ICASSP 2026 - 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3315-6701-9/26/$31.00 ©2026 IEEE | DOI: 10.1109/ICASSP55912.2026.11463309

ATTENTION OUTPUT PROJECTION IMPORTANCE SCORE FOR KEY-VALUE EVICTION

Jian Yuan[1], *Ziwei He*[2], *Zhouhan Lin*[1], *Bo Jiang*[1*]

[1]Shanghai Jiao Tong University [2]Shanghai Innovation Institute
{yuanjian, bjiang}@sjtu.edu.cn ziweihe@outlook.com lin.zhouhan@gmail.com

ABSTRACT

Large language models (LLMs) store Key-Value (KV) Cache to avoid redundant computations and KV eviction methods are used to compress KV Cache to reduce its storage overhead. Existing KV eviction strategies focus on maintaining critical tokens relying on attention weights or their variants, yet this approach to gauge KV importance overlooks value states which also contribute to attention module outputs. To address this limitation this paper proposes the Attention Output Projection Score (AOPS), a novel token importance scoring approach for long sequences that integrates attention weights and value states while remaining compatible with existing KV eviction strategies. Experimental results on LongBench and Needle-in-a-Haystack demonstrate combining this method with conventional eviction schemes consistently enhances LLM performance.

***Index Terms*—** Large Language Models, Key-Value Cache Eviction, Token Importance Evaluation

1. INTRODUCTION

Large language models (LLMs) based on transformer framework have been widely applied to numerous tasks [1, 2, 3, 4]. Key-Value (KV) Caches are stored during the autoregressive inference procedure of LLMs to avoid repetitive operations. However, for lengthy sequences, the memory cost of KV cache becomes substantial, creating a major memory bottleneck [5]. To address this, KV cache eviction strategies have been designed to retain a small portion of critical KV pairs in the cache, significantly reducing storage and computational overhead while maintaining the model's performance.

Current KV eviction strategies generally focus on the attention weights of tokens when determining which KV pairs to retain or discard [5, 6, 7, 8, 9, 10]. These approaches operate under the implicit assumption that attention weights alone can sufficiently indicate the importance of tokens in maintaining model performance. However, in the attention modules, attention weights only determine how much each token's information contributes to the output, but value states are core components that get weighted and summed to generate the final attention output. Such strategies overlook the influential role of value states, which may lead to a loss of accuracy due to misjudging the importance of tokens.

We find that the value states corresponding to tokens with large attention weights do not necessarily make significant contributions to the attention output. Given that the attention output is derived from the weighted vector sum of value states based on attention weights, we employ the projection of a value state onto the attention output to quantify the contribution of that value state to the attention output. We observe that tokens with large attention weights are generally ineffective in covering value states that exhibit large projections onto the attention output. As an illustrative example, Figure 1 generated on the first sample of the TREC dataset [11] presents the connection between the relative cumulative attention weights of the final token and the relative cumulative value projections of the corresponding tokens. Here the tokens corresponding to the horizontal axis are sorted in descending order based on their attention weights and cumulation is applied along the dimension of the sorted sequence. Note that we exclude sink tokens, which are known for large attention weight but no semantic information [12]. The figure shows that even when selecting tokens that account for a large proportion of attention such as 80% the relative cumulative value projection of these tokens remains below 40%. Based on this phenomenon we conjecture that a KV selection method that jointly considers both attention weights and value states will yield superior performance compared to strategies that rely purely on attention weights.

In this paper, we propose attention output projection score (AOPS), a novel token importance scoring approach for long input sequences that considers not only attention weights but also magnitudes and directions of value states. First, this method calculates the attention weights for each token in the input sequence and uses these scores to compute the attention output of recent tokens. Subsequently, the product of value states and attention weights is projected onto the direction of the average attention output. Finally, the KV pairs corresponding to the tokens with the largest projection components are retained in the cache. Experimentally, we replace the original attention-only KV selection mechanisms of H2O [7], SnapKV [8], and FastKV [10] with our proposed method. Evaluations conducted on the LongBench [11] and Needle-in-a-Haystack (NIAH) [13] benchmarks demonstrate consistent

* Bo Jiang is the corresponding author.

