
Two heads are better than one: simulating large transformers with small ones

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 The quadratic complexity of self-attention prevents transformers from scaling
2 effectively to long input sequences. On the other hand, modern GPUs and other
3 specialized hardware accelerators are well-optimized for processing small input
4 sequences in transformers during both training and inference. A natural question
5 arises: can we take advantage of the efficiency of small transformers to deal with
6 long input sequences?

7 In this paper, we show that transformers with long input sequences (large trans-
8 formers) can be efficiently simulated by transformers that can only take short input
9 sequences (small transformers). Specifically, we prove that any transformer with
10 input length N can be efficiently simulated by only $O((N/M)^2)$ transformers
11 with input length $M \ll N$, and that this cannot be improved in the worst case.
12 However, we then prove that in various natural scenarios including average-case
13 inputs, sliding window masking and attention sinks, the optimal number $O(N/M)$
14 of small transformers suffice.

1 Introduction

16 The transformer architecture [VSP⁺17] has revolutionized modern machine learning, natural language
17 processing and computer vision. It achieves state-of-the-art performance on various tasks such as
18 language reasoning [Dee21, Ope20], image recognition [KDW⁺21, CMS⁺20] and many others. At
19 the core of the transformer architecture is the attention mechanism, which captures correlations
20 between all pairs of tokens. However, this is also a major bottleneck for transformers, as the quadratic
21 complexity (in both time and memory) of the attention mechanism prohibits effective scaling of
22 transformers as the sequence grows in length. Moreover, it has been theoretically proved that the
23 quadratic complexity cannot be avoided (under popular complexity-theoretic assumptions) [AS24].
24 To address this fundamental issue, there has been a fruitful literature on the design of “subquadratic
25 alternatives” to transformers, where researchers come up with mechanisms that replace the attention
26 mechanism and take subquadratic amount of time [KKL20, CLD⁺21, DFE⁺22, KMZ24, BPC20,
27 GD23]. However, their performances are usually worse than standard transformers, especially on
28 downstream tasks and translation [VPSP23, JBKM24, AY25].

29 In the meantime, modern GPUs are increasingly optimized for handling short-to-moderate transformer
30 contexts [WXQ⁺21, DFE⁺22]. Some companies are even producing specialized hardware for
31 efficient transformer inference that have superior performance on inputs of length between 128 to
32 2048 [Kim24, Etc24]. This approach motivates the following questions:

33 *Can we use small transformers to perform tasks more efficiently than large transformers? Are*
34 *(multiple) small transformers inherently capable of dealing with long contexts?*

35 In this paper, we give positive answers to these questions through the lens of representational strength,
 36 which studies whether one can select parameters for transformers so that they can perform certain
 37 tasks of interest. The representational strength of transformers has been studied broadly in recent years
 38 [BHBK24, SHT23, LAG⁺23], and it is believed that it is one of the core reasons why transformers
 39 outperform previous architectures such as RNN and LSTM [WDL25, AET⁺24, SHT23].

40 Our problem can be stated as follows. Suppose that we have all the parameters of a large transformer
 41 \mathcal{T} with input length N , as well as an input X that we would like to evaluate \mathcal{T} on. However, to
 42 evaluate $\mathcal{T}(X)$, we are not allowed to perform any particularly complicated computations; we are
 43 restricted to simple operations, and to making use of a small transformer \mathcal{O} (as an oracle) that can
 44 only take input sequences of length $M \ll N$. We can input into \mathcal{O} any sequence and parameters that
 45 we can easily compute, and obtain its output. Our goal is to minimize the number of calls to \mathcal{O} that
 46 we need to obtain $\mathcal{T}(X)$ for arbitrary input X of length N .

47 Our main results show that roughly $O((N/M)^2)$ oracle calls suffice, which we show is optimal. In
 48 addition, our algorithm requires minimal processing outside of the oracle calls, and it has properties
 49 needed for efficient training and inference, including that the gradients of its parameters are easily
 50 computed, and that its oracle calls can be computed in parallel in only $O(L)$ rounds of adaptivity,
 51 where L is the number of layers in the large transformer \mathcal{T} .

52 In addition, we show that in many scenarios arising in practice, such as when certain masking
 53 schemes are used, or when the data is not “worst-case” and satisfies some boundedness guarantees,
 54 the information-theoretically optimal $O(N/M)$ oracle calls suffice.

55 Our results provide a new way to deal with long input sequences for transformers, as we prove that
 56 any computation performed by large transformers can be decomposed into computation that only
 57 uses smaller transformers. Our algorithm still takes quadratic amount of floating-point operations,
 58 but if modern GPUs enable faster transformer inference with respect to the “wall-clock” time
 59 when the input sequence is short-to-moderate, then our algorithms allow faster wall-clock time
 60 inference. For example, if the oracle can compute the output using $O(M)$ wall-clock time compared
 61 to the standard $O(M^2)$ time, then the total wall-clock running time of our algorithms will be
 62 $O(N^2/M)$. Our approach is fundamentally different from designing “subquadratic alternatives” to
 63 transformers [KKL20, CLD⁺21, BPC20, KMZ24, GD23]. In particular, our algorithm preserves the
 64 representational strength of transformers (or even improves it), whereas it has been shown that all the
 65 subquadratic alternatives to transformers will lose representational strength as they cannot capture all
 66 the pairwise relationship even approximately [AY25].

67 Now we define our model of computation and state our main contributions in more detail.

68 1.1 Computational Model for Simulating Large Transformers

69 We now describe our model of computation in more detail. We are careful to allow only very simple
 70 operations beyond oracle calls, to ensure that the vast majority of computation can be performed by
 71 efficient hardware for evaluating small transformers, and that the number of oracle calls accurately
 72 measures the complexity of the problem.

73 We are given a large transformer \mathcal{T} with input length N , L layers, H attention heads in each layer,
 74 and embedding dimension d (all the parameters, including the query, key, value matrices in each
 75 of its attention head and multilayer perceptron functions). Throughout this paper, we assume that
 76 $L, H \ll N, d = O(\log N), \Omega(\log N) \leq M < o(N)$, and one can typically imagine $M \approx \sqrt{N}$. Our
 77 goal is to design an algorithm that (approximately) output $\mathcal{T}(X) \in \mathbb{R}^{N \times d}$ for arbitrary input X
 78 (length at most N).

79 We have a limited set of operations we can perform as part of the algorithm. We critically have access
 80 to a small transformer (oracle) \mathcal{O} that can take as input a sequence of length at most $M \ll N$, as
 81 well as the parameters for a transformer which has L' layers and H' attention heads in each layer,
 82 and outputs the transformer evaluated on that sequence. Our algorithm is allowed to:

- 83 1. Feed the oracle \mathcal{O} with input sequences and parameters which are currently in memory to
 84 obtain its output;
- 85 2. Processing: Edit existing vectors or matrices in memory by padding at most $O(d^2)$ fixed
 86 numbers (constants independent of the input) to them.

87 We also assume that all the numbers in input matrices, parameters, and algorithms have $O(\log N)$ -bit
 88 representations. If there exists an algorithm that outputs $Y \in \mathbb{R}^{N \times d}$ such that

$$\|Y[i, :] - \mathcal{T}(X)[i, :]\|_2 \leq \varepsilon \|\mathcal{T}(X)[i, :]\|_2$$

89 for all $i \in [N]$ for very small error $\varepsilon = \Theta(\frac{1}{2^N})$, then we say that this algorithm *simulates* \mathcal{T} . We
 90 want to design algorithms that simulate \mathcal{T} with as fewer oracle calls as possible. (Such an ε is
 91 essentially unavoidable in limited precision architectures, but we will see it will be very helpful in
 92 some algorithms below.)

93 Notice that any such algorithm can be viewed as a composition of oracles and the padding function.
 94 Since we only allow for very simple processing, it is straightforward to compute the gradients of
 95 the padding functions, so training the large model could be done as effectively as computing the
 96 gradients of the small transformer oracles.

97 1.2 Main Results

98 **Quadratic small transformers are sufficient and necessary for worst-case inputs.** As summa-
 99 rized below, our main result shows that any computation performed on a large transformer can be
 100 decomposed into multiple instances of computation performed on smaller transformers with the same
 101 computational complexity or floating-point operations. Since the oracle can only tell us the final
 102 output instead of intermediate embeddings, it might be somewhat surprising that we are able to utilize
 103 all the layers in small transformers.

104 **Theorem 1.1** (Theorem 3.4, Theorem 3.5). *For any transformer \mathcal{T} with L layers, H attention heads*
 105 *in each layer, input length N , embedding dimension d , there exists an algorithm that simulates \mathcal{T}*
 106 *with $O((\frac{N}{M})^2 \cdot \frac{HL}{H'L'})$ calls to a transformer oracle with L' layers, H' attention heads in each layer,*
 107 *input length M , embedding dimension $O(\frac{dH'L'}{H})$. The result still holds when we add causal masking*
 108 *to both large and small transformers.*

109 Notice that these simulations are *tight*, and roughly $(N/M)^2$ oracle calls are necessary in the worst-
 110 case due to computational complexity constraints. To see this, note that when $L = H = L' = H' = 1$,
 111 a straightforward algorithm can compute the responses of T oracle calls in time only $\tilde{O}(TM^2)$. Thus,
 112 since it is known that even approximation of a large attention requires time $\Omega(N^{2-o(1)})$ (under
 113 standard complexity-theoretic assumptions) [AS24], we must have $T \geq ((N/M)^2)^{1-o(1)}$.

114 One might be concerned with the fact that $O((N/M)^2)$ small transformers have way more parameters
 115 compared to one large transformer due to the fact that the query, key and value matrix all have size
 116 independent of N . However, this is not a problem because in our construction, we reuse the parameters
 117 such that the total number of parameters does not depend on N . In fact, all the query, key and value
 118 matrices that we feed into the oracle share most entries with the query, key, value matrices in the
 119 large transformer that are given.

120 **Linear small transformers are weaker but sufficient with average-case inputs.** Even though we
 121 cannot use $O(N/M)$ oracle calls to simulate a large transformer in the worst case, we show that it is
 122 possible when we have assumptions on the queries, keys and values.

123 **Theorem 1.2** (Informal version of Theorem 4.1). *Let \mathcal{T} be a transformer with L layers, H attention*
 124 *heads in each layer, input length N and embedding dimension d . Suppose that the queries, keys and*
 125 *values in the attention heads are all somewhat bounded in how much they may differ from each other,*
 126 *then there exists an algorithm using $O(\frac{N}{M} \cdot \frac{HL}{H'L'})$ oracle calls to simulate \mathcal{T} .*

127 Models such as Hierarchical Transformers [PZV⁺19, LL19, CDF⁺22] split the input sequence into
 128 chunks of size M and send each chunk into a transformer before aggregating the outputs. We give
 129 the first provable guarantees for this approach, showing that $O(N/M)$ small transformers have
 130 approximately equal expressivity as a large transformer when the input data satisfies our assumptions.
 131 This provides a possible explanation of the success of Hierarchical Transformers and relevant ideas
 132 from an expressivity viewpoint.

