High Attention Scores And Where To Find Them

Nicolas Andrin Menet Jacky Choi
Chandra de Viragh Leila Chettata

1. Introduction and Related Work

The Transformer architecture introduced by Vaswani et al. 2017 revolutionised a myriad of machine learning subfields. Despite being more parallelisable than their predecessors (RNNs), Transformers have been scaled to the point of incurring incredibly high compute and memory costs, especially for natural language processing (Brown et al., 2020; Touvron et al., 2023). A particular challenge for very long sequences is the quadratic cost in the number of processed tokens of self-attention, constituting a major bottleneck. A celebrated line of work addressing the quadratic scaling are Linear Transformers. In Choromanski et al. 2022 the softmax kernel is linearised through an explicit feature transformation $\phi : \mathbb{R}^D \to \mathbb{R}^R$ of keys and queries. Attending to other tokens no longer costs $\mathcal{O}(LD)$, where L is the maximal sequence length and D the dimensionality of key/query/value projections, but rather $\mathcal{O}(DR)$ where R is the dimensionality of the kernel feature transformation. Despite the kernel estimation of softmax in Choromanski et al. 2022 being unbiased, in practice Linear Transformers have been rarely used due to the high variance for large attention scores, see also our analysis in Section 4.4. Our preliminary study of attention scores[1] in a GPT-2 model showed that in autoregressive text generation, only a small fraction of them is of significant magnitude, especially after the first layer (see also Zhou et al. 2021). Consequently, we strive to locate these terms without computing the full attention matrix. Our repository can be accessed on: `https://gitlab.ethz.ch/devirac/casinoformer-deep-learning-project-hs23`

Our contributions are as follows: 1) We design an algorithm for computing attention scores that requires equally many inner products as standard attention, while being much more informative along the way. In particular, we obtain the geometric mean of contiguous blocks of attention scores at intermediate steps. 2) We experiment with six different methods that determine the order in which attention is computed and evaluate their induced perplexity on WikiText-2 (Merity et al., 2016) using the GPT-2 architecture (Radford et al., 2019). 3) We analyse the theoretical complexity of our algorithm and give theoretical explanations to the observed performance of each (sub-)method.

[1] see the README.md of our repository

2. Methods

To ease the exposition, we will omit the rescaling by $1/\sqrt{D}$ of query-key inner products that is standard in Transformers.

2.1. Tree Hierarchy

Similarly to Zhu & Soricut 2021, we consider a tree of key and value superpositions, arranged into blocks (see Figure 1a) and defined using the recursive relations

$$K_j^h = K_{2j}^{h-1} + K_{2j+1}^{h-1} \quad \text{and} \quad V_j^h = V_{2j}^{h-1} + V_{2j+1}^{h-1}, \tag{1}$$

where h denotes height and j the horizontal position. Due to the inner product being linear, the query-key inner products of a block, henceforth called alignments, are given by

$$A_j^h = A_{2j}^{h-1} + A_{2j+1}^{h-1} \iff A_{2j}^{h-1} = A_j^h - A_{2j+1}^{h-1}. \tag{2}$$

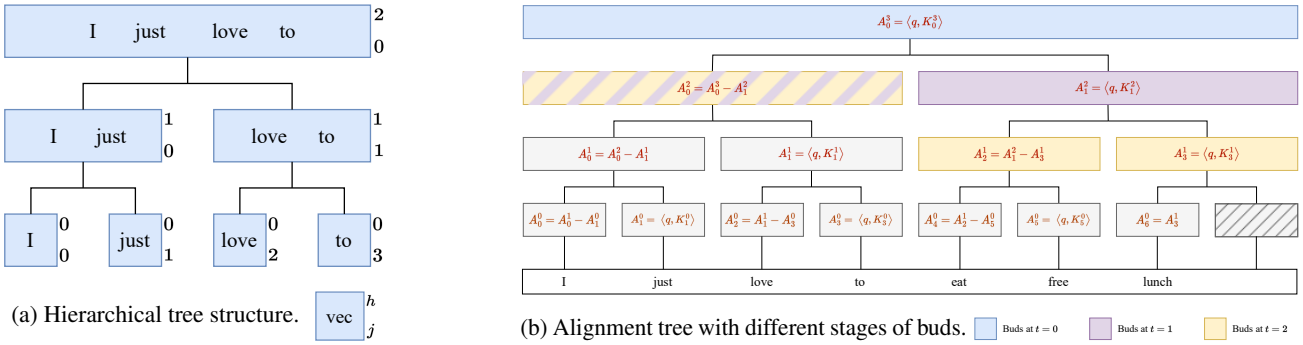
Thus, computing alignments in a perfect tree starting from the root, with $2L-1$ nodes and L leaves, only requires L inner products, just like if we only evaluated the leaves as in standard attention. Figure 1b illustrates the procedure.