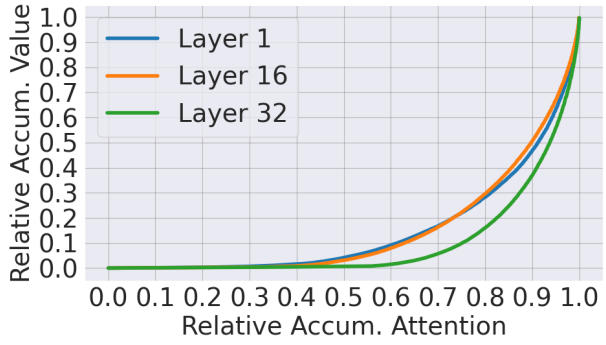


Fig. 1. The relation between the relative cumulative attention weights and the relative cumulative value projections based on the first sample of TREC using Llama3.1-8B-Instruct.

performance improvements on the given model, validating the advantages of our proposed approach. The contributions of this study can be outlined in three aspects:

- We propose a novel token importance scoring approach, AOPS for long input sequences, which integrates attention weights and the projection of value states onto attention outputs.
- We discover that tokens with large attention weights may have value states with small projections onto attention outputs, indicating current KV eviction methods relying solely on attention weights may not accurately assess a token's importance to attention output.
- We validate the effectiveness of AOPS through extensive experiments, demonstrating that its combination with existing KV cache eviction strategies yields stable improvements in overall performance.

2. RELATED WORKS

At the outset, StreamingLLM[14] and LM-Infinite[6] statically retain the KV pairs of initial tokens and recent tokens with relatively high attention weights in cache. Subsequent methods dynamically configure different sets of tokens to be retained for different inputs based on these weights. H2O[7] and Scissorhands[5] retain the KV pairs corresponding to the maximum attention weights and binarized attention weights in the cache, respectively. SnapKV[8] and PyramidKV[9] also utilize these weights as the criteria for evaluating the importance of KV pairs, with PyramidKV additionally featuring a layer-dependent cache allocation strategy where assigned KV cache sizes decrease as layer depth increases. Prior to evicting tokens, both methods apply a pooling operation to the importance scores, ensuring that the KV pairs retained in the cache maintain contextual continuity. By replacing the method for calculating token importance scores with our proposed one, our method can reduce the accuracy loss caused by compression. FastKV [10] incorporates hidden states compression in the middle layer on the basis of selecting KVs using pooled attention weights and the importance scores it uses for hidden states compression are token-wise. Surprisingly, applying our method to the selection of such inter-layer hidden states also leads to an improvement in accuracy.

Algorithm 1 AOPS

Input: $\tilde{\mathbf{Q}}, \mathbf{K}, \mathbf{V}$.
Output: $\mathbf{S}$.

Get attention weights for the recent window:
$\tilde{\mathbf{A}} = \text{Softmax}\left(\tilde{\mathbf{Q}} \cdot \mathbf{K}^\top / \sqrt{d_h}\right)$.
Get attention outputs for the recent window:
$\tilde{\mathbf{O}} = \tilde{\mathbf{A}} \cdot \mathbf{V}$.
Average attention outputs along the query dimension:
$\overline{\mathbf{O}} = \text{Mean}\left(\tilde{\mathbf{O}}, \text{dim} = 1\right)$.
Get importance scores of tokens by projection:
$\mathbf{P} = \text{Mean}\left(\tilde{\mathbf{A}}, \text{dim} = 1\right)^\top \odot \mathbf{V}, \mathbf{S} = \mathbf{P} \cdot \overline{\mathbf{O}}^\top$.
return $\mathbf{S}$.

3. BACKGROUND

In LLMs, the attention module within each layer transforms the hidden states of an input sequence into attention outputs through a structured process, which is fundamental to capturing contextual dependencies between tokens. First, the attention module takes the input hidden states $\mathbf{H} \in \mathbb{R}^{n \times d}$ where n and d denote the sequence length and model dimension respectively. Then the module generates three distinct vector representations *query*, *key* and *value* states for each head via linear mapping

$$\mathbf{Q} = \mathbf{H} \cdot \mathbf{W}_Q,\ \mathbf{K} = \mathbf{H} \cdot \mathbf{W}_K,\ \mathbf{V} = \mathbf{H} \cdot \mathbf{W}_V,$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_h}$. d_h is the head dimension of the model. Next, the module computes attention weights to quantify the relevance between each pair of tokens by

$$\mathbf{A} = \text{Softmax}\left(\frac{\mathbf{Q} \cdot \mathbf{K}^\top}{\sqrt{d_h}}\right).$$

The softmax function is applied to the scaled scores along the sequence dimension of $\mathbf{K}$, normalizing the scores such that they sum to 1 for each query state. Finally, the attention output $\mathbf{O} \in \mathbb{R}^{d_h}$ is obtained by a weighted sum of the value states using the attention weights $\mathbf{O} = \mathbf{A} \cdot \mathbf{V}$. The attention output of the module is assembled by concatenating the outputs of all heads along the head dimension. In this computation process, key states and value states are stored in the KV cache to avoid redundant computations in subsequent generation.