133 On the other hand, we supplement Theorem 1.2 with its converse, which shows that linear small
 134 transformers are at most as good as one large transformer in terms of expressivity for worst-case
 135 inputs. As a result, when the inputs follow assumptions in Theorem 1.2, linear small transformers are
 136 equally expressive as a large one.

137 **Theorem 1.3** (Theorem 4.2). *Given N/M instances of single layer, single head transformers with*
 138 *input length M and embedding dimension d , there exists an algorithm that simulates them with one*
 139 *call of a single layer, single head transformer with input length $O(N)$ and embedding dimension*
 140 *$O(d)$, along with $O(N/M)$ many matrix multiplications of size $M \times d \times d$.*

141 The small matrix multiplications could be computed in the straightforward way in nearly linear time
 142 $O(Md^2)$ (since $d = O(\log N)$) and thus do not substantially contribute to the running time, and
 143 could not simulate the transformer oracles on their own. We mention their presence (which seems
 144 unavoidable because of the $O(N/M)$ weight matrices of the oracles which must be simulated by a
 145 single large transformer with only a constant number of weight matrices) to emphasize that our other
 146 reductions are even simpler, and do not need such computations.

147 **Efficient simulation of transformers with sliding window and StreamingLLMs.** Sliding window
 148 and StreamingLLM [XTC⁺24] provide ways to make transformer inference more memory efficient.
 149 Both sliding window and StreamingLLM are based on the observation that some attention scores are
 150 higher than others. Sliding window is based on the intrinsic structure of languages, where each token
 151 is more correlated to the previous few tokens. Therefore, for each query we only take into account
 152 the contributions of the keys that are positionally close to it. The StreamingLLM framework is
 153 motivated by the observation that autoregressive LLMs have a surprisingly large amount of attention
 154 score concentrated to the initial tokens, and thus each query only takes into account keys that are
 155 positionally close to it, as well as the first few (usually $3 \sim 5$) keys, which are called “attention
 156 sinks”.

157 We show that in both cases we can use a linear number of small transformer oracle calls to simulate
 158 them, even in the worst case. As summarized below, our result indicates that oracles can capture
 159 efficient attention based on sliding windows efficiently.

160 **Theorem 1.4** (Theorem 5.1). *For any transformer \mathcal{T} with L layers, H attention heads in each layer,*
 161 *input length N , embedding dimension d , constant-size sliding window, there exists an algorithm that*
 162 *simulates \mathcal{T} with $O(\frac{N}{M} \cdot \frac{HL}{H'L'})$ calls to a transformer oracle with L' layers, H' attention heads in*
 163 *each layer, input length M and embedding dimension $O(\frac{dH'L'}{H})$ with causal masking. This result*
 164 *still holds if we have constant-size attention sink.*

165 1.3 Related Work

166 **Representational strength and limitations of transformers.** The representational strength of
 167 transformers has been intensively studied in recent years from a variety of perspectives. To list a
 168 few, [MSS22, MS23, SMW⁺24] study the class of problems that transformers can solve from a
 169 circuit complexity viewpoint; [BAG20, LAG⁺23, Hah20] aim to understand whether transformers
 170 can recognize formal languages; [SHT23] focus on reasoning tasks and show that transformers
 171 are inherently capable of solving sparse averaging; [SHT24] gives important connections between
 172 transformers and the massively parallel computation model; [HSK⁺25, HWL⁺24] uses computational
 173 complexity to characterize the computational limits of diffusion transformers and low-rank adaptation
 174 for transformers; [LCW23] studies attention’s capability of approximating sparse matrices; [YBR⁺20,
 175 YCB⁺20] shows that transformers and many subquadratic variants are universal approximators for
 176 sequence-to-sequence functions;

177 **Fast attention mechanisms.** There has been a fruitful literature of dealing with long input se-
 178 quence by designing “subquadratic alternatives” to transformers, which are variants on the attention
 179 mechanism which can be performed in subquadratic time. For example, researchers have studied
 180 various sparse attention mechanisms that only consider the query-key pairs that have high correlation,
 181 including Reformer [KKL20], Longformer [BPC20], and Hyperattention [HJK⁺24]. Additionally,
 182 there has been work on kernel/low-rank attention that approximates attention mechanism using
 183 kernels such as Performer [CLD⁺21] and Polysketchformer [KMZ24], and there has been a growing
 184 interest in state space models such as Mamba [GD23]. See [TDBM22] for a comprehensive survey
 185 on efficient attention mechanisms. However, [AY25] proves that none of these subquadratic models
 186 can capture all pairwise correlations even approximately as the sequence length grows.

2 Preliminaries

2.1 Transformers

We first define the standard attention mechanism in Transformer.

Definition 2.1 (Attention Mechanism). *Given input $X \in \mathbb{R}^{N \times d}$, query, key, value matrices $W^Q, W^K, W^V \in \mathbb{R}^{d \times m}$, the attention mechanism computes*

$$\text{Attn}(X) = \text{softmax}((XW^Q)(XW^K)^\top)(XW^V) \in \mathbb{R}^{N \times m}.$$

Here we say N is the *context length*, and d is the embedding dimension. We will also call each attention mechanism an *attention head* in the transformer architecture. For convenient notation, we let

$$\{q_1, \dots, q_N\} \in \mathbb{R}^m, \{k_1, \dots, k_N\} \in \mathbb{R}^m, \{v_1, \dots, v_N\} \in \mathbb{R}^m$$

be the rows of XW^Q, XW^K, XW^V respectively. As a result, the attention mechanism is computing

$$\frac{1}{\sum_{j=1}^N \exp(\langle q_i, k_j \rangle)} \sum_{j=1}^N \exp(\langle q_i, k_j \rangle) \cdot v_j$$

for each query q_i .

Another important component in transformers is the Multilayer Perceptron (MLP). A *multilayer perceptron* is a feed-forward, fully-connected neural network consisting of one or more hidden layers using ReLU activation. The universal approximation theorem states that any continuous function with a finite support can be approximated by a neural network with one hidden layer [HSW89]. In light of this, in many relevant works [SHT23, SHT24], MLPs are modeled as arbitrary functions on compact domains.

In this paper, our goal is to use small transformers to simulate large transformers, and we would like to ensure that the MLPs in the small transformers are as simple as possible. We will therefore assume that MLPs in small transformers compute functions $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{O(d)}$ such that:

1. They are at least as strong as the MLPs in the large transformers, i.e. they can do whatever computation that MLPs in the large transformers can do, and
2. They can do basic arithmetic operations on the input vector $x \in \mathbb{R}^d$ or pad fixed numbers to it (both are simple continuous functions) as long as they take $O(d)$ time.

An *attention layer* f with H attention heads consists of H attention mechanisms with query embedding W_h^Q, W_h^K, W_h^V for the h -th attention such that $m = \frac{d}{H}$. The input X is partitioned into H matrices $X[:, D_1], \dots, X[:, D_H] \in \mathbb{R}^{N \times \frac{d}{H}}$ column-wise, where $D_i = \{(i-1)H + 1, \dots, iH\}$ for all i , such that the h -th attention head computes

$$\text{softmax}((XW_h^Q)(XW_h^K)^\top)(XW_h^V) \in \mathbb{R}^{N \times \frac{d}{H}}.$$

All attention outputs are concatenated column-wise and fed through a *layer MLP* ψ such that

$$f(X) := \psi\left(\left[\text{softmax}((XW_h^Q)(XW_h^K)^\top)(XW_h^V)\right]_{h=1}^H\right) \in \mathbb{R}^{N \times d}.$$

Definition 2.2 (Transformer). *A transformer \mathcal{T} with L layers and H attention heads in each layer consists of an input MLP $\phi : \mathbb{R}^{d'} \rightarrow \mathbb{R}^d$ applied token-wise on the input $X \in \mathbb{R}^{N \times d'}$, L attention layers $f_1, \dots, f_L : \mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{N \times d}$ which contain L layer MLPs ψ_1, \dots, ψ_L applied token-wise at the end of each attention layer. For each $2 \leq \ell \leq L$,*

$$X^{(1)} = \psi_1(f_1(\phi(X))), X^{(\ell)} = \psi_\ell(f_\ell(X^{(\ell-1)})).$$

Finally, the transformer \mathcal{T} outputs $\mathcal{T}(X) = X^{(L)}$.

We will simplify the notion of positional encoding into input MLP ϕ and further assume that all the MLPs have positional information of the tokens. In other words, if $x_i = X[i, :]$ is the i -th input token, then $\phi(x_i)$ and $\psi_j(x_i)$ are also functions of i .

Transformer is powerful as computational model, and we refer the readers to Appendix A for more operations that transformers can do that will be useful in our proofs.

Transformer Oracle. A *transformer oracle* \mathcal{O} is a small transformer that can only take inputs of length at most M (its embedding dimension, number of layers, number of heads in each layer, causal masking etc will be specified in result statements). In this paper we are mostly concerned with simulating transformers with large input length N using transformer oracles with input length M such that $M \ll N$ (recall that we assume $\Omega(\log N) \leq M < o(N)$).

2.2 Causal masking, sliding window and StreamingLLM

We first define the most commonly used causal masking in attention heads.

Definition 2.3 (Causal Masking Attention). *Given input $X \in \mathbb{R}^{N \times d}$, query, key, value matrices $W^Q, W^K, W^V \in \mathbb{R}^{d \times m}$, the attention mechanism with causal masking computes*

$$\text{Attn}(X) = \text{softmax}\left(\text{mask}((XW^Q)(XW^K)^\top)\right)(XW^V) \in \mathbb{R}^{N \times m},$$

where the mask function sets all upper triangular entries (not including diagonal entries) to $-\infty$.

Another commonly used masking scheme for efficient transformer inference/training is sliding window, where we only keep the keys whose indices are close to the query index.

Definition 2.4 (Sliding Window Attention). *Given input $X \in \mathbb{R}^{N \times d}$, query, key, value matrices $W^Q, W^K, W^V \in \mathbb{R}^{d \times m}$ and window size $r \geq 1$, the attention mechanism with sliding window of size W computes*

$$\text{Attn}(X) = \text{softmax}\left(\text{window}((XW^Q)(XW^K)^\top)\right)(XW^V) \in \mathbb{R}^{N \times m},$$

where the window function sets all entries in $\{(i, j) : j > i \text{ or } j \leq i - r\}$ to $-\infty$.

In other words, for each query q_i we only look at k_j such that $i - r + 1 \leq j \leq i$.

