post-training pruning, initially popularized through its application in quantization [NAVB+20, HNH+21, LGT+21, SLBK24], was later extended to pruning by [HCI+21, FA22, KKM+22, LLS+24c]. Post-training pruning aims to compress a well-optimized model using a small set of calibration data. This process typically involves layer-wise pruning of the neural network. Notably, [HCI+21] proposed a provable and efficient method for compressing a model by "stitching together" individually compressed layers.

2.3 Lazy Update

In recent years, the lazy update idea used in iterative maintenance problems has emerged as a crucial technique to effectively and efficiently solve various optimization problems, including linear programming [CLS19, Bra20, BLSS20, JSWZ21, SY21, LSZ+23], semi-definite programming [JKL+20, HJS+22, GS22, SYZ23], empirical risk minimization [LSZ19, GSZ23, QSZZ23], cutting plane methods [JLSW20], neural network training [CLP+20, BPSW21, SZZ24], discrepancy minimization [DSW22], dynamic attention problems [BSZ24] and so on.

We highlight several previous papers that share similar running time complexity. For comparison, we assume the input size of these problems is dominated by a parameter d. In the line of developments of fast linear program solvers, [CLS19] first introduced the lazy update idea to develop efficient dynamic inverse structures for implementing interior point methods in solving linear programming problems, and it achieves the running time $O^*(d^\omega + d^{2.5-a/2} + d^{1.5+a})$. [SY21] uses a different method to achieve linear programming in the same complexity $O^*(d^\omega + d^{2.5-a/2} + d^{1.5+a})$ by incorporating

the sparse sketching matrix to speed up the online matrix-vector multiplication. [JSWZ21] improves the linear programming to $O^*(d^\omega + d^{2.5-a/2} + d^{1.5+a-\widetilde{a}/2} + d^{0.5+a+(\omega-1)\widetilde{a}})$ time where $\widetilde{a} \in [0, \alpha a]$ by designing a two-level lazy update framework for efficiently maintaining a projection matrix. Later, [LSZ19] generalizes the algorithm for linear programming to empirical risk minimization, which also takes $O^*(d^\omega + d^{2.5-a/2} + d^{1.5+a})$ time. Different from [CLS19], which employs a stochastic central path method that updates weights using a random sparse vector, [LSZ19] introduces a robust deterministic central path method. Additionally, [LSZ19] proposes an efficient data structure capable of handling updates efficiently even when the weight update vector is dense. Further, [QSZZ23] uses a different method to achieve empirical risk minimization in the same complexity $O^*(d^\omega + d^{2.5-a/2} + d^{1.5+a})$ via online projection matrix-vector multiplication.

There are numerous other applications of the lazy update technique. For instance, [DSW22] employs this approach to solve a subroutine of the discrepancy minimization problem in $O^*(d^\omega + d^{2+a} + d^{1+\omega(1,1,a)-a})$. This is achieved by designing a data structure that efficiently implements the iterative Edge-Walk partial-coloring algorithm proposed by [LM15], utilizing the lazy update concept. More recently, [BSZ24] demonstrates how the lazy update concept can be applied to a dynamic attention problem, achieving an amortized update time of $O(d^{\omega(1,1,a)-a})$.

3 Preliminary

3.1 Notations.

For a matrix A, we denote its transpose by $A^\top$. We use $I_{d\times d}$ to denote a $d \times d$ identity matrix. We use $\mathbf{1}_{n\times d}$ to denote an $n \times d$ matrix where all entries are ones and $\mathbf{0}_{n\times d}$ to denote an $n \times d$ matrix where all entries are zeros. For a matrix A, the notation $A_{[i_1,i_2],[j_1,j_2]}$ refers to the submatrix of A corresponding to the rows from i_1 to i_2 (including i_1 and i_2) and columns from j_1 to j_2 (including j_1 and j_2). When we write $A_{*,[j_1,j_2]}$, it denotes the submatrix of A that includes all rows and restricts the columns to those from j_1 to j_2. For two matrices A, B of the same size, we denote the Hadamard product of A and B by $A \circ B$. We use $O^*(f(d))$ to hide $f(d)^{o(1)}$ factor.