2.2. Core Algorithm for Partial Tree Evaluation

With the cost of full tree evaluation coinciding with standard attention, we propose to instead evaluate the tree only partially. At each stage of the following algorithm, we keep track of a set of "buds", nodes that may be split further, but which have a disjoint set of leaves, see Figure 1b. The final set of buds is then used to construct a T-term approximation to attention, see Algorithm 1. Each term takes care of a set of consecutive leaves which share a single attention score given precisely by the geometric mean of the true individual scores. Ideally, the number of terms T is much less than the sequence length. However, to that end the probability mass according to which buds are expanded must reliably indicate the presence of high attention scores.

Algorithm 1 Bud Expansion

determine initial set of buds B (e.g. only root node)
while $|B| < T$
sample $n \in B$ according to its "probability mass"
compute alignment of its children n_1 and n_2
set $B \leftarrow (B \setminus \{n\}) \cup \{n_1, n_2\}$
end
output $\sum_{n\in B} \frac{\exp\left(\sum_{k\in n}\langle q,k\rangle/|n|\right)V_n}{\sum_{m\in B}\exp\left(\sum_{k\in m}\langle q,k\rangle/|n|\right)}$
where $k \in n$ iterates over the $|n|$ leaf-keys covered by bud n.



(a) Hierarchical tree structure.

(b) Alignment tree with different stages of buds.

Figure 1. Our hierarchical tree structure allows computation of attention with sums of alignments as intermediate results.

Deviation from Proposal: As discovered through analysis of linear attention[2] and as elaborated in Section 4.4, Linear Transformers estimate attention extremely unreliably. Hence, we decided against the proposed plan of replacing (the useless) attention scores from linear transformers with recomputed accurate ones in order to avoid the overhead of additionally computing approximate attention at the leaf nodes to obtain the correction delta.

2.3. Associating Buds with Probability Mass

In Algorithm 1, buds are expanded according to an associated probability mass. We wish to expand buds whose sub-tree leaves have large attention scores. To that end, we devise several methods and evaluate them in Section 3. Four of our sampling methods use a feature transformation of keys and queries to obtain sampling probabilities. There, in addition to the usual alignments, which we still require in order to construct the output, we compute transformed alignments $\tilde{A}_n$ for the buds, based on the same hierarchical scheme as in Figure 1b, but with transformed keys and queries (in $\mathbb{R}^R$). $\tilde{A}_n$ are then used to derive the probability mass according to which buds are expanded.

Exponentially Decaying Horizon (EDH)
Given a query at position l, we can associate past tokens at position t with an unnormalised decaying probability mass b^{l-t-1} for $b \in (0,1)$, we use $b = 0.99$. We can efficiently compute the aggregate unnormalised probability mass for a bud covering tokens at leaf node positions $\{s, \dots, t\}$ by

$$\sum_{k=l-s-1}^{l-t-1} b^k = \frac{b^{l-s-1} - b^{l-t}}{1-b}. \tag{3}$$

Alignments (Align.)
We may normalise the geometric mean of attention scores $\exp\left(\sum_{k\in n}\langle q,k\rangle/|n|\right) = \exp(A_n/|n|)$, for a bud n with alignment A_n, to a probability mass over all current buds. This biases sampling toward regions of high attention scores.

Positive Alignments (Pos. Align.)

[2] see the kernel.py file in our repository

According to the results in Section 4.2, negative query-key inner-products dominate aggregate alignments of a block despite being irrelevant to attention. To ensure positive alignments, we can make use of a feature transformation:

$$\phi(x) = \begin{bmatrix} (x)_+ \\ (-x)_+ \end{bmatrix}, \quad \langle \phi(q), \phi(k) \rangle = \sum_{i=1}^{D} (q_i \cdot k_i)_+, \tag{4}$$

where $(x)_+ = \max(x, 0)$. With $\tilde{A}_n = \sum_{k\in n}\langle \phi(q), \phi(k)\rangle$, the associated probability mass becomes $\exp\left(\tilde{A}_n/|n|\right)$.