Finally, StreamingLLM [XTC⁺24] is a framework designed for efficient training with a finite length window. Upon having a fixed-size sliding window, each query also attends to the first s keys (called “attention sinks”), where s is usually a small positive constant (around 3 \sim 5).

Definition 2.5 (Attention Sink). *Given input $X \in \mathbb{R}^{N \times d}$, query, key, value matrices $W^Q, W^K, W^V \in \mathbb{R}^{d \times m}$ and window size $r \geq 1$, sink size $s \geq 1$, the attention mechanism with attention sink computes*

$$\text{Attn}(X) = \text{softmax}\left(\text{sink}((XW^Q)(XW^K)^\top)\right)(XW^V) \in \mathbb{R}^{N \times m},$$

where the sink function sets all entries in $\{(i, j) : j > i \text{ or } s < j \leq i - r\}$ to $-\infty$.

Transformers with causal masking attention, sliding window, and StreamingLLMs are defined exactly the same as transformers except that we replace standard attention mechanisms by attention with causal masking, attention with sliding window and attention with sinks.

2.3 Notation

Throughout the paper, we denote $X \in \mathbb{R}^{N \times d}$ as the input to the transformer, where N is the input length and d is the embedding dimension. For a $N \times d$ matrix X , we use $X[i, :]$ to denote its i -th row, $X[:, j]$ to denote its j -th column, and $X[i, j]$ to denote its (i, j) -th entry. Given sets $S \subseteq [N]$, $D \subseteq [d]$, we use $X[S, :]$ to denote the submatrix consisting of the rows in S , and $X[:, D]$ to denote the submatrix consisting of the columns in D .

We use W^Q, W^K, W^V to denote the query, key and value matrices for attention heads, and we use q_i, k_i, v_i to denote the i -th row of XW^Q, XW^K, XW^V respectively. We denote $D_i = \{(i - 1)H + 1, \dots, iH\}$ and $S_i = \{(i - 1)M + 1, iM\}$.

We use $\mathbf{1}_{a \times b}$ to denote the $a \times b$ matrix whose entries are all 1, and $\mathbf{0}_{a \times b}$ to denote the $a \times b$ matrix whose entries are all 0.

We now turn to proving our results. We give proof sketches and main ideas here; full proofs are deferred to the appendix.

3 Quadratic calls suffice for simulation

In this section, we prove that $O((\frac{N}{M})^2 \cdot \frac{HL}{H'L'})$ small transformers with L' layers and H' attention heads in each layer suffice to simulate a large transformer with L layers and H attention heads in each layer (Theorem 3.4). Our proof roadmap is illustrated as follows, where the arrows $A \rightarrow B$ indicate that A can be simulated by B .

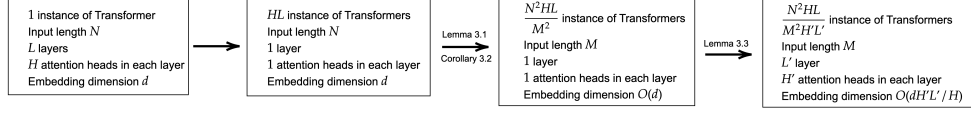


Figure 1: Roadmap of proof of Theorem 3.4

Complete proofs of all the statements can be found in Appendix B. We first show that this is the case when they both only have a single attention head, i.e $H' = H = L = L' = 1$.

Lemma 3.1. *For any single layer, single head transformer \mathcal{T} with input length N , embedding dimension d , there exists an algorithm that simulates \mathcal{T} with $O((\frac{N}{M})^2)$ calls to a transformer oracle with input length M , embedding dimension $O(d)$.*

Proof Sketch. Our high-level idea is to partition the $N \times N$ attention matrix of \mathcal{T} into $\frac{N^2}{M^2}$ blocks of size $M \times M$, and then use a constant number of oracles to separately handle each block. In particular, each block corresponds to two sub-intervals of length M out of the input sequence of length N (one interval for the rows, or queries, and one interval for the columns, or keys), so we can aim to have an oracle for sequence length $O(M)$ compute the contribution of each block. However, two main hurdles arise when implementing this approach which must be overcome.

The first hurdle is that the definition of attention requires us to perform softmax normalization across the entire sequence, so any given block does not have enough information to determine what normalization factor to use. To be more precise, as in Definition 2.1 above, let

$$\{q_1, \dots, q_N\} \in \mathbb{R}^m, \{k_1, \dots, k_N\} \in \mathbb{R}^m, \{v_1, \dots, v_N\} \in \mathbb{R}^m$$

be the rows of XW^Q, XW^K, XW^V respectively. Define the unnormalized attention matrix $\bar{A} \in \mathbb{R}^{N \times N}$, and the normalized (standard) attention matrix $A \in \mathbb{R}^{N \times N}$, by

$$\bar{A}[i, j] = \exp(\langle q_i, k_j \rangle), \text{ and } A[i, j] = \frac{\bar{A}[i, j]}{\sum_{\ell=1}^N \bar{A}[i, \ell]}.$$

The goal of the attention mechanism is to compute, for all i , the sum $\sum_j A[i, j]v_j$, but one could also define the unnormalized attention mechanism, where the goal is to compute $\sum_j \bar{A}[i, j]v_j$. Since $\bar{A}[i, j]$ does not depend on other j s, it is not hard to express a larger unnormalized attention as the sum of unnormalized attentions applied to blocks. The sum over j' in the definition of $A[i, j]$ is what makes this strategy more difficult for attention itself.

We address this in two phases. In the first phase, we show how an oracle for attention can simulate an oracle for unnormalized attention. We do this by adding in synthetic, highly-correlated tokens to the sequence, so that their contribution dominates the normalization factor, but can be easily removed by giving their value token a weight of 0. Using unnormalized oracles, we then compute what the normalization factors should be, as well as the incorrect normalization factors of each block (if attention were just applied to each block separately and summed together). In the second phase, we appropriately rescale the value tokens by the ratio, thus correcting the normalization factor.

The second hurdle is that actually combining the results of different blocks requires a substantial amount of computation. Indeed, in our approach, for each output entry, we compute N/M different values which must be summed together, which would require time $\Theta(N \cdot N/M)$ when done in a straightforward way. To overcome this, we find that carefully-selected weight matrices in the oracles can be used to perform this aggregation more efficiently instead. The idea of using attention to aggregate values appears in prior work on the representational strength of attention as well; we particularly use a construction of [SHT24]. \square

We now move on to generalizing Lemma 3.1 to general H, L, H', L' , i.e., when the transformer \mathcal{T} and the oracles can have multiple heads and layers. A first attempt to do this might use different layers of \mathcal{O} to simulate different layers of \mathcal{T} , but this appears difficult to implement, since Lemma 3.1 requires some processing between layers of attentions that is not available to us when the different layers are connected only through MLPs within an oracle. Indeed, since the output of each attention head needs processing before it can be used in another attention head, it is unclear how to take advantage of more than one layer of each oracle in this way. We instead take a different approach: we use all the attention heads in all the layers of the oracle completely independently from each other to simultaneously simulate $\Theta(H'L')$ different attention heads, and we use these all together to simulate one layer at a time of \mathcal{T} .

First, it is not hard to generalize Lemma 3.1 to general H, L (but still $H' = L' = 1$) by separately simulating each attention head in \mathcal{T} regardless of which layer it is in:

Corollary 3.2. *For any transformer \mathcal{T} with L layers, H attention heads in each layer, input length N , embedding dimension d , there exists an algorithm that simulates \mathcal{T} with $O((\frac{N}{M})^2 \cdot HL)$ calls to a single head, single layer transformer oracle with input length M and embedding dimension $O(\frac{d}{H})$.*

Next we show that a transformer with L' layers, H' attention heads in each layer, input length M , and embedding dimension $O(\frac{dH'L'}{H})$ can be used to simulate $H'L'$ instances of single head, single layer transformers with input length M and embedding dimension $\frac{d}{H}$. In other words, we are able to independently use each attention head in the transformer, regardless of which of the L' layers it appears in:

Lemma 3.3. *One transformer with L' layers, H' attention heads in each layer, input length M and embedding dimension $O(\frac{dH'L'}{H})$ can be used to simulate $H'L'$ independent instances of single layer, single head transformers with input length M and embedding dimension $\frac{d}{H}$.*

Proof Sketch. Consider first when $L' = 1$. A transformer with H' attention heads naturally partitions (by definition) the embedding dimension $O(\frac{dH'}{H})$ into H' parts of size $O(\frac{d}{H})$, and each head separately computes an attention mechanism on one of those parts. The result follows almost directly, with some care to details about MLPs and aggregation.

More care is needed when $L' > 1$. We partition the $\Theta(\frac{dH'L'}{H})$ coordinates of the embedding dimension into L' parts of size $\Theta(\frac{dH'}{H})$ each, and the key idea is that each layer will operate on one of those parts while leaving the rest unchanged. Indeed, weights for the query and keys can be selected so that only the relevant part of the coordinates will impact the attention matrices at each layer, then weights for the values can be selected so that the other parts are passed through the layer without being changed. \square

Finally, we combine Lemma 3.1, Corollary 3.2 and Lemma 3.3 to obtain our main result.

Theorem 3.4. *For any transformer \mathcal{T} with L layers, H attention heads in each layer, input length N , embedding dimension d , there exists an algorithm that simulates \mathcal{T} with $O((\frac{N}{M})^2 \cdot \frac{HL}{H'L'})$ calls to a transformer oracle with L' layers, H' attention heads in each layer, input length M , embedding dimension $O(\frac{dH'L'}{H})$.*

We additionally show that these results still hold when both the large and small transformers have causal masking. The proofs are similar to the proof of Theorem 3.4, and are deferred to Appendix B.

Theorem 3.5. *For any causal transformer \mathcal{T} with L layers, H attention heads in each layer, input length N , embedding dimension d , there exists an algorithm that simulates \mathcal{T} with $O((\frac{N}{M})^2 \cdot \frac{HL}{H'L'})$ calls to a causal transformer oracle with L' layers, H' attention heads in each layer, input length M , embedding dimension $O(\frac{dH'L'}{H})$.*

4 Efficient simulation with average-case input assumptions

4.1 Linear calls suffice for average-case inputs

In this section we prove that if the queries, keys and values in the attention heads are somewhat bounded in how much they may differ from each other, then $O(\frac{N}{M})$ small transformers suffice to

approximate a large transformer. We provide a proof sketch below, and the complete proof can be found in Appendix C.1.