Table 1. Evaluation on Longbench using Mistral-7B-Instruct-v0.3 and Llama-3.1-8B-Instruct with cache size 512. The KV cache preserved by our method outperforms the vanilla KV cache eviction methods in most tasks under each setting and consistently achieves better overall scores than the baseline across all settings.

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PRe	Lcc	RB-P	
Mistral-7B-Instruct-v0.3																	
H2O	24.36	29.58	43.49	42.82	30.86	22.44	25.38	23.89	24.46	42.00	**89.18**	45.69	**5.50**	**96.50**	60.06	59.76	41.62
+AOPS	**25.97**	**31.97**	**44.42**	**43.95**	**31.65**	**22.95**	**25.89**	**24.05**	**24.57**	**43.00**	88.96	**45.99**	5.00	**96.50**	**60.20**	**60.35**	**42.21**
SnapKV	25.61	32.92	**49.09**	48.37	32.46	24.35	25.97	**24.36**	24.39	71.00	**89.61**	44.56	**5.00**	**96.00**	60.83	61.87	44.77
+AOPS	**25.80**	**33.14**	48.88	**48.97**	**33.13**	**24.62**	**26.25**	24.24	**24.41**	**72.00**	89.29	**44.76**	**5.00**	**96.00**	**60.99**	**61.93**	**44.96**
FastKV	24.96	32.55	49.64	48.63	31.87	24.73	**25.57**	**23.97**	23.94	68.50	89.44	**44.80**	4.50	95.00	**60.23**	61.36	44.36
+AOPS	**24.98**	**33.06**	**49.93**	**49.39**	**32.95**	**25.96**	25.51	23.66	**24.05**	**70.50**	**89.50**	44.75	**5.00**	**96.00**	60.21	**61.59**	**44.82**
Llama-3.1-8B-Instruct																	
H2O	27.06	32.07	47.64	**54.20**	45.52	**29.54**	26.47	23.77	**24.35**	44.00	**91.69**	41.51	**7.29**	**99.50**	**63.07**	**55.70**	44.59
+AOPS	**28.42**	**33.90**	**50.28**	53.84	**46.48**	29.24	**26.66**	**23.85**	**24.35**	**44.50**	**91.69**	**41.64**	**7.29**	**99.50**	62.75	55.55	**45.00**
SnapKV	**29.05**	40.32	**52.97**	55.57	44.83	29.70	25.88	24.22	24.44	**67.50**	91.69	42.28	**8.20**	**99.50**	64.13	**57.39**	47.35
+AOPS	28.38	**40.33**	52.86	**56.01**	**45.02**	**31.10**	**26.43**	**24.37**	**24.60**	**67.50**	**91.94**	**42.72**	8.16	**99.50**	**64.48**	57.13	**47.53**
FastKV	28.96	**38.86**	**53.82**	54.45	**46.72**	29.74	25.43	23.96	**24.25**	65.50	**92.35**	41.74	7.64	**99.50**	63.87	57.02	47.11
+AOPS	**29.33**	38.85	53.51	**55.44**	46.65	**30.72**	**25.52**	**24.13**	24.01	**66.00**	92.04	**42.18**	**8.51**	**99.50**	**64.22**	**57.29**	**47.37**

4. METHODOLOGY

In this section, we will elaborate on how our proposed method AOPS calculates importance scores for tokens in long input sequences based on the information from attention weights and value states, as well as how it integrates with the existing KV eviction strategies. Algorithm 1 presents the specific process of implementing our method.