Random Fourier Features (RFF)
Introduced by Rahimi & Recht 2007 to approximate the Gaussian kernel, Choromanski et al. 2022 showed how to adapt Random Fourier Features to an unbiased estimation of the softmax kernel. We transform keys and queries as

$$\phi(x) = \frac{e^{\frac{\|x\|_2^2}{2}}}{\sqrt{R/2}} \left[\sin w_1^T x \;\; \cos w_1^T x \;\; \dots \;\; \cos w_{R/2}^T x\right]^T \tag{5}$$

where $(w_1, \dots, w_{R/2})$ is a uniform sample from the Stiefel manifold with entries rescaled by samples from the χ-distribution to ensure a standard normal marginal. Since $\mathbb{E}[\langle \phi(q), \phi(k)\rangle] = \exp(\langle q, k\rangle)$, the associated probability mass for bud n is set to $(\tilde{A}_n)_+ = (\sum_{k\in n}\langle \phi(q), \phi(k)\rangle)_+$.

FAVOR+
Choromanski et al. 2022 proposed an alternative unbiased method of approximating the softmax kernel for attention using random positive orthogonal hyperbolic features, i.e.

$$\phi(x) = \frac{e^{\frac{-\|x\|_2^2}{2}}}{\sqrt{R}} \left[e^{w_1^T x} \;\; e^{-w_1^T x} \;\; \dots \;\; e^{-w_{R/2}^T x}\right]^T, \tag{6}$$

where $w_1, \dots, w_{R/2}$ is distributed as above. Since it still holds $\mathbb{E}[\langle \phi(q), \phi(k)\rangle] = \exp(\langle q, k\rangle)$, but now $\tilde{A}_n = \sum_{k\in n}\langle \phi(q), \phi(k)\rangle \geq 0$, we sample right according to $\tilde{A}_n$.

FAVOR+ReLU
As will be discussed meticulously in Section 4.4, the standard deviation of FAVOR+ exceeds its mean $\exp(\langle q, k\rangle)$ by orders of magnitude for reasonable R, unless keys and

queries are heavily downscaled. Choromanski et al. 2022 also propose a biased kernel based on ReLU, given by

$$\phi(x) = \frac{1}{\sqrt{R}} \left[(w_1^T x)_+ \ (-w_1^T x)_+ \ \dots \ (-w_{R/2}^T x)_+ \right]^T \quad (7)$$

where $(w_1, \dots, w_{R/2})$ is a uniform sample from the Stiefel manifold with entries rescaled by $\sqrt{D}$. As the analysis in Appendix C.2 of Menet et al. 2023 shows,

$$\begin{aligned} \phi(q)^T \phi(k) &\approx \mathbb{E}_{w \sim \mathcal{U}(\sqrt{D}\mathcal{S}^{D-1})} [(w^T q)_+ \cdot (w^T k)_+] \\ &= \|q\| \|k\| \frac{\rho + g(\rho)}{4} \approx \|q\| \|k\| \frac{(\rho+1)^{\log_2(\pi)}}{2\pi} \end{aligned} \quad (8)$$

for $g(\rho) = \frac{2}{\pi}[\sqrt{1-\rho^2} + |\rho| \arctan(\frac{|\rho|}{\sqrt{1-\rho^2}})]$ with cosine similarity $\rho = \langle q, k \rangle / (\|q\| \|k\|)$. Contrary to softmax attention, the estimated kernel is bilinear in the norms of keys and queries and monomial in the shifted cosine distance.

2.4. Practical Efficiency Considerations

As Hua et al. 2022 noted, the memory access pattern of Linear Transformers for non-autoregressive tasks leads to a serious runtime overhead. Despite our work focusing on autoregressive inference, the evaluation of attention along a hierarchical tree also leads to irregular memory-accesses, additional sampling operations, and bookkeeping for control. We soften these impracticalities in two ways:

1. Instead of sampling and expanding one bud at a time we sample without replacement P buds and compute their associated alignments concurrently.
2. We share one hierarchical tree among all attention heads, meaning once a bud is sampled and expanded, alignments for all heads are computed. The unnormalized probability mass of a bud is naturally given by the sum of probability masses across all heads.