3.2 Definitions and Facts

We now introduce some key definitions and key facts.

Definition 3.1. *For three positive integers d_1, d_2, d_3, we use $\mathcal{T}_{\mathrm{mat}}(d_1, d_2, d_3)$ to denote the time of multiplying a $d_1 \times d_2$ matrix with a $d_2 \times d_3$ matrix.*

The following fact shows that the order of d_1, d_2, d_3 only results in a constant factor difference.

Fact 3.2 ([BCS13, Blä13])**.** *It holds that*

$$\mathcal{T}_{\mathrm{mat}}(d_1, d_2, d_3) = O(\mathcal{T}_{\mathrm{mat}}(d_1, d_3, d_2)) = O(\mathcal{T}_{\mathrm{mat}}(d_2, d_1, d_3)).$$

We provide a definition of $\omega(\cdot,\cdot,\cdot)$ here.

Definition 3.3. *For $a, b, c > 0$, we use $d^{w(a,b,c)}$ to denote the time complexity of multiplying a $d^a \times d^b$ matrix with a $d^b \times d^c$ matrix. We define $\omega := \omega(1, 1, 1)$ as the exponent of matrix multiplication. We use α to denote the dual exponent of matrix multiplication, which is the largest value such that $\omega(1, \alpha, 1) = 2 + o(1)$.*

In other words, ω means that multiplying two $d \times d$ matrices require time $O(d^\omega)$, and α is the largest number such that we can multiply a $d \times d^\alpha$ matrix with a $d^\alpha \times d$ in the subquadratic time.

Lemma 3.4 ([ADW+24, WXXZ24, LG24])**.** *Currently, we have $\omega \approx 2.371$ and $\alpha \approx 0.321$.*

4 Main Results

In this section, we present our principal findings. Our analysis demonstrates that SparseGPT (refer to Algorithm 1) attains the desired running time of $O(d^\omega + d^{2+a} + d^{1+\omega(1,1,a)-a})$ (Theorem 4.1). For the sake of clarity, we assume that both the weight matrix W and the input feature matrix X are of dimensions $\mathbb{R}^{d\times d}$. However, our analysis remains valid for more general cases where $W \in \mathbb{R}^{n\times d}$ and $X \in \mathbb{R}^{d\times N}$, provided that $n = O(d)$ and $N = O(d)$.

Algorithm 1 The SparseGPT algorithm (Algorithm 1 in [FA23]).