Theorem 4.1. *Let \mathcal{T} be a transformer with L layers, H attention heads in each layer, input length N and embedding dimension d . Suppose there exist absolute constants $C, D > 0$ such that*

$$\frac{1}{C} \leq a_{i,j} \leq C, \text{ and } DN \cdot \max_j \|b_{i,j}\|_2 \leq \left\| \sum_{j=1}^N b_{i,j} \right\|_2$$

where

$$a_{i,j} = \exp(\langle q_i, k_j \rangle), b_{i,j} = \exp(\langle q_i, k_j \rangle) \cdot v_j$$

for any query q_i, k_j, v_j in any attention head. In addition, suppose $M \geq \Omega(\frac{\log N}{\varepsilon^2})$. Then, there exists an algorithm using $O(\frac{N}{M} \cdot \frac{HL}{H'L'})$ oracle calls to a small transformer with L' layers, H' attention heads in each layer, embedding dimension $O(\frac{dH'L'}{H})$ to obtain an $(1 \pm \varepsilon)$ approximation of \mathcal{T} with probability at least 0.9.

Proof Sketch. The high-level idea is to partition the queries into N/M parts of size M each, and then permute the keys (but not the queries) using a random permutation τ . We then aim to approximate the desired quantities $\sum_{j=1}^N \exp(\langle q_i, k_j \rangle)$ and $\sum_{j=1}^N \exp(\langle q_i, k_j \rangle) \cdot v_j$ using rescalings of $\sum_{j \in S_t} \exp(\langle q_i, k_{\tau(j)} \rangle)$ and $\sum_{j \in S_t} \exp(\langle q_i, k_{\tau(j)} \rangle) \cdot v_{\tau(j)}$ (where S_t is the part of size M that contains query q_i). This can be seen as estimating the desired sums by sampling only M of the N summands at random. At the same time, by blocking the queries and keys like this, the samples can be computed using oracle calls similar to Lemma 3.1 above. \square

4.2 Linear small transformers are weaker than large transformers

In this section, we show that $O(\frac{N}{M})$ small transformers can be simulated by a large transformer along with an oracle for performing very small matrix multiplications ($M \times d$ with $d \times d$).

Theorem 4.2. *Given N/M instances of single layer, single head transformers with input length M and embedding dimension d , there exists an algorithm that simulates them with one call of a single layer, single head transformer with input length $O(N)$ and embedding dimension $O(d)$, along with $O(N/M)$ many matrix multiplications of size $M \times d \times d$.*

The key idea behind Theorem 4.2 is to concatenate the tokens from all N/M input sequences into a single long sequence of length N , but then slightly increase the embedding dimension in a way which makes tokens from different short sequences highly uncorrelated with each other. Thus, the large attention will not give much weight to pairs of tokens from different short sequences. The complete proof is delayed to Appendix C.2.

5 Simulation of transformers with sliding window and StreamingLLMs

In our last section, we show that small transformers work well when we add sliding window masking to attention heads, and when in the StreamingLLM framework. Our constructions are similar to above, but with additional techniques to take advantage of the masking structures, and can be found in the Appendix D.

When we consider sliding window attention, we only need to deal with keys that are close to each query. It is intuitive to see that in such scenario, the attention scores of consecutive $M - r$ queries can be covered with a $M \times M$ block matrix, which can be computed using our oracle. Transformers with attention sink is similar to transformers with sliding window, except that we have an extra "sink window" in the beginning for all the queries. These sinks windows can be computed using $O(\frac{N}{M})$ oracle calls.

Theorem 5.1. *For any transformer \mathcal{T} with L layers, H attention heads in each layer, input length N , embedding dimension d , constant-size sliding window, there exists an algorithm that simulates \mathcal{T} with $O(\frac{N}{M} \cdot \frac{HL}{H'L'})$ calls to a transformer oracle with L' layers, H' attention heads in each layer, input length M and embedding dimension $O(\frac{dH'L'}{H})$ with causal masking. This result still holds if we have constant-size attention sink.*

References

- [AET⁺24] Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Re. Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations*, 2024.
- [AS24] Josh Alman and Zhao Song. Fast attention requires bounded entries. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23*, Red Hook, NY, USA, 2024. Curran Associates Inc.
- [AY25] Josh Alman and Hantao Yu. Fundamental limitations on subquadratic alternatives to transformers. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [BAG20] Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the Ability and Limitations of Transformers to Recognize Formal Languages. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online, November 2020. Association for Computational Linguistics.
- [BHBK24] Satwik Bhattamishra, Michael Hahn, Phil Blunsom, and Varun Kanade. Separations in the representational capabilities of transformers and recurrent architectures. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [BPC20] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.
- [CDF⁺22] Ilias Chalkidis, Xiang Dai, Manos Fergadiotis, Prodromos Malakasiotis, and Desmond Elliott. An exploration of hierarchical attention transformers for efficient long document classification, 2022.
- [CLD⁺21] Krzysztof Marcin Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamás Szepesvári, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [CMS⁺20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, page 213–229, Berlin, Heidelberg, 2020. Springer-Verlag.
- [Dee21] Google DeepMind. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446, 2021.
- [DFE⁺22] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: fast and memory-efficient exact attention with io-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS ’22*, Red Hook, NY, USA, 2022. Curran Associates Inc.
- [Etc24] Etched. Etched is Making the Biggest Bet in AI. <https://www.etched.com/announcing-etched>, June 2024. Accessed on May 20, 2025.
- [GD23] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.
- [Hah20] Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020.
- [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.

- [HSK⁺25] Jerry Yao-Chieh Hu, Maojiang Su, En-Jui Kuo, Zhao Song, and Han Liu. Computational limits of low-rank adaptation (LoRA) fine-tuning for transformer models. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- [HWL⁺24] Jerry Yao-Chieh Hu, Weimin Wu, Zhuoru Li, Sophia Pi, Zhao Song, and Han Liu. On statistical rates and provably efficient criteria of latent diffusion transformers (dits). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [JBKM24] Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: transformers are better than state space models at copying. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org, 2024.
- [KDW⁺21] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.
- [Kim24] Hanjoon Kim. RNGD preview: The world’s most efficient AI chip for LLM inference. <https://furiosa.ai/blog/rngd-preview-furiosa-ai>, 2024. Accessed on May 20, 2025.
- [KKL20] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [KMZ24] Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. Polysketchformer: fast transformers via sketching polynomial kernels. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org, 2024.
- [LAG⁺23] Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023.
- [LCW23] Valerii Likhoshesterov, Krzysztof Choromanski, and Adrian Weller. On the expressive flexibility of self-attention matrices. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7):8773–8781, Jun. 2023.
- [LL19] Yang Liu and Mirella Lapata. Hierarchical transformers for multi-document summarization. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy, July 2019. Association for Computational Linguistics.
- [MS23] William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023.
- [MSS22] William Merrill, Ashish Sabharwal, and Noah A. Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.
- [Ope20] OpenAI. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS ’20*, Red Hook, NY, USA, 2020. Curran Associates Inc.
- [PZV⁺19] Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical transformers for long document classification. pages 838–844, 12 2019.
- [SHT23] Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23*, Red Hook, NY, USA, 2023. Curran Associates Inc.

494 [SHT24] Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Transformers, parallel computation,
495 and logarithmic depth. In *Forty-first International Conference on Machine Learning*,
496 *ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

497 [SMW⁺24] Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What
498 formal languages can transformers express? a survey. *Transactions of the Association*
499 *for Computational Linguistics*, 12:543–561, 2024.

500 [TDBM22] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A
501 survey. *ACM Comput. Surv.*, 55(6), December 2022.

502 [VPSP23] Ali Vardasbi*, Telmo Pessoa Pires*, Robin M. Schmidt, and Stephan Peitz. State spaces
503 aren’t enough: Machine translation needs attention. In *EAMT*, 2023.

504 [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N.
505 Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of*
506 *the 31st International Conference on Neural Information Processing Systems*, NIPS’17,
507 page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

508 [WDL25] Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. RNNs are not transformers (yet): The
509 key bottleneck on in-context retrieval. In *The Thirteenth International Conference on*
510 *Learning Representations*, 2025.

511 [WXQ⁺21] Xiaohui Wang, Ying Xiong, Xian Qian, Yang Wei, Lei Li, and Mingxuan Wang. Light-
512 seq2: Accelerated training for transformer-based models on gpus. *arXiv preprint*
513 *arXiv:2110.05722*, 2021.

514 [XTC⁺24] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient
515 streaming language models with attention sinks. In *The Twelfth International Conference*
516 *on Learning Representations*, 2024.

517 [YBR⁺20] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv
518 Kumar. Are transformers universal approximators of sequence-to-sequence functions?
519 In *International Conference on Learning Representations*, 2020.

520 [YCB⁺20] Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J.
521 Reddi, and Sanjiv Kumar. O(n) connections are expressive enough: Universal approx-
522 imability of sparse transformers. In *NeurIPS*, 2020.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: Our main claims made in our abstract and introduction accurately reflect our main results of simulating large transformers with small ones. We discuss different scenarios when we need different number of small transformers as oracles.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: The results are mostly about theoretical worst-case results, even though we have one section on average-case settings. In that section, we clearly discuss why our average-case assumptions are reasonable (because they are observed in experiments in previous work). In addition, our algorithms are efficient in theory, and we expect them to be efficient in practice as well. This is due to the fact that our algorithms use operations that can be efficiently implemented in practice.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All of our results are theoretical, and we provide all the proofs in supplementary material due to the page limit. In addition, we provide proof sketches and high-level ideas of our main results in the main paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [NA]

Justification: This is a purely theoretical paper with no experiment.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [NA]

Justification: This is a purely theoretical paper with no experiment.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [NA]

Justification: This is a purely theoretical paper with no experiment.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [NA]

Justification: This is a purely theoretical paper with no experiment.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [NA]

Justification: This is a purely theoretical paper with no experiment.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We have read the code of ethics and believe that our work conforms with it in every respect.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: We believe that our papers are foundational and theoretical research that are not directly linked to societal impacts. We study the theory of transformers.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper is a purely theoretical paper that poses no risk related to data and models.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification: This paper is a purely theoretical paper that does not use existing assets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: This paper is a purely theoretical paper that does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper is a purely theoretical paper that does not involve crowdsourcing nor research with human objects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper is a purely theoretical paper that does not involve crowdsourcing nor research with human objects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- 835 • Depending on the country in which research is conducted, IRB approval (or equivalent)
836 may be required for any human subjects research. If you obtained IRB approval, you
837 should clearly state this in the paper.
- 838 • We recognize that the procedures for this may vary significantly between institutions
839 and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the
840 guidelines for their institution.
- 841 • For initial submissions, do not include any information that would break anonymity (if
842 applicable), such as the institution conducting the review.

843 16. Declaration of LLM usage

844 Question: Does the paper describe the usage of LLMs if it is an important, original, or
845 non-standard component of the core methods in this research? Note that if the LLM is used
846 only for writing, editing, or formatting purposes and does not impact the core methodology,
847 scientific rigorousness, or originality of the research, declaration is not required.

848 Answer: [NA]

849 Justification: The core method development in this research is a set of algorithmic ideas that
850 do not involve LLMs in any form.

851 Guidelines:

- 852 • The answer NA means that the core method development in this research does not
853 involve LLMs as any important, original, or non-standard components.
- 854 • Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>)
855 for what should or should not be described.