Expanding multiple buds at once leads to operations on non-contiguous tensors, which PyTorch and NumPy solve by operating on contiguous copies. To avoid this, a custom kernel code would be necessary, which can achieve notable benefits even for standard attention, see Dao et al. 2022.

2.5. Theoretical Complexity Analysis

In order to sample according to the evolving distribution over buds, we can make use of a binary sum heap[3]. Sampling from the heap and updating it both costs $\mathcal{O}(\log T)$. Accordingly, the complexity of Algorithm 1 is given by $\mathcal{O}(T \cdot (D + R) + T \log T)$ compared to the complexity of autoregressive attention $\mathcal{O}(l \cdot D)$ where $l \geq T$. Both standard attention and our method share the same constant coefficient[4]. As discussed in Section 2.4, in practice, we amortize the cost of tree traversal across H heads. Thus, we obtain a complexity of $\mathcal{O}(T \cdot (D + R) + \frac{T \log T}{H})$ per head, which for moderate H completely removes the overhead for sampling. R is zero unless feature maps are used.

[3] see Vieira 2016

[4] namely 2, which stems from the computation of the query-key alignment and the weighted superposition of value vectors.

3. Results

Algorithm 1 was incorporated into the GPT-2 architecture (Radford et al., 2019), specifically, the nanoGPT[5] implementation with 124M parameters. We benchmarked it on 40% of the WikiText-2 test set (Merity et al., 2016) without retraining, and we recorded perplexity scores (Jelinek et al., 1977), an appropriate performance metric for LLMs. In Figure 2 the methods described in Section 2.3 are compared. The table header measures the number of buds expanded, where given a history of l tokens, a $T = \lceil l^E \rceil$-term approximation is performed. As an unselective baseline, we also evaluate uniform sampling from all buds B in Algorithm 1. In all experiments, we use $P = 2^{\lfloor \log_2 L^{E/2} \rfloor}$ to increase efficiency. For exponential decay we use $b = 0.99$, and for feature map methods we use $R = 2D = 128$ features.

L^E (E)	4 (0.2)	8 (0.3)	16 (0.4)	32 (0.5)
Uniform	$36.4^{\pm 3.8}$	$33.6^{\pm 3.5}$	$26.8^{\pm 0.2}$	$31.4^{\pm 0.4}$
EDH	$\mathbf{28.2}^{\pm 2.2}$	$\mathbf{24.7}^{\pm 1.7}$	$\mathbf{16.1}^{\pm 0.3}$	$\mathbf{13.7}^{\pm 0.5}$
Align.	$\underline{38.6}^{\pm 4.4}$	$\underline{34.3}^{\pm 0.1}$	$\underline{29.5}^{\pm 0.7}$	$30.8^{\pm 0.3}$
Pos. Align.	$35.7^{\pm 3.8}$	$30.0^{\pm 3.0}$	$27.1^{\pm 0.0}$	$23.6^{\pm 1.4}$
RFF	$35.1^{\pm 3.2}$	$27.2^{\pm 1.5}$	$23.5^{\pm 0.3}$	$19.9^{\pm 1.0}$
FAVOR+	$37.8^{\pm 4.3}$	$33.4^{\pm 3.8}$	$27.4^{\pm 0.4}$	$\underline{35.4}^{\pm 1.5}$
FAVOR+ReLU	$36.1^{\pm 3.8}$	$33.0^{\pm 3.3}$	$26.7^{\pm 0.9}$	$29.2^{\pm 0.2}$
L^E (E)	64 (0.6)	128 (0.7)	256 (0.8)	512 (0.9)
Uniform	$33.8^{\pm 4.9}$	$37.9^{\pm 0.8}$	$24.6^{\pm 1.3}$	$20.4^{\pm 3.0}$
EDH	$20.4^{\pm 0.1}$	$31.3^{\pm 0.3}$	$22.0^{\pm 1.9}$	$15.4^{\pm 0.1}$
Align.	$30.6^{\pm 3.3}$	$\underline{98.1}^{\pm 3.4}$	$97.8^{\pm 1.1}$	$16.0^{\pm 0.3}$
Pos. Align.	$\underline{39.8}^{\pm 0.8}$	$38.9^{\pm 3.2}$	$57.8^{\pm 6.8}$	$11.0^{\pm 0.9}$
RFF	$\mathbf{19.3}^{\pm 0.2}$	$\mathbf{15.1}^{\pm 0.2}$	$19.8^{\pm 1.3}$	$18.0^{\pm 1.1}$
FAVOR+	$38.0^{\pm 4.3}$	$77.6^{\pm 6.0}$	$\underline{99.9}^{\pm 0.7}$	$\underline{60.0}^{\pm 2.8}$
FAVOR+ReLU	$30.2^{\pm 3.6}$	$26.9^{\pm 1.2}$	$\mathbf{15.8}^{\pm 0.2}$	$\mathbf{10.3}^{\pm 1.1}$