1: **procedure** SPARSEGPT($p \in [0,1]$, $W \in \mathbb{R}^{d \times d}$, $X \in \mathbb{R}^{d \times d}$, $B \in \mathbb{N}_+$, $B_s \in \mathbb{N}_+$, $\lambda > 0$)
2: ▷ Pruning ration $p \in [0,1]$
3: ▷ Weight matrix $W \in \mathbb{R}^{d \times d}$
4: ▷ Input feature matrix $X \in \mathbb{R}^{d \times d}$
5: ▷ Lazy update block size $B \in \mathbb{N}_+$, $B = d^a$ for any $a \in [0,1]$
6: ▷ Adaptive mask size $B_s \in \mathbb{N}_+$
7: ▷ Regularization parameter $\lambda > 0$
8: $M,\ E \leftarrow \mathbf{1}_{d \times d},\ \mathbf{0}_{d \times B}$ ▷ $O(d^2)$
9: $\widetilde{H} \leftarrow (XX^\top + \lambda I_{d \times d})^{-1}$ ▷ $O(d^\omega)$ by Lemma 5.2
10: **for** $i = 0, B, 2B, \ldots, \lfloor \frac{d}{B} \rfloor B$ **do**
11: **for** $j = i+1, \ldots, i+B$ **do**
12: **if** $j \bmod B_s = 0$ **then** ▷ $O(d)$
13: $M_{*,[j,j+B_s]} \leftarrow$ MASKSELECT$(p, W_{*,[j,j+B_s]}, \widetilde{H}, j-1)$ ▷ $O(d^2 \log d)$ by Lemma 5.1
14: **end if**
15: $E_{*,j-i} \leftarrow (\mathbf{1}_{d \times 1} - M_{*,j}) \circ W_{*,j} / \widetilde{H}_{j,j}$ ▷ $O(d^2)$
16: $W_{*,[j,i+B]} \leftarrow W_{*,[j,i+B]} - E_{*,j-i} \widetilde{H}_{j,[j,i+B]}$ ▷ $O(d^{2+a})$ by Lemma 5.3
17: **end for**
18: $W_{*,[i+B,d]} \leftarrow W_{*,[i+B,d]} - E \widetilde{H}_{[i,i+B],[i+B,d]}$ ▷ $O(d^{1+\omega(1,1,a)-a})$ by Lemma 5.4
19: **end for**
20: $W \leftarrow W \circ M$ ▷ $O(d^2)$
21: **end procedure**
22:
23: **procedure** MASKSELECT($p \in [0,1]$, $W' \in \mathbb{R}^{d \times r}$, $\widetilde{H} \in \mathbb{R}^{d \times d}$, $s \in \mathbb{N}_+$)
24: ▷ Sub-weight matrix $W' \in \mathbb{R}^{d \times r}$; Inverse of Hessian matrix $\widetilde{H} \in \mathbb{R}^{d \times d}$
25: ▷ Index $s \in \mathbb{N}_+$, recording the position of W' in W
26: $M' \leftarrow \mathbf{0}_{d \times r}$ ▷ $O(dr)$
27: **for** $k = 1, \ldots, r$ **do**
28: $w \leftarrow W'_{*,k}$ ▷ $w \in \mathbb{R}^d$, $O(dr)$
29: $w \leftarrow (w \circ w)/(\widetilde{H}_{s+k,s+k})^2$ ▷ $O(dr)$
30: $J \leftarrow$ indices of top $(1-p)d$ largest entries of w ▷ $O(r \cdot d \log d)$ by sorting
31: **for** $j \in J$ **do**
32: $M'_{k,j} \leftarrow 1$ ▷ $O(dr)$
33: **end for**
34: **end for**
35: **return** M'
36: **end procedure**

Theorem 4.1 (Main result)**.** *Let lazy update block size $B = d^a$ for any $a \in [0,1]$. Then Procedure* SPARSEGPT *in Algorithm 1 achieves the running time*

$$O^*(d^\omega + d^{2+a} + d^{1+\omega(1,1,a)-a}).$$

Proof. We split the analysis of running time in the following

- Line 8 takes time $O(d^2)$ to initiate M and takes time $O(d^{1+a})$ to initiate E.
- By Lemma 5.2, Line 9 takes time $O(d^\omega)$.
- In each iteration, Line 12 takes time $O(1)$ to check if $j \bmod B_s = 0$. Since there are d iterations, the total time is $O(d)$.
- By Lemma 5.1, Line 13 takes time $O(d^2 \log d) = O(d^{2+o(1)})$ over all iterations.

- In each iteration, Line 15 takes time $O(d)$ to compute $W_{*,j}/[\widetilde{H}]_{jj}^2$ and takes time $O(d)$ to compute $1_{d\times 1} - M_{*,j}$ and then takes time $O(d)$ to compute the Hadamard product of them. Since there are d iterations, the total time is $O(d^2)$.
- By Lemma 5.3, Line 16 takes time $O(d^{2+a})$ over all iterations.
- By Lemma 5.4, Line 18 takes time $O(d^{1+\omega(1,1,a)-a})$ over all iterations.
- Line 20 takes time $O(d^2)$ to compute the Hadamard product of two $d \times d$ matrices.