A Transformers as Basic Functions

We show that a single layer, single head transformer (attention mechanism with two MLPs) can act as a few different basic functions on the input matrix X that we will need later.

First we show that we can turn any matrix X into a matrix that is consisted of a block matrix of ones and zeros everywhere else.

Lemma A.1. *There exists a fixed matrix U and input MLP $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d+1}$ such that for any input $X \in \mathbb{R}^{N \times d}$,*

$$\phi(X)U = \begin{pmatrix} \mathbf{1}_{a \times b} & 0 \\ 0 & 0 \end{pmatrix}$$

for any $a \leq N, b \leq d$.

Proof. Let $x_i = X[i, :]$ for all $i \in [N]$. Define $\phi(x_i) = (x_i, 1)$ if $i \leq a$ and $\phi(x_i) = (x_i, 0)$ if $i \geq a + 1$. Let

$$U = \begin{pmatrix} 0_{d \times b} & 0_{d \times (d-b)} \\ \mathbf{1}_{1 \times b} & 0_{d \times (d-b)} \end{pmatrix}$$

It is straightforward to check that $\phi(X)U$ is the matrix desired. \square

Using our newly equipped information, we can construct a transformer that computes the sum of all input tokens.

Lemma A.2. *Given input matrix $X \in \mathbb{R}^{N \times d}$, there exists a single layer, single head transformer with embedding dimension d that can compute $\sum_{i=1}^N X[i, :]$.*

Proof. By Lemma A.1, there exists W^Q, W^K such that $(XW^Q)[i, :] = (XW^K)[i, :] = \mathbf{1}_{1 \times d}$ and $k_i = N \cdot X[i, :]$ for all $1 \leq i \leq N$. As a result, the i -th output will exactly be $\sum_{i=1}^N X[i, :]$. \square

A single layer, single head transformer also allows us to construct a look-up table such that each token can find information from other tokens. Notice that this is impossible if we only have MLPs because MLPs can only operate on a single token.

Lemma A.3 (Lemma D.1 of [SHT24]). *Given input matrix $X \in \mathbb{R}^{N \times d}$, an indexing function $\tau : \mathbb{R}^d \times [N] \rightarrow [N]$ and $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$, there exists a single layer, single head transformer with embedding dimension d such that the i -th output is $\rho(X[\tau(X[i, :], i), :])$.*

B Missing Proofs in Section 3

Lemma B.1 (Lemma 3.1). *For any single layer, single head transformer \mathcal{T} with input length N , embedding dimension d , there exists an algorithm that simulates \mathcal{T} with $O(\frac{N^2}{M^2})$ calls to a transformer oracle with input length M , embedding dimension $O(d)$.*

Proof. Let \mathcal{T} be any single layer, single head transformer with query, key, value matrices $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$ with arbitrary input $X \in \mathbb{R}^{N \times d}$. Let B be an upper bound of the absolute value of all entries in X, W^Q, W^K, W^V . Without losing of generality we can assume the first MLP ϕ is the identity function because otherwise we will compose it with the input MLP in the oracles. In addition, we can also assume that the layer MLP is the identity function, and this is because if not, we can still use identity functions in our oracle layer MLPs to compute the output of the large transformer before the layer MLP, and then use $O(\frac{N}{M})$ oracles as MLP to compute the final output.

Define $Q = XW^Q, K = XW^K, V = XW^V$ and $q_i = Q[i, :], k_i = K[i, :], v_i = V[i, :]$ for all $1 \leq i \leq N$, and let $S_t = \{(t-1)M, \dots, tM\}$ for all $1 \leq t \leq \frac{N}{M}$. Our goal is to simulate

$$\text{softmax}(q_i K^\top) V = \frac{\sum_{j=1}^N \exp(\langle q_i, k_j \rangle) \cdot v_j}{\sum_{j=1}^N \exp(\langle q_i, k_j \rangle)} =: \frac{\sum_{j=1}^N b_{i,j}}{\sum_{j=1}^N a_{i,j}} \quad (1)$$

893 for all $1 \leq i \leq N$, where we define

$$a_{i,j} := \exp(\langle q_i, k_j \rangle), b_{i,j} := \exp(\langle q_i, k_j \rangle) \cdot v_j.$$

894 Our algorithm will proceed in the following two steps:

- 895 1. We approximate $\sum_{j \in S_t} a_{i,j}$ for all $1 \leq i \leq N, 1 \leq t \leq \frac{N}{M}$ up to $(1 + \varepsilon/10)$ multiplicative
896 error using $(\frac{N}{M})^2$ oracle calls;
- 897 2. We (element-wise) approximate $\sum_{j \in S_t} b_{i,j}$ for all $1 \leq i \leq N, 1 \leq t \leq \frac{N}{M}$ up to $(1 + \varepsilon/10)^2$
898 multiplicative error using $(\frac{N}{M})^2$ oracle calls.

899 Now we have enough information to approximate Equation 1 for all $1 \leq i \leq N$ up to $(1 + \varepsilon)^2$
900 multiplicative error using $2(\frac{N}{M})^2$ oracle calls because we can sum up all $\sum_{j \in S_t} a_{i,j}$ and $\sum_{j \in S_t} b_{i,j}$
901 over all S_t for each i and divide. Lemma A.2 implies that this can be done with $O((\frac{N}{M})^2)$ oracle
902 calls.

903 **Step 1: Approximating $\sum_{j \in S_t} a_{i,j}$ for all i, t .** We first consider the case when $i \in S_t$. For any
904 $1 \leq t \leq \frac{N}{M}$, define $W^{Q'} = \begin{pmatrix} W^Q & 0 \\ 0 & 1 \end{pmatrix}, W^{K'} = \begin{pmatrix} W^K & 0 \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{(d+1) \times (d+1)}$ and $C > 0$ to be
905 determined such that

$$Q' := \phi(X[S_t, :]) \cdot W^{Q'} = \begin{pmatrix} X[S_t, :] & 1 \\ 0 & C \end{pmatrix} \cdot \begin{pmatrix} W^Q & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} q_{(t-1)M+1} & 1 \\ \vdots & \vdots \\ q_{tM} & 1 \\ 0 & C \end{pmatrix} \in \mathbb{R}^{(M+1) \times (d+1)},$$

$$K' = \phi(X[S_t, :]) \cdot W^{K'} = \begin{pmatrix} X[S_t, :] & 1 \\ 0 & C \end{pmatrix} \cdot \begin{pmatrix} W^K & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} k_{(t-1)M+1} & 1 \\ \vdots & \vdots \\ k_{tM} & 1 \\ 0 & C \end{pmatrix} \in \mathbb{R}^{(M+1) \times (d+1)},$$

$$V' = \begin{pmatrix} 1 & \dots & 1 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 1 & \dots & 1 & 0 \\ 0 & \dots & 0 & 0 \end{pmatrix} \in \mathbb{R}^{(M+1) \times (d+1)}.$$

906 (We can construct $W^{V'}$ and add an extra dimension using MLP such that we obtain V' using Lemma
907 A.1, but we omit it for simplicity.) Therefore, we have

$$Q'(K')^\top = \begin{pmatrix} \langle q_{(t-1)M+1}, k_{(t-1)M+1} \rangle + 1 & \dots & \langle q_{(t-1)M+1}, k_{tM} \rangle + 1 & C \\ \vdots & \ddots & \vdots & \vdots \\ \langle q_{tM}, k_{(t-1)M+1} \rangle + 1 & \dots & \langle q_{tM}, k_{tM} \rangle + 1 & C \\ C & \dots & C & C^2 \end{pmatrix}$$

908 Finally, we can calculate that the (i, j) -th entry ($1 \leq i, j \leq M$) of $\text{softmax}(Q'(K')^\top)V'$ is

$$\frac{\sum_{j=1}^M \exp(\langle q_{(t-1)M+i}, k_{tM+j} \rangle + 1)}{\sum_{j=1}^M \exp(\langle q_{(t-1)M+i}, k_{tM+j} \rangle + 1) + \exp(C)} = \frac{\sum_{j=1}^M a_{(t-1)M+i, tM+j}}{\sum_{j=1}^M a_{(t-1)M+i, tM+j} + \exp(C-1)}.$$

909 For any $\varepsilon > 0$, letting $C \geq \log M + \log \frac{3}{\varepsilon} + dB^2 = \text{poly}(N)$ ensures that this is a $(1 + \varepsilon/10)$ -
910 approximation because

$$\frac{1}{1 + \varepsilon/10} \cdot \sum_{j=1}^M a_{(t-1)M+i, tM+j} \leq \frac{\sum_{j=1}^M a_{(t-1)M+i, tM+j}}{\sum_{j=1}^M a_{(t-1)M+i, tM+j} + \exp(C-1)} \leq \sum_{j=1}^M a_{(t-1)M+i, tM+j}.$$

911 Therefore, we can use one oracle call to approximate $\sum_{\ell \in S_t} \exp(\langle q_i, k_\ell \rangle)$ up to $(1 + \varepsilon/10)$ multi-
912 plicative error for all $1 \leq i \leq M$ when $i \in S_t$.

Now suppose $i \in S_{t'}$ for some $t' \neq t$, feeding the oracle with $X[S_t \cup S_{t'}, :]$ using the same $W^{Q'}, W^{K'}, W^{V'}$ above will give us a $(1 + \varepsilon/10)$ approximation of $\sum_{j \in S_t \cup S_{t'}} a_{i,j}$ for all $i \in S_t \cup S_{t'}$. Since we already have an $(1 + \varepsilon/10)$ approximation of $\sum_{j \in S_{t'}} a_{i,j}$ for all $i \in S_{t'}$, subtracting one from another gives us an $(1 + \varepsilon/10)$ approximation of $\sum_{j \in S_t} a_{i,j}$ for all $i \in S_{t'}$. As a result, we can use $(\frac{N}{M})^2$ oracle calls to approximate $\sum_{j \in S_t} a_{i,j}$ for all $1 \leq i \leq N$ and $1 \leq t \leq \frac{N}{M}$ up to $(1 + \varepsilon/10)$ multiplicative error.

Step 2: Approximating $\sum_{j \in S_t} b_{i,j}$ for all i, t . Firstly, since we have already (approximately) computed $\sum_{\ell \in S_t} a_{i,\ell}$ for all i, t in step 1, if $i \in S_t$ we can exactly compute $\sum_{j \in S_t} b_{i,j}$ using one extra oracle call simply by feeding the oracle with $X[S_t, :], W^Q, W^K, W^V$ and multiply the output with $\sum_{\ell \in S_t} a_{i,\ell}$. As a result, $\frac{N}{M}$ oracle calls suffice to give us an $(1 + \varepsilon/10)$ approximation of $\sum_{j \in S_t} b_{i,j}$ for all $i \in S_t$ and $1 \leq t \leq \frac{N}{M}$. It remains for us to approximate $\sum_{j \in S_t} b_{i,j}$ for all $i \in S_{t'}$ where $t' \neq t$.