Figure 2. Perplexity scores for different E's. The table displays the mean $\pm$ the unbiased standard deviation estimated from two runs, each employing distinct 20% portions of the WikiText-2 testset. As a comparison, original attention (i.e. $E = 1.0$) yields $11.2^{\pm 0.3}$.

4. Discussion

Although our proposed algorithm has very favourable theoretical properties (small worst-case computational overhead, geometric means as useful intermediate results, flexibility of sampling method), our empirical results show the difficulty of finding an easily obtainable yet informative probability mass that allows to only evaluate a small fraction of the tree. Depending on the number of terms ($T = \lceil l^E \rceil$) in the approximation of attention, different methods work best.

4.1. The Overengineering of EDH

Exponential decay shows strong empirical performance for few-term approximations $E \leq 0.5$, but it ignores attention

[5] see https://github.com/karpathy/nanoGPT

scores, making it less competitive for $E \geq 0.5$. Also, a similar effect can be achieved using local sliding windows while avoiding the overhead of a tree-structure altogether.

4.2. The Unfortunate Distribution of Alignments

As Figure 3 demonstrates, in practice, query-key alignments are frequently dominated by negative values, which drown the relevant positive alignments in noise. This issue is most detrimental for more selective and fine-grained many-term estimations ($E = 0.7, E = 0.8$), see Figure 2.

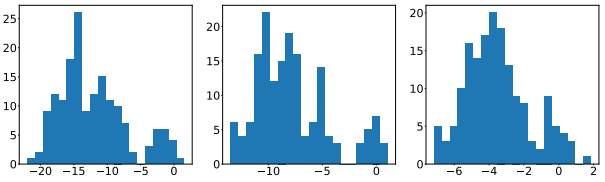


Figure 3. Histogram of query(".")-key alignments (*x-axis*) for three heads at layer six of a GPT-2 model (124M parameters).

4.3. Drowning in Positivity

With the positive alignment feature map we strive to filter out the aforementioned negative alignments by only considering positive contributions to the inner product (see Equation 4). Because even the transformed (positive) alignments of low-attention keys are of significant magnitude[6], the relevant high-attention keys are still hidden in noise.

4.4. Debunking FAVOR+

The variance of hyperbolic positive features[7] scales as

$$\frac{\mathrm{Var}[\widehat{SM}_R^{hyp+}(q,k)]}{\mathbb{E}[\widehat{SM}_R^{hyp+}(q,k)]^2} = \frac{2}{R}\left(\cosh\left(\frac{\|q+k\|^2}{\sqrt{D}}\right) - 1\right) \tag{9}$$

with R indicating the dimension of the kernel approximation. Unfortunately, $\cosh(\cdot)$ grows exponentially fast. As a result, a reliable use of FAVOR+ requires heavy downscaling of keys and queries, which leads to highly uniform attention scores, significantly damaging the discriminative nature of softmax. Indeed, defining the dynamic range of softmax for keys and queries with angle $\measuredangle q, k \in [\pi/2 - \theta, \pi/2 + \theta]$ for $\theta > 0$ and norm $0 \leq \|q\|, \|k\| \leq N$, i.e. $(q, k) \in V_\theta^N$, as

$$\mathrm{DR}_{\theta,N} = \frac{\max_{(q,k)\in V_{\theta,N}} \exp\left(\langle q,k\rangle/\sqrt{D}\right)}{\min_{(q,k)\in V_{\theta,N}} \exp\left(\langle q,k\rangle/\sqrt{D}\right)} = e^{\frac{2\sin(\theta)N^2}{\sqrt{D}}}, \tag{10}$$

it holds that

$$\max_{(q,k)\in V_{\theta,N}} e^{\frac{\|q+k\|^2}{\sqrt{D}}} = e^{\frac{2N^2}{\sqrt{D}}} \cdot \mathrm{DR}_{\theta,N} = \mathrm{DR}_{\theta,N}^{\frac{1}{\sin(\theta)}+1} \tag{11}$$

[6] see the kernel.py file in our repository

[7] See Equation 8 in Choromanski et al. 2022 with $\exp\left(\|q+k\|^2\right)\left(1-\exp\left(-\|q+k\|^2\right)\right)^2 = 2\cosh\left(\|q+k\|^2\right) - 2$.