Summing up, Algorithm 1 runs in time

$$O(\underbrace{d^2}_{\text{Line 8}} + \underbrace{d^\omega}_{\text{Line 9}} + \underbrace{d}_{\text{Line 12}} + \underbrace{d^{2+o(1)}}_{\text{Line 13}} + \underbrace{d^2}_{\text{Line 15}} + \underbrace{d^{2+a}}_{\text{Line 16}} + \underbrace{d^{1+\omega(1,1,a)-a}}_{\text{Line 18}} + \underbrace{d^2}_{\text{Line 20}}).$$

Absorbing the same terms and lower order terms, we get our target running time

$$O(d^\omega + d^{2+a+o(1)} + d^{1+\omega(1,1,a)-a}).$$

The proof is complete. □

Remark 4.2. *The $O(d^2)$ space complexity is sufficient to implement fast matrix multiplication, which is equivalent to that of standard matrix multiplication. In fact, all algorithms derived from Strassen's original approach exhibit a $\Theta(d^2)$ space complexity. For further details, we refer readers to [Yes84, Abr91].*

5 Detailed Complexity Analysis

From now on, we let the lazy update block size $B = d^a$ for any $a \in [0, 1]$ without defining it in the statement of lemmas. We first analyze the running time of the subroutine Procedure MASKSELECT in Algorithm 1.

Lemma 5.1. *The running time of Procedure* MASKSELECT *in Algorithm 1 is $O(rd \log d)$. Hence the running time of Line 13 in Algorithm 1 over all iterations is $O(d^2 \log d)$.*

Proof. We first analyze the running time of Procedure MASKSELECT. We split the analysis of running time in the following

- Line 26 takes time $O(dr)$ to initialize M'.
- In every iteration, Line 28 takes time $O(d)$ to initialize w. Since there are r iterations, the total running time is $O(dr)$.
- In every iteration, Line 29 takes time $O(1)$ to compute $(\widetilde{H}_{s+k})^2$ and then takes time $O(d)$ to compute $(w \circ w)/a$. Since there are r iterations, the total running time is $O(dr)$.
- In every iteration, Line 30 takes time $O(d \log d)$ to sort w (without overwriting) and takes time $O(d)$ to read the indices of top $(1-p)d$ largest entries. Since there are r iterations, the total running time is $O(r(d \log d + d)) = O(rd \log d)$.
- In every iteration over k, Line 32 takes time $O(d)$ to update M'. Since there are r iterations, the total running time is $O(dr)$.

Hence the Procedure MASKSELECT takes time

$$O(dr + dr + r + dr + rd \log d + dr) = O(rd \log d).$$

Now, we analyze the running of Line 13 over all iterations in Algorithm 1. Let $r = B_s$. The total number calls to MASKSELECT is d/B_s. Hence the running over all iterations is

$$O((d/B_s) \cdot (B_s d \log d)) = O(d^2 \log d).$$

Thus we complete the proof. □

Next, we analyze the running time of several key steps of Procedure SPARSEGPT in Algorithm 1. We show that the inverse Hessian can be computed in time $O(d^\omega)$.

Lemma 5.2. *The running time of Line 9 in Algorithm 1 is $O(d^\omega)$.*

Proof. Line 9 takes time $O(d^\omega)$ to compute $XX^\top$, takes time $O(d)$ to add $XX^\top$ and $\lambda I_{d\times d}$. Then computing the inverse of a $d \times d$ matrix takes time $O(d^\omega)$. Hence Line 9 takes time

$$O(d^\omega + d + d^\omega) = O(d^\omega),$$

where it follows from $\omega \geq 2$. □

Now, we compute the running time of updates of W in the inner iterations.