We feed the oracle with $X[S_t \cup S_{t'}, :], W^Q, W^K, W^V$ such that it gives us

$$\begin{aligned} \frac{\sum_{j \in S_t \cup S_{t'}} \exp(\langle q_i, k_j \rangle) \cdot v_j}{\sum_{j \in S_t \cup S_{t'}} \exp(\langle q_i, k_j \rangle)} &= \frac{\sum_{j \in S_t} \exp(\langle q_i, k_j \rangle) \cdot v_j}{\sum_{j \in S_t \cup S_{t'}} \exp(\langle q_i, k_j \rangle)} + \frac{\sum_{j \in S_{t'}} \exp(\langle q_i, k_j \rangle) \cdot v_j}{\sum_{j \in S_t \cup S_{t'}} \exp(\langle q_i, k_j \rangle)} \\ &= \frac{\sum_{j \in S_t} b_{i,j}}{\sum_{j \in S_t \cup S_{t'}} a_{i,j}} + \frac{\sum_{j \in S_{t'}} b_{i,j}}{\sum_{j \in S_t \cup S_{t'}} a_{i,j}} \end{aligned}$$

for all $i \in S_t \cup S_{t'}$. Notice that we already have $(1 + \varepsilon/10)$ approximation of $\sum_{j \in S_{t'}} b_{i,j}$, $\sum_{j \in S_t} a_{i,j}$ and $\sum_{j \in S_{t'}} a_{i,j}$, we can now $(1 + \varepsilon/10)^2$ approximate $\sum_{j \in S_t} b_{i,j}$ with the output above. Therefore, $O((\frac{N}{M})^2)$ oracle calls allow us to $(1 + \varepsilon/10)^2$ approximate $\sum_{j \in S_t} b_{i,j}$ for all i, t .

In total we need $(\frac{N}{M})^2$ oracle calls in step 2. To obtain $(1 + O(\frac{1}{2^N}))$ approximation, we need $\varepsilon \leq O(\frac{1}{2^N})$, which means that the number C with most bits requires $\text{poly}(N)$ bits, concluding the proof. \square

Corollary B.2 (Corollary 3.2). *For any transformer \mathcal{T} with L layers, H attention heads in each layer, input length N , embedding dimension d , there exists an algorithm that simulates \mathcal{T} with $O((\frac{N}{M})^2 \cdot HL)$ calls to a single head, single layer transformer oracle with input length M and embedding dimension $O(\frac{d}{H})$.*

Proof. We simply compute \mathcal{T} layer by layer and head by head. For each layer, we compute the output of each attention head, which requires $O((\frac{N}{M})^2 \cdot H)$ oracle calls using Lemma B.1, and repeat for each layer. \square

Lemma B.3 (Lemma 3.3). *One transformer with L' layers, H' attention heads in each layer, input length M and embedding dimension $O(\frac{dH'L'}{H})$ can be used to simulate $H'L'$ independent instances of single layer, single head transformers with input length M and embedding dimension $\frac{d}{H}$.*

Proof. Given $H'L'$ independent instances with input $X_i \in \mathbb{R}^{M \times \frac{d}{H}}$, we label the instances by $(h, \ell) \in H' \times L'$ and concatenate them together by columns to $X \in \mathbb{R}^{M \times \frac{dH'L'}{H}}$ such that

$$X = [X_{1,1}, X_{1,2}, \dots, X_{1,L'}, X_{2,1}, \dots, X_{H',L'}].$$

As a result, in layer one, $[X_{h,1}, \dots, X_{h,L'}] \in \mathbb{R}^{M \times \frac{dL'}{H}}$ is sent to attention head h . Let $W^Q, W^K, W^V \in \mathbb{R}^{\frac{d}{H} \times \frac{d}{H}}$ denote the query, key and value matrices for $X_{h,1}$. We construct $W^{Q'}, W^{K'}, W^{V'}$ as

$$W^{Q'} = \begin{pmatrix} W^Q \\ 0_{\frac{d}{H} \times \frac{d}{H}} \\ \vdots \\ 0_{\frac{d}{H} \times \frac{d}{H}} \end{pmatrix}, W^{K'} = \begin{pmatrix} W^K \\ 0_{\frac{d}{H} \times \frac{d}{H}} \\ \vdots \\ 0_{\frac{d}{H} \times \frac{d}{H}} \end{pmatrix}, W^{V'} = \begin{pmatrix} W^K \\ 0_{\frac{d}{H} \times \frac{d}{H}} \\ \vdots \\ 0_{\frac{d}{H} \times \frac{d}{H}} \end{pmatrix}$$

and layer MLP the same as the layer MLP in instance $X_{h,1}$ such that attention head h in layer 1 exactly computes instance $X_{h,1}$. The same argument holds for all h , and therefore layer one of the large transformer computes instance $X_{h,1}$ for all $1 \leq h \leq H'$. Since the outputs of all H' attention heads are stored at predefined locations after each layer, we can repeat this process for L' times to compute all instances. \square

Theorem B.4 (Theorem 3.4). *For any transformer \mathcal{T} with L layers, H attention heads in each layer, input length N , embedding dimension d , there exists an algorithm that simulates \mathcal{T} with $O((\frac{N}{M})^2 \cdot \frac{HL}{H'L'})$ calls to a transformer oracle with L' layers, H' attention heads in each layer, input length M , embedding dimension $O(\frac{dH'L'}{H})$.*

Proof. This simply follows from Corollary B.2 and Lemma B.3. \square

Theorem B.5 (Theorem 3.5). *For any transformer \mathcal{T} with L layers, H attention heads in each layer, input length N , embedding dimension d and causal masking, there exists an algorithm that simulates \mathcal{T} with $O((\frac{N}{M})^2 \cdot \frac{HL}{H'L'})$ calls to a transformer oracle with L' layers, H' attention heads in each layer, input length M , embedding dimension $O(\frac{dH'L'}{H})$ and causal masking.*

We first show that a single layer, single head transformer with input length M and causal masking can $(1 \pm \varepsilon)$ approximate $\sum_{j=1}^i \exp(\langle q_i, k_j \rangle)$ for all $1 \leq i \leq M$ and $\sum_{j=1}^i \exp(\langle q_i, k_j \rangle) \cdot v_j$ given X, W^Q, W^K, W^V matrices.

Claim B.6. *Given any $X \in \mathbb{R}^{M \times d}, W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$, 2 calls to a single layer, single head transformer oracle \mathcal{O} with input length M , embedding dimension d and causal masking suffices to $(1 \pm \varepsilon)$ approximate $\sum_{j=1}^i \exp(\langle q_i, k_j \rangle)$ for all $1 \leq i \leq M$ and $\sum_{j=1}^i \exp(\langle q_i, k_j \rangle) \cdot v_j$.*

Proof. We define

$$W^{Q'} = \begin{pmatrix} 1 & 0 \\ 0 & W^Q \end{pmatrix}, W^{K'} = \begin{pmatrix} 1 & 0 \\ 0 & W^K \end{pmatrix} \in \mathbb{R}^{(d+1) \times (d+1)}$$

and $C > 0$ to be determined such that

$$Q' = \phi(X[S_t, :]) \cdot W^{Q'} = \begin{pmatrix} 0 & C \\ X[S_t, :] & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & W^Q \end{pmatrix} = \begin{pmatrix} 0 & C \\ q_{(t-1)M+1} & 1 \\ \vdots & \vdots \\ q_{tM} & 1 \end{pmatrix} \in \mathbb{R}^{(M+1) \times (d+1)}$$

$$K' = \phi(X[S_t, :]) \cdot W^{K'} = \begin{pmatrix} 0 & C \\ X[S_t, :] & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & W^K \end{pmatrix} = \begin{pmatrix} 0 & C \\ k_{(t-1)M+1} & 1 \\ \vdots & \vdots \\ k_{tM} & 1 \end{pmatrix} \in \mathbb{R}^{(M+1) \times (d+1)}$$

$$V' = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 1 & \dots & 1 \\ 0 & \vdots & \ddots & \vdots \\ 0 & 1 & \dots & 1 \end{pmatrix} \in \mathbb{R}^{(M+1) \times (d+1)}.$$

As a result, we have

$$Q'(K')^\top = \begin{pmatrix} C^2 & C & \dots & C \\ C & \langle q_{(t-1)M+1}, k_{(t-1)M+1} \rangle & \dots & q_{(t-1)M+1}, k_{tM} \\ \vdots & \vdots & \ddots & \vdots \\ C & q_{tM}, k_{(t-1)M+1} & \dots & q_{tM}, k_{tM} \end{pmatrix}.$$

Finally, we can calculate that the (i, j) -th entry ($2 \leq i, j \leq M+1$) of $\text{softmax}(\text{mask}(Q'(K')^\top))V'$ is

$$\frac{\sum_{j=1}^i \exp(\langle q_{(t-1)M+i-1}, k_{tM+j-1} \rangle + 1)}{\sum_{j=1}^i \exp(\langle q_{(t-1)M+i-1}, k_{tM+j-1} \rangle + 1) + \exp(C)} = \frac{\sum_{j=1}^i a_{(t-1)M+i-1, tM+j-1}}{\sum_{j=1}^i a_{(t-1)M+i-1, tM+j-1} + \exp(C-1)}.$$

972 Now we can set $C \geq \log M + \log \frac{1}{\varepsilon} + dB^2 = \text{poly}(N)$ and $(1 + \varepsilon)$ approximate
 973 $\sum_{j \in S_t \cap [i]} \exp(\langle q_i, k_j \rangle)$. Now we can $(1 + \varepsilon/1)$ approximate $\sum_{j \in S_t \cap [i]} \exp(\langle q_i, k_j \rangle) \cdot v_j$ using
 974 one extra oracle with $X[S_t, :], W^Q, W^K, W^V$. \square

975 *Proof of Theorem B.5.* The proof is similar to the proof of Theorem B.4. First notice that Lemma
 976 B.3 still holds if we add causal masking to the transformers because the proof is not affected by
 977 causal masking. In addition, the analog of Corollary B.2 will hold even if we add causal masking to
 978 the transformers if we can show that Lemma B.1 holds under causal masking since the proof is the
 979 same. Therefore, it suffices to prove Lemma B.1 under causal masking.

980 Let \mathcal{T} be any single layer, single head transformer with query, key, value matrices $W^Q, W^K, W^V \in$
 981 $\mathbb{R}^{d \times d}$ and causal masking, with arbitrary input $X \in \mathbb{R}^{N \times d}$. For the same reason as Lemma B.1, we
 982 assume without losing of generality that both the input MLP and layer MLP are identity functions,
 983 and we use the same notation as in Lemma B.1. Our goal is to approximate

$$\frac{\sum_{j=1}^i \exp(\langle q_i, k_j \rangle) \cdot v_j}{\sum_{j=1}^i \exp(\langle q_i, k_j \rangle)} =: \frac{\sum_{j=1}^i b_{i,j}}{\sum_{j=1}^i a_{i,j}}.$$

984 for all $1 \leq i \leq N$, and notice that we already include causal masking by letting $j \leq i$.