As per Equation 9, $R \gg \exp\left(\frac{\|q+k\|^2}{\sqrt{D}}\right) = \mathrm{DR}_{\theta,N}^{\frac{1}{\sin(\theta)}+1}$ must hold. Since according to the Curse of Dimensionality in high dimensions almost all vectors are orthogonal (see Appendix A.2 in Menet et al. 2023), it follows that $\theta \approx 0$. For example, if we wanted a modest dynamic range from 0.1 to 10.0 for $\theta = 1/9$, we would already require a very large $R \gg 10^{20}$ for a reliable softmax approximation. Empirically, FAVOR+ is consistently among the worst methods for any tested E.

4.5. The Narrow Dynamic Range of FAVOR+ReLU

While in practice for FAVOR+ReLU the noise is much lower[8], the dynamic range is also more limited. Indeed, for $\measuredangle q, k \in [\pi/2 - \theta, \pi/2 + \theta]$ and $\theta = 1/9$ as above, as well as a fixed norm $\|q\| = \|k\| = N$, the dynamic range is given by $\left(\frac{\cos(\pi/2-1/9)+1}{\cos(\pi/2+1/9)+1}\right)^{\log_2 \pi} \approx 1.44$, see Equation 8. With its limited ability to discriminate, FAVOR+ReLU works best for many term approximations, see Figure 2.

4.6. Adding Bias to Random Fourier Features

Although a naive analysis yields for RFF a similar variance vs dynamic range tradeoff as FAVOR+, the additional clamping to non-negative probabilities gives rise to a different picture. Indeed, clamping reduces the relative variance at the cost of a biased estimation, see Figure 4. Empirically, RFF works best in the intermediate regime $E = 0.6$ and $E = 0.7$ without much of a performance drop, see Figure 2.

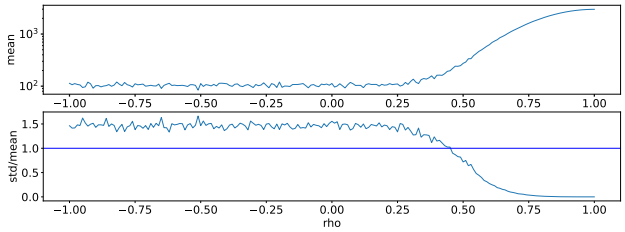


Figure 4. Clamped RFF softmax mean and $\frac{\text{std}}{\text{mean}}$ for $D = 64$, $R = 64$, and $\|q\| = \|k\| = \sqrt{D} = 8$. Estimated using 500 samples.

5. Summary

Based on the insight that attention can be computed not term-by-term, but instead using a hierarchical tree with the geometric mean as an intermediate result, we explored six different heuristics to find regions with high attention.
We identified the benefits and drawbacks of all methods using theoretical and experimental evidence. In particular, we managed to showcase a key logical fallacy in the celebrated Performer, a Linear Transformer based on FAVOR+. On the other hand, EDH, RFF and FAVOR+ReLU each showed promising performance for different values of E.
Finally, more research into alternative probability masses such as ones based on past attention scores will be needed.

[8] see the kernel.py file in our repository

References

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. Rethinking attention with performers, 2022.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022.

Hua, W., Dai, Z., Liu, H., and Le, Q. V. Transformer quality in linear time, 2022.

Jelinek, F., Mercer, R. L., Bahl, L. R., and Baker, J. K. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63, 1977.

Menet, N., Hersche, M., Karunaratne, G., Benini, L., Sebastian, A., and Rahimi, A. Mimonets: Multiple-input-multiple-output neural networks exploiting computation in superposition. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models, 2016.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In Platt, J., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Vieira, T. Heaps for incremental computation, 2016. URL https://timvieira.github.io/blog/post/2016/11/21/heaps-for-incremental-computation/.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021.

Zhu, Z. and Soricut, R. H-transformer-1d: Fast one-dimensional hierarchical attention for sequences, 2021.