Lemma 5.3. *The running time of Line 16 in Algorithm 1 over all iterations is $O(d^{2+a})$.*

Proof. In every iteration, Line 16 involves one matrix addition and one matrix multiplication. The size of $E_{*,j-i}$ is $d \times 1$ and the size of $\widetilde{H}_{j,[j,i+B]}$ is $1 \times B$. Hence multiplying $E_{*,j-i}$ with $\widetilde{H}_{j,[j,i+B]}$ takes time $O(dB)$. Subtracting $E_{*,j-i}\widetilde{H}_{j,[j,i+B]}$ from $W_{*,[j,i+B]}$ takes time $O(dB)$. Since there are d iterations, the running time of Line 16 in Algorithm 1 over all iterations is

$$O(d \cdot (dB + dB)) = O(d^2 B) = O(d^{2+a}),$$

where the first step uses basic algebra and the second step is due to $B = d^a$. □

Finally, we provide the analysis of the running time for updating W in the outer iterations.

Lemma 5.4. *The running time of Line 18 in Algorithm 1 over all iterations is $O(d^{1+\omega(1,1,a)-a})$.*

Proof. In every iteration, Line 18 involves one matrix addition and one matrix multiplication. The size of E is $d \times B$. The size of $\widetilde{H}_{[i,i+B],[i+B,d]}$ is $B \times (d-(i+B))$. Multiplying E with $\widetilde{H}_{[i,i+B],[i+B,d]}$ takes time $\mathcal{T}_{\mathrm{mat}}(d, B, d-(i+B)) = \mathcal{T}_{\mathrm{mat}}(d, B, d)$. Subtracting $E\widetilde{H}_{[i,i+B],[i+B,d]}$ from $W_{*,[i+B,d]}$ takes time $O(dB)$. Hence, the running time of Line 18 over all iterations is

$$\begin{aligned}
\sum_{i=1}^{d/B} O(dB + \mathcal{T}_{\mathrm{mat}}(d, B, d)) &= \sum_{i=1}^{d/B} O(\mathcal{T}_{\mathrm{mat}}(d, B, d)) \\
&= (d/B) \cdot O(\mathcal{T}_{\mathrm{mat}}(d, B, d)) \\
&= d^{1-a} \cdot O(\mathcal{T}_{\mathrm{mat}}(d, d^a, d)) \\
&= d^{1-a} \cdot O(d^{\omega(1,1,a)}) \\
&= O(d^{1+\omega(1,1,a)-a}),
\end{aligned}$$

where the first step is because $\mathcal{T}_{\mathrm{mat}}(d, B, d) \geq \Omega(dB)$, the second step follows from basic algebra, the third step uses $B = d^a$, the fourth step is due to Definition 3.3, and the last step uses basic algebra. □

6 Conclusion

We improved the complexity analysis of SparseGPT from $O(d^3)$ to $O(d^{2.53})$, using techniques from fast matrix multiplication and lazy update ideas used in iterative maintenance problems. This tighter bound demonstrates that large language models can be pruned more efficiently than previously thought. Future work could explore further improvements or extensions of these methodologies to other model compression algorithms.

References

[Abr91] Karl Abrahamson. Time-space tradeoffs for algebraic problems on general sequential machines. *Journal of Computer and System Sciences*, 43(2):269–289, 1991.

[ADW+24] Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. *arXiv preprint arXiv:2404.16349*, 2024.

[AS23] Josh Alman and Zhao Song. Fast attention requires bounded entries. *Advances in Neural Information Processing Systems*, 36, 2023.

[AS24] Josh Alman and Zhao Song. The fine-grained complexity of gradient computation for training large language models. *arXiv preprint arXiv:2402.04497*, 2024.

[BCS13] Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science & Business Media, 2013.

[BGOFG20] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[Blä13] Markus Bläser. Fast matrix multiplication. *Theory of Computing*, pages 1–60, 2013.

[BLSS20] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 775–788, 2020.

[BMBE20] Brian Bartoldson, Ari Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. *Advances in Neural Information Processing Systems*, 33:20852–20864, 2020.

[BMR+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[BPSW21] Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. Training (Over-parametrized) Neural Networks in Near-Linear Time. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 63:1–63:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Bra20] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 259–278. SIAM, 2020.

[BSZ24] Jan van den Brand, Zhao Song, and Tianyi Zhou. Algorithm and hardness for dynamic attention maintenance in large language models. In *Forty-first International Conference on Machine Learning*, 2024.