985 Our algorithm is similar to that of Lemma B.1. Notice that in attention with causal masking, after
 986 we partition the $N \times N$ attention matrix into $M \times M$ block matrices, there are two types of block
 987 matrices:

- 988 1. **Type 1.** Corresponds to $S_t \times S_{t'}$ for some $t' < t$, such that no entry is masked. We show
 989 that we can $(1 \pm \varepsilon/3)$ approximate $\sum_{j \in S_t} \exp(\langle q_i, k_j \rangle)$ and $\sum_{j \in S_t} \exp(\langle q_i, k_j \rangle) \cdot v_j$ for
 990 all $i \in S_t, 1 \leq t \leq N/M$. Notice that Claim B.6 shows that we can $(1 \pm \varepsilon/3)$ approximate
 991 $\sum_{j \in S_t \cap [i]} \exp(\langle q_i, k_j \rangle)$ and $\sum_{j \in S_t \cap [i]} \exp(\langle q_i, k_j \rangle) \cdot v_j$ for all $i \in S_t, 1 \leq t \leq N/M$. Now
 992 it suffices to $(1 \pm \varepsilon/3)$ approximate $\sum_{j=i+1}^{tM} \exp(\langle q_i, k_j \rangle)$ and $\sum_{j=i+1}^{tM} \exp(\langle q_i, k_j \rangle) \cdot v_j$
 993 for all $i \in S_t, 1 \leq t \leq N/M$ such that

$$\begin{aligned} \sum_{j \in S_t} \exp(\langle q_i, k_j \rangle) &= \sum_{j \in S_t \cap [i]} \exp(\langle q_i, k_j \rangle) + \sum_{j=i+1}^{tM} \exp(\langle q_i, k_j \rangle) \\ \sum_{j \in S_t} \exp(\langle q_i, k_j \rangle) \cdot v_j &= \sum_{j \in S_t \cap [i]} \exp(\langle q_i, k_j \rangle) \cdot v_j + \sum_{j=i+1}^{tM} \exp(\langle q_i, k_j \rangle) \cdot v_j. \end{aligned}$$

994 Notice that Lemma A.3 allows us to permute the queries and keys in any order, and
 995 therefore we can let $q'_i = (q_i, 1), k'_i = (k_i, 1)$ and add a dummy token $x_0 = (0, -C)$
 996 such that $\langle q'_i, x_0 \rangle = -C$ for any i . Finally, we order the queries from top to bottom by
 997 $x_0, q_{tM}, \dots, q_{(t-1)M+1}$ and keys from left to right by $x_0, k_{tM}, \dots, k_{(t-1)M+1}$ such that
 998 each q_i is only attending to k_j such that $i+1 \leq j \leq tM$ and x_0 . Since we set $C \leq \text{poly}(N)$
 999 to be large enough, a similar argument in Lemma B.1 allows us to approximate.

- 1000 2. **Type 2.** Corresponds to $S_t \times S_t$, such that all the entries above the diagonal are masked.
 1001 This directly follows from Claim B.6 by letting the approximation factor to be $\varepsilon/3$ and extra
 1002 divisions.

1003 In total we need $O((\frac{N}{M})^2)$ oracle calls, and the proof is complete. \square

1004 C Missing Proofs in Section 4

1005 C.1 Missing Proofs in Section 4.1

1006 **Theorem C.1** (Theorem 4.1). *Let \mathcal{T} be a transformer with L layers, H attention heads in each layer,*
 1007 *input length N and embedding dimension d . Suppose there exist absolute constants $C, D > 0$ such*
 1008 *that*

$$\frac{1}{C} \leq a_{i,j} \leq C, \text{ and } DN \cdot \max_j \|b_{i,j}\|_2 \leq \left\| \sum_{j=1}^N b_{i,j} \right\|_2$$

1009 where

$$a_{i,j} = \exp(\langle q_i, k_j \rangle), b_{i,j} = \exp(\langle q_i, k_j \rangle) \cdot v_j$$

1010 for any query q_i, k_j, v_j in any attention head. In addition, suppose $M \geq \Omega(\frac{\log N}{\varepsilon^2})$. Then, there exists
 1011 an algorithm using $O(\frac{N}{M} \cdot \frac{HL}{H'L'})$ oracle calls to a small transformer with L' layers, H' attention
 1012 heads in each layer, embedding dimension $O(\frac{dH'L'}{H})$ to obtain an $(1 \pm \varepsilon)$ approximation of \mathcal{T} with
 1013 probability at least 0.9.

1014 *Proof.* By Corollary B.2 and Lemma B.3, it suffices to prove the Theorem with $H = L = 1$ because
 1015 when we generalize, all the attention head computation will be in parallel with each other.

1016 Let \mathcal{T} be any attention head with query, key, value matrices $W^Q, W^K, W^V \in \mathbb{R}^{d \times m}$ with arbitrary
 1017 input $X \in \mathbb{R}^{N \times d}$. Without losing of generality we can assume the first MLP ϕ is the identity function
 1018 because otherwise we will compose our oracle MLP with ϕ . In addition, we can also assume that the
 1019 layer MLP is the identity function, and this is because if not, we can still use identity functions in our
 1020 oracle layer MLPs to compute the output of the large transformer before the layer MLP, and then use
 1021 $O(\frac{N}{M})$ oracles as MLP to compute the final output.

1022 Define $Q = XW^Q, K = XW^K, V = XW^V$ such that $q_i = Q[i, :], k_i = K[i, :], v_i = V[i, :]$ for all
 1023 $1 \leq i \leq N$, and let $S_t = \{(t-1)M, \dots, tM\}$ for all $1 \leq t \leq \frac{N}{M}$. Our goal is to approximate

$$\text{softmax}(q_i K^\top) V = \frac{\sum_{j=1}^N \exp(\langle q_i, k_j \rangle) \cdot v_j}{\sum_{j=1}^N \exp(\langle q_i, k_j \rangle)} = \frac{\sum_{j=1}^N b_{i,j}}{\sum_{j=1}^N a_{i,j}} \quad (2)$$

1024 for all $1 \leq i \leq N$. The algorithm will be divided into two parts:

- 1025 1. Approximating $\sum_{j=1}^N a_{i,j}$ up to $(1 \pm \frac{\varepsilon}{10})$ multiplicative error for all $1 \leq i \leq N$ using $O(\frac{N}{M})$
 1026 oracle calls;
- 1027 2. Approximating $\sum_{j=1}^N b_{i,j}$ up to $(1 \pm \frac{\varepsilon}{10})^2$ multiplicative error (element-wise) for all $1 \leq$
 1028 $i \leq N$ using $O(\frac{N}{M})$ oracle calls.

1029 Therefore, combining the approximation (with an extra $O(\frac{N}{M})$ oracles for division) gives us a $(1 \pm \varepsilon)$
 1030 approximation of \mathcal{T} .

1031 **Approximating $\sum_{j=1}^N a_{i,j}$ for all $1 \leq i \leq N$.** Let $i \in S_t$ for some $1 \leq t \leq \frac{N}{M}$. We pick a
 1032 random permutation τ of $[N]$, and by Lemma A.3 we can use $O(\frac{N}{M})$ oracle calls¹ (one for each
 1033 $X[S_t, :]$) to map x_i to $[x_i, x_{\tau(i)}]$ for all $1 \leq i \leq N$. Now we use the first MLP in the oracles to map
 1034 $(X[S_t, :], X[\tau(S_t), :])$ to

$$X' = \begin{pmatrix} X[S_t, :] & X[\tau(S_t), :] & 1 \\ 0 & 0 & C \end{pmatrix}$$

1035 for a constant $C \geq \log M + \log \frac{3}{\varepsilon} + dB^2 = \text{poly}(N)$ exactly the same as in Lemma B.1.

$$W^{Q'} = \begin{pmatrix} W^Q & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}, W^{K'} = \begin{pmatrix} 0 & 0 \\ W^K & 0 \\ 0 & 1 \end{pmatrix}$$

1036 in the oracle such that

$$Q' = X'W^{Q'} = \begin{pmatrix} q_{(t-1)M+1} & 1 \\ \vdots & 1 \\ q_{tM} & 1 \\ 0 & C \end{pmatrix}, K' = X'W^{K'} = \begin{pmatrix} k_{\tau((t-1)M+1)} & 1 \\ \vdots & 1 \\ k_{\tau(tM)} & 1 \\ 0 & C \end{pmatrix}, V' = \begin{pmatrix} 1 & \dots & 1 & 0 \\ \vdots & \ddots & \vdots & 0 \\ 1 & \dots & 1 & 0 \\ 0 & \dots & 0 & 0 \end{pmatrix}.$$

¹In practice, one would likely perform this permutation directly rather than using oracle calls, such as using the pytorch utility `randperm`. However, since it is a negligible additional number of calls, we use oracle calls here to simplify the computational model.

1037 Now we can $(1 + \varepsilon/10)$ -approximate $\sum_{j \in S_t} a_{i,\tau(j)}$ for all i using $\frac{N}{M}$ oracle calls (the remaining
 1038 proof is exactly the same as the proof of Lemma 3.1). Our estimator of $\sum_{j=1}^N a_{i,j}$ will be

$$\frac{N}{M} \sum_{j \in S_t} a_{i,\tau(j)}.$$

1039 Our estimator is unbiased because

$$\mathbf{E} \left[\sum_{j \in S_t} a_{i,\tau(j)} \right] = \frac{M}{N} \cdot \sum_{j=1}^N a_{i,j}.$$

1040 We can use Hoeffding's Inequality to show that

$$\begin{aligned} \Pr \left[\left| \sum_{j \in S_t} a_{i,\tau(j)} \cdot \frac{N}{M} - \sum_{j=1}^N a_{i,j} \right| \geq \frac{\varepsilon}{10} \sum_{j=1}^N a_{i,j} \right] &= \Pr \left[\left| \sum_{j \in S_t} a_{i,\tau(j)} - \frac{M}{N} \sum_{j=1}^N a_{i,j} \right| \geq \frac{\varepsilon M}{10N} \sum_{j=1}^N a_{i,j} \right] \\ &\leq 2 \exp \left(-0.08 M \varepsilon^2 \left(\frac{\sum_{j=1}^N a_{i,j}}{N \cdot C} \right)^2 \right) \\ &\leq 2 \exp \left(-\frac{0.08 M \varepsilon^2}{C^4} \right), \end{aligned}$$

1041 which is at most $\frac{1}{20N}$ if $M \geq \frac{C^4(\log 40 + \log N)}{0.08 \varepsilon^2}$, which is true by our assumption on M . A union
 1042 bound over all $i \in [N]$ allows us to show that we get a $(1 \pm \varepsilon/10)^2$ approximation of $\sum_{j=1}^N a_{i,j}$ with
 1043 probability at least 0.95.