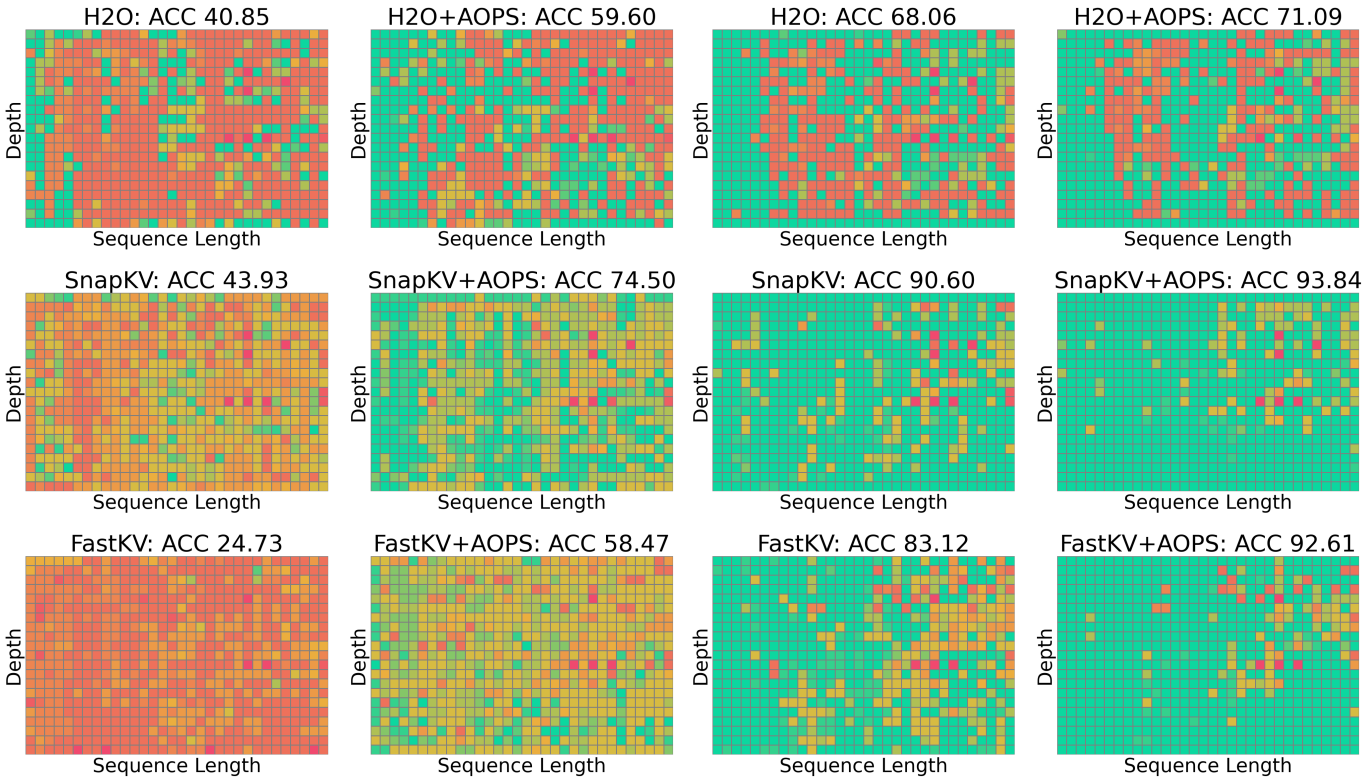
First, our algorithm calculates the attention weight for each token based on the query states $\mathbf{Q}$ and key states $\mathbf{K}$ of the current layer. For efficient computation, we use the query states within a recent window $\tilde{\mathbf{Q}}$ and key states to perform matrix multiplication and apply the Softmax function along the dimension of key states as suggested by SnapKV[8]. Subsequently, the obtained attention weights are used to weight and sum the value states, which yields the attention outputs for the tokens in the recent window $\tilde{\mathbf{O}}$. We compute the attention-value product $\mathbf{P}$ by taking the dot product of the attention weights and value states, which represents the vector component of each token's contribution to the attention output. Finally, this product is projected onto the average vector of the attention outputs and the magnitude of the projected vector $\mathbf{S}$ is used as the score to evaluate token importance. Such a method of estimating token importance via projection size incorporates both the magnitude and direction of value states alongside attention weights, while evaluating each token's representational contribution to the attention output. For methods that dynamically maintain KV pairs corresponding to large attention weights like H2O and SnapKV, our method can seamlessly integrate with them by replacing the method for calculating token importance scores with AOPS. For inter-layer hidden state eviction such as FastKV, importance scores of hidden states can be derived by averaging AOPS across the head dimension.