[CJD+21] Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. Only train once: A one-shot neural network training and pruning framework. *Advances in Neural Information Processing Systems*, 34:19637–19651, 2021.

[CLG+24] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

[CLL+24] Bo Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. Bypassing the exponential dependency: Looped transformers efficiently learn in-context by multi-step gradient descent. *arXiv preprint arXiv:2410.11268*, 2024.

[CLP+20] Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. Mongoose: A learnable lsh framework for efficient neural network training. In *International Conference on Learning Representations*, 2020.

[CLS19] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 938–942, 2019.

[CLS+24] Bo Chen, Yingyu Liang, Zhizhou Sha, Zhenmei Shi, and Zhao Song. Hsr-enhanced sparse attention acceleration. *arXiv preprint arXiv:2410.10165*, 2024.

[CTWC24] Ruisi Cai, Yuandong Tian, Zhangyang Wang, and Beidi Chen. Lococo: Dropping in convolutions for long context compression. *arXiv preprint arXiv:2406.05317*, 2024.

[Dao23] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[DFE+22] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[DG24] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

[DMS24] Majid Daliri, Christopher Musco, and Ananda Theertha Suresh. Coupling without communication and drafter-invariant speculative decoding. *arXiv preprint arXiv:2408.07978*, 2024.

[DSW22] Yichuan Deng, Zhao Song, and Omri Weinstein. Discrepancy minimization in input-sparsity time. *arXiv preprint arXiv:2210.12468*, 2022.

[DYZ+24] Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. Get more with less: Synthesizing recurrence with kv cache compression for efficient llm inference. *arXiv preprint arXiv:2402.09398*, 2024.

[ESL+24] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

[FA22] Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

[FA23] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR, 2023.

[FC18] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[GD23] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[GS22] Yuzhou Gu and Zhao Song. A faster small treewidth sdp solver. *arXiv preprint arXiv:2211.06033*, 2022.

[GSWY23] Yeqi Gao, Zhao Song, Weixin Wang, and Junze Yin. A fast optimization view: Reformulating single layer attention in llm based on tensor and svm trick, and solving it in matrix multiplication time. *arXiv preprint arXiv:2309.07418*, 2023.

[GSZ23] Yuzhou Gu, Zhao Song, and Lichen Zhang. Faster algorithms for structured linear and kernel support vector machines, 2023.

[GZL+23] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.

[HABN+21] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[HCI+21] Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in neural information processing systems*, 34:21099–21111, 2021.

[HCL+24] Jerry Yao-Chieh Hu, Pei-Hsuan Chang, Haozheng Luo, Hong-Yu Chen, Weijian Li, Wei-Po Wang, and Han Liu. Outlier-efficient hopfield layers for large transformer-based models. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[HCW+24] Jerry Yao-Chieh Hu, Bo-Yu Chen, Dennis Wu, Feng Ruan, and Han Liu. Nonparametric modern hopfield models. *arXiv preprint arXiv:2404.03900*, 2024.

[HJK+24] Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. In *The Twelfth International Conference on Learning Representations*, 2024.

[HJS+22] Baihe Huang, Shunhua Jiang, Zhao Song, Runzhou Tao, and Ruizhe Zhang. Solving sdp faster: A robust ipm framework and efficient implementation. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 233–244. IEEE, 2022.

[HLSL24] Jerry Yao-Chieh Hu, Thomas Lin, Zhao Song, and Han Liu. On computational limits of modern hopfield models: A fine-grained complexity analysis. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[HNH+21] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, pages 4466–4475. PMLR, 2021.

[HPTD15] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[HS92] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

[HWL24] Jerry Yao-Chieh Hu, Dennis Wu, and Han Liu. Provably optimal memory capacity for modern hopfield models: Tight analysis for transformer-compatible dense associative memories. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

[HWSL24] Jerry Yao-Chieh Hu, Weimin Wu, Zhao Song, and Han Liu. On statistical rates and provably efficient criteria of latent diffusion transformers (dits). *arXiv preprint arXiv:2407.01079*, 2024.