5. EXPERIMENTS

We apply AOPS to three KV eviction methods H2O, SnapKV and FastKV. We compared the performance of the vanilla versions of these methods with the versions integrated with AOPS. For FastKV, the hidden states were reduced to 2048 during inter-layer compression as recommended in its original paper. To assess the performance of different methods, we conduct experiments on 2 benchmarks LongBench and Needle-in-a-Haystack (NIAH) using models Mistral-7B-Instruct-v0.3[15] and Llama-3.1-8B-Instruct[16].

5.1. LongBench

LongBench acts as a thorough evaluation benchmark made to check the long-text comprehension skills of large language models. In this study, we conducted LongBench evaluations for various tasks with a KV cache size of 512. We report scores for sixteen tasks categorized into six types single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion. As shown in the Table 1 the KV cache preserved by our method outperforms the vanilla KV cache eviction methods in most



(a) Cache Size = 32

(b) Cache Size = 64

Fig. 2. We conduct NIAH evaluations for Mistral-7B-Instruct-v0.3 with KV cache sizes of 64 and 32. The horizontal axis represents sequence length with values ranging from 1000 to 32000 from left to right. The vertical axis represents 20 different needle placement depths. Red-to-green colors represent accuracy from 0 to 100.

tasks under each setting and consistently achieves better results than the baseline across all settings. Specifically AOPS yields an average score improvement of 0.59 and 0.41 in scenarios using the Mistral-7B-Instruct-v0.3 and Llama-3.1-8B-Instruct models respectively. This indicates that when AOPS is applied to the evicting of KV and hidden states, it achieves a general improvement compared with the original method that only uses attention weights.

5.2. NIAH

The Needle in a Haystack (NIAH) task assesses LLMs by inserting a particular key information piece into a large text collection and guiding the models to retrieve this information. In our research, we carried out the Needle in a Haystack task under the traditional experimental setting as stated in [13]. Figure 2 shows heatmaps of NIAH evaluations accuracy using Mistral-7B-Instruct-v0.3 with KV cache sizes of 64 and 32. The horizontal axis represents sequence length with values ranging from 1000 to 32000 from left to right. The vertical axis represents 20 different needle placement depths with values ranging from 0% to 100% from top to bottom. The left and right columns in Figures 2(a) and 2(b) represent the KV Cache Eviction method without and with AOPS respectively. The AOPS-enabled version achieves higher accuracy than the original method under two different cache sizes. The AOPS can bring accuracy improvements of 33.74 and 9.49 under the cache sizes of 32 and 64 respectively. The improvement effect of our method becomes increasingly significant as the KV cache size decreases.

6. CONCLUSION

This paper proposes attention output projection score (AOPS), a novel token importance score that can be integrated into existing KV eviction methods. AOPS calculates each token's attention weight from key states and recent query states then uses these scores to obtain attention output. It computes the attention-value product and projects this product onto the average attention output vector with the projected magnitude as the token importance score. Experimental results on LongBench and Needle-in-a-Haystack demonstrate that combining this scoring method with conventional compression schemes consistently enhances the overall performance of LLMs.

7. REFERENCES

[1] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito, "Wordcraft: story writing with large language models," in *Proceedings of the 27th International Conference on Intelligent User Interfaces*, 2022, pp. 841–852.

[2] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al., "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.

[3] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto, "Benchmarking large language models for news summarization," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 39–57, 2024.

[4] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al., "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, pp. 186345, 2024.

[5] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava, "Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[6] Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang, "Lm-infinite: Zero-shot extreme length generalization for large language models," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 3991–4008.

[7] 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*, vol. 36, 2024.

[8] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen, "Snapkv: Llm knows what you are looking for before generation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 22947–22970, 2024.

[9] Yichi Zhang, Bofei Gao, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, Wen Xiao, et al., "Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling," *arXiv preprint arXiv:2406.02069*, 2024.

[10] Dongwon Jo, Jiwon Song, Yulhwa Kim, and Jae-Joon Kim, "Fastkv: Kv cache compression for fast long-context processing with token-selective propagation," *arXiv preprint arXiv:2502.01068*, 2025.

[11] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al., "Longbench: A bilingual, multitask benchmark for long context understanding," *arXiv preprint arXiv:2308.14508*, 2023.

[12] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis, "Efficient streaming language models with attention sinks," in *The Twelfth International Conference on Learning Representations*, 2024.

[13] Gregory Kamradt, "Needle in a haystack - pressure testing llms," 2023.

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

[15] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed, "Mistral 7b," 2023.

[16] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al., "The llama 3 herd of models," *arXiv e-prints*, pp. arXiv–2407, 2024.