[HYW+23] Jerry Yao-Chieh Hu, Donglin Yang, Dennis Wu, Chenwei Xu, Bo-Yu Chen, and Han Liu. On sparse modern hopfield model. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[JCR+22] Tian Jin, Michael Carbin, Dan Roy, Jonathan Frankle, and Gintare Karolina Dziugaite. Pruning's effect on generalization through the lens of training and regularization. *Advances in Neural Information Processing Systems*, 35:37947–37961, 2022.

[JKL+20] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programming. In *2020 IEEE 61st annual symposium on foundations of computer science (FOCS)*, pages 910–918. IEEE, 2020.

[JLSW20] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 944–953, 2020.

[JSWZ21] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. A faster algorithm for solving general lps. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 823–832, New York, NY, USA, 2021. Association for Computing Machinery.

[KKM+22] Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35:24101–24116, 2022.

[KMZ23] Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. Polysketchformer: Fast transformers via sketches for polynomial kernels. *arXiv preprint arXiv:2310.01655*, 2023.

[LAT19] N Lee, T Ajanthan, and P Torr. Snip: single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*. Open Review, 2019.

[LCH+24] Minghan Li, Xilun Chen, Ari Holtzman, Beidi Chen, Jimmy Lin, Wen-tau Yih, and Xi Victoria Lin. Nearest neighbor speculative decoding for llm generation and attribution. *arXiv preprint arXiv:2405.19325*, 2024.

[LDLG23] Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*, 2023.

[LDS89] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[LG24] François Le Gall. Faster rectangular matrix multiplication by combination loss analysis. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3765–3791. SIAM, 2024.

[LGT+21] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.

[LGU18] François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1029–1046. SIAM, 2018.

[LLS+24a] Xiaoyu Li, Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Fine-grained attention i/o complexity: Comprehensive analysis for backward passes. *arXiv preprint arXiv:2410.09397*, 2024.

[LLS+24b] Yingyu Liang, Heshan Liu, Zhenmei Shi, Zhao Song, and Junze Yin. Conv-basis: A new paradigm for efficient attention inference and gradient computation in transformers. *arXiv preprint arXiv:2405.05219*, 2024.

[LLS+24c] Yingyu Liang, Jiangxuan Long, Zhenmei Shi, Zhao Song, and Yufa Zhou. Beyond linear approximations: A novel pruning approach for attention matrix. *arXiv preprint arXiv:2410.11261*, 2024.

[LM15] Shachar Lovett and Raghu Meka. Constructive discrepancy minimization by walking on the edges. *SIAM Journal on Computing*, 44(5):1573–1582, 2015.

[LSS+24] Yingyu Liang, Zhizhou Sha, Zhenmei Shi, Zhao Song, and Yufa Zhou. Multi-layer transformers gradient can be approximated in almost linear time. *arXiv preprint arXiv:2408.13233*, 2024.

[LSSY24] Yingyu Liang, Zhenmei Shi, Zhao Song, and Chiwun Yang. Toward infinite-long prefix in transformer. *arXiv preprint arXiv:2406.14036*, 2024.

[LSSZ24] Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Tensor attention training: Provably efficient learning of higher-order transformers. *arXiv preprint arXiv:2405.16411*, 2024.

[LSZ19] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *Conference on Learning Theory*, pages 2140–2157. PMLR, 2019.

[LSZ+23] S. Cliff Liu, Zhao Song, Hengjie Zhang, Lichen Zhang, and Tianyi Zhou. Space-Efficient Interior Point Method, with Applications to Linear Programming and Maximum Weight Bipartite Matching. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 88:1–88:14, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[LTT+24] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

[LWD+23] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR, 2023.

[LZ20] Tianlin Liu and Friedemann Zenke. Finding trainable sparse networks through neural tangent transfer. In *International Conference on Machine Learning*, pages 6336–6347. PMLR, 2020.

[MVK+24] Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing large language models. *arXiv preprint arXiv:2405.06640*, 2024.

[NAVB+20] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020.

[PMN+23] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023.

[QSZZ23] Lianke Qin, Zhao Song, Lichen Zhang, and Danyang Zhuo. An online and unified algorithm for projection matrix vector multiplication with application to empirical risk minimization. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 101–156. PMLR, 25–27 Apr 2023.

[SBZ+24] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608*, 2024.

[SCY+24] Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. *arXiv preprint arXiv:2404.11912*, 2024.

[SLBK24] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[SMN+24] Zhenmei Shi, Yifei Ming, Xuan-Phi Nguyen, Yingyu Liang, and Shafiq Joty. Discovering the gems in early layers: Accelerating long-context llms with 1000x input token reduction. *arXiv preprint arXiv:2409.17422*, 2024.

[SSU18] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

[SY21] Zhao Song and Zheng Yu. Oblivious sketching-based central path method for linear programming. In *International Conference on Machine Learning*, pages 9835–9847. PMLR, 2021.

[SY24] Barna Saha and Christopher Ye. The i/o complexity of attention, or how optimal is flash attention? *arXiv preprint arXiv:2402.07443*, 2024.

[SYZ23] Zhao Song, Mingquan Ye, and Lichen Zhang. Streaming semidefinite programs: $o(\sqrt{n})$ passes, small space and fast runtime, 2023.

[SZZ24] Zhao Song, Lichen Zhang, and Ruizhe Zhang. Training multi-layer over-parametrized neural network in subquadratic time. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024.

[TKYG20] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020.

[TZZ+24] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

[WHHL24] Dennis Wu, Jerry Yao-Chieh Hu, Teng-Yun Hsiao, and Han Liu. Uniform memory retrieval with larger capacity for modern hopfield models. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[WHL+24] Dennis Wu, Jerry Yao-Chieh Hu, Weijian Li, Bo-Yu Chen, and Han Liu. STanhop: Sparse tandem hopfield model for memory-enhanced time series prediction. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[WXXZ24] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3792–3835. SIAM, 2024.

[WZG19] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2019.

[XHH+24] Chenwei Xu, Yu-Chao Huang, Jerry Yao-Chieh Hu, Weijian Li, Ammar Gilani, Hsi-Sheng Goan, and Han Liu. Bishop: Bi-directional cellular learning for tabular data with generalized sparse modern hopfield model. In *Forty-first International Conference on Machine Learning (ICML)*, 2024.

[XTC+23] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[Yes84] Yaacov Yesha. Time-space tradeoffs for matrix multiplication and the discrete fourier transform on any general sequential random-access computer. *Journal of Computer and System Sciences*, 29(2):183–197, 1984.

[YLG+23] Hongru Yang, Yingbin Liang, Xiaojie Guo, Lingfei Wu, and Zhangyang Wang. Pruning before training may improve generalization, provably. *arXiv preprint arXiv:2301.00335*, 2023.

[ZBKR24] Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *arXiv preprint arXiv:2402.04347*, 2024.

[ZDH24] Amir Zandieh, Majid Daliri, and Insu Han. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead. *arXiv preprint arXiv:2406.03482*, 2024.

[ZHDK23] Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. Kdeformer: Accelerating transformers via kernel density estimation. In *International Conference on Machine Learning*, pages 40605–40623. PMLR, 2023.

[ZHMK24] Amir Zandieh, Insu Han, Vahab Mirrokni, and Amin Karbasi. Subgen: Token generation in sublinear time and memory. *arXiv preprint arXiv:2402.06082*, 2024.

[ZLC+24] Zhenyu Zhang, Shiwei Liu, Runjin Chen, Bhavya Kailkhura, Beidi Chen, and Atlas Wang. Q-hitter: A better token oracle for efficient llm inference via sparse-quantized kv cache. *Proceedings of Machine Learning and Systems*, 6:381–394, 2024.

[ZRG+22] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[ZSZ+24] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.