1044 **Approximating** $\sum_{j=1}^N b_{i,j}$ for all $1 \leq i \leq N$. Let $i \in S_t$ for some $1 \leq t \leq \frac{N}{M}$. We pick a random
 1045 permutation τ of $[N]$, and the exact same construction for $X', W^{Q'}, W^{K'}$ with

$$W^{V'} = \begin{pmatrix} 0 & 0 \\ W^V & 0 \\ 0 & 1 \end{pmatrix}, X' W^{V'} = \begin{pmatrix} v_{\tau((t-1)M+1)} & 1 \\ \vdots & 1 \\ v_{\tau(tM)} & 1 \\ 0 & C \end{pmatrix}$$

1046 allows us to calculate

$$\sum_{j=1}^M \frac{e \cdot a_{i,\tau((t-1)M+1)}}{e \cdot \sum_{j=1}^M a_{i,\tau((t-1)M+j)} + C} \cdot v_{\tau((t-1)M+j)} = \frac{e}{e \cdot \sum_{j=1}^M a_{i,\tau((t-1)M+j)} + C} \cdot \sum_{j \in S_t} b_{i,\tau(j)}.$$

1047 Therefore, we can $(1 \pm \varepsilon/10)^2$ -approximate $\sum_{j \in S_t} b_{i,\tau(j)}$ for all i using $\frac{N}{M}$ calls since we have
 1048 $(1 \pm \varepsilon/10)^2$ approximation of all $\sum_{j \in S_t} a_{i,\tau(j)}$. Our estimator of $\sum_{j=1}^N b_{i,j}$ will be

$$\frac{N}{M} \sum_{j \in S_t} b_{i,\tau(j)}.$$

1049 A similar argument shows that our estimator is unbiased, and we can again use Hoeffding's Inequality
 1050 to show that

$$\begin{aligned} \Pr \left[\left\| \frac{N}{M} \sum_{j \in S_t} b_{i,\tau(j)} - \sum_{j=1}^N b_{i,j} \right\|_2 \geq \frac{\varepsilon}{10} \left\| \sum_{j=1}^N b_{i,j} \right\|_2 \right] &\leq 2 \cdot \exp \left(-\frac{\varepsilon^2 M \cdot \left\| \sum_{j=1}^N b_{i,j} \right\|_2^2}{800 N^2 (\max_j \|b_{i,j}\|_2)^2} \right) \\ &\leq 2 \cdot \exp \left(-\frac{\varepsilon^2 M D^2}{800} \right) \end{aligned}$$

1051 which is at most $\frac{1}{20N}$ when $M \geq \frac{800(\log d + \log 40 + \log N)}{\varepsilon^2 D^2}$. A union bound over all $1 \leq i \leq N$ allows
 1052 us to show that we get a $(1 \pm \varepsilon/10)^3$ approximation with probability at least 0.95. Finally, a union
 1053 bound shows that we can approximate $\sum_{j=1}^N a_{i,j}$ and $\sum_{j=1}^N b_{i,j}$ with up to $(1 \pm \varepsilon/10)^3 < (1 \pm \varepsilon)$
 1054 approximation with probability at least 0.9. The number of oracle calls that we need is $O(\frac{N}{M})$ in
 1055 total. \square

1056 C.2 Missing Proofs in Section 4.2

1057 **Theorem C.2** (Theorem 4.2). *Given $\frac{N}{M}$ instances of single layer, single head transformers with input*
 1058 *length M and embedding dimension d , there exists an algorithm that simulates them with one call*
 1059 *of a single layer, single head transformer with input length $O(N)$ and embedding dimension $O(d)$,*
 1060 *along with $O(\frac{N}{M})$ many matrix multiplications of size $M \times d \times d$.*

1061 *Proof.* We assume that the input/layer MLPs in all the instances are identity functions for the same
 1062 reason as in Lemma B.1. Let $X_i \in \mathbb{R}^{M \times d}$ be the input, $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d \times d}$ be the query, key
 1063 and value matrices for each instance $1 \leq i \leq \frac{N}{M}$. We assume

$$\|X_i\|_{\infty,2}^2 \cdot \|W_i^Q\|_2 \cdot \|W_i^V\|_2 \leq C_1 \leq \text{poly}(N), \|X_i\|_{\infty,2} \cdot \|W_i^V\|_2 \leq C_2 \leq \text{poly}(N)$$

1064 for some C_1, C_2 . Our goal is to calculate

$$\text{softmax}((X_i W_i^Q)(X_i W_i^K)^\top)(X_i W_i^V)$$

1065 for all $1 \leq i \leq \frac{N}{M}$. We first compute $X_i W_i^Q, X_i W_i^K, X_i W_i^V$ for all $1 \leq i \leq \frac{N}{M}$. For convenience
 1066 we will denote

$$q_{i,j} = (X_i W_i^Q)[j, :], k_{i,j} = (X_i W_i^K)[j, :], v_{i,j} = (X_i W_i^V)[j, :]$$

1067 such that our goal is to compute $\text{softmax}(q_{i,j}(X_i W_i^K)^\top)(X_i W_i^V)$ for all $1 \leq i \leq \frac{N}{M}, 1 \leq j \leq M$.

1068 Let $B \in \mathbb{R}$ be a large value to be determined. Define

$$u_1, \dots, u_{\frac{N}{M}} \in \{0, -B\}^r$$

1069 such that $\binom{r}{r/2} \geq N/M$ (we only need $r \leq O(\log(N/M))$ by Stirling approximation) and all u_i
 1070 have exactly $r/2$ zeros (if there are more vectors than needed, simply make sure they are distinct),
 1071 and let

$$v_i = B \cdot (1, \dots, 1) + u_i$$

1072 for all $1 \leq i \leq \frac{N}{M}$ such that $\langle u_i, v_i \rangle = 0$ for all i . For any $i \neq j$, notice that there must exist an index
 1073 at which u_i is $-B$ and v_j is B . This is because the set of nonzeros in v_i is the compliment of the set
 1074 of nonzeros in u_i , and the former set must be different from the set of nonzeros in v_j . Now append
 1075 u_i to $q_{i,j}$ to get $q'_{i,j}$ and v_i to $k_{i,j}$ to get $k'_{i,j}$ for all $1 \leq j \leq M$. Set $d' = d + r$. For each pair of $q'_{i,j}$
 1076 and $k'_{i',j'}$:

1077 • If $i = i'$, i.e. q_i and $k_{i'}$ are in the same small transformer instance, then

$$\langle q'_{i,j}, k'_{i',j'} \rangle = \langle q_{i,j}, k_{i',j'} \rangle + \langle u_i, v_{i'} \rangle = \langle q_{i,j}, k_{i',j'} \rangle$$

1078 • If $i \neq i'$, i.e. q_i and $k_{i'}$ are in different small transformer instances, then

$$\langle q'_{i,j}, k'_{i',j'} \rangle = \langle q_{i,j}, k_{i',j'} \rangle + \langle u_i, v_{i'} \rangle \leq \langle q_{i,j}, k_{i',j'} \rangle - B^2.$$

1079 Now we let $Q \in \mathbb{R}^{N \times d'}$ be the matrix of $q'_{i,j}$ and $K \in \mathbb{R}^{N \times d'}$ be the matrix of $k'_{i,j}$ and $V = X_i W_i^V$.
 1080 We set $X = [Q, K, V] \in \mathbb{R}^{N \times (3d')}$ (note that we can always pad zeros to V to make dimensions
 1081 match),

$$W^Q = \begin{pmatrix} I_{d' \times d'} \\ 0_{d' \times d'} \\ 0_{d' \times d'} \end{pmatrix}, W^K = \begin{pmatrix} 0_{d' \times d'} \\ I_{d' \times d'} \\ 0_{d' \times d'} \end{pmatrix}, W^V = \begin{pmatrix} 0_{d' \times d'} \\ 0_{d' \times d'} \\ I_{d' \times d'} \end{pmatrix}$$

1082 such that

$$[Q, K, V]W^Q = Q, [Q, K, V]W^K = K, [Q, K, V]W^V = V.$$

1083 Furthermore, for each query $(q_{i,j}, u_i)$, its inner product between all keys in the same small transformer
 1084 will be preserved, while its inner product between all keys in a different small transformer will be at

most $C_1 - B^2$. Therefore, for each query $q_{i,j}$, the we can calculate the error as:

$$\begin{aligned}
& \left\| \sum_{j'=1}^M \frac{\exp(\langle q_{i,j}, k_{i,j'} \rangle)}{\sum_{\ell=1}^M \exp(\langle q_{i,j}, k_{i,\ell} \rangle)} \cdot v_{i,j'} - \sum_{i'=1}^{N/M} \sum_{j'=1}^M \frac{\exp(\langle q_i, k_{i',j'} \rangle)}{\sum_{\ell=1}^{N/M} \sum_{\ell'=1}^M \exp(\langle q_i, k_{\ell,\ell'} \rangle)} v_{i',j'} \right\|_2 \\
& \leq \left\| \sum_{j'=1}^M \frac{\exp(\langle q_{i,j}, k_{i,j'} \rangle)}{\sum_{\ell=1}^M \exp(\langle q_{i,j}, k_{i,\ell} \rangle)} \cdot v_{i,j'} - \sum_{j'=1}^M \frac{\exp(\langle q_{i,j}, k_{i,j'} \rangle)}{\sum_{\ell=1}^{N/M} \sum_{\ell'=1}^M \exp(\langle q_i, k_{\ell,\ell'} \rangle)} \cdot v_{i,j'} \right\|_2 + \frac{C_2 \exp(C_1 - B)}{\sum_{\ell=1}^M \exp(\langle q_{i,j}, k_{i,\ell} \rangle)} \\
& \leq \left\| \sum_{j'=1}^M \frac{\exp(\langle q_{i,j}, k_{i,j'} \rangle)}{\sum_{\ell=1}^M \exp(\langle q_{i,j}, k_{i,\ell} \rangle)} \cdot v_{i,j'} - \sum_{j'=1}^M \frac{\exp(\langle q_{i,j}, k_{i,j'} \rangle)}{\sum_{\ell=1}^M \exp(\langle q_{i,j}, k_{i,\ell} \rangle) + (N - M) \exp(C_1 - B)} \cdot v_{i,j'} \right\|_2 \\
& + \frac{C_2 \exp(C_1 - B)}{\sum_{\ell=1}^M \exp(\langle q_{i,j}, k_{i,\ell} \rangle)} \\
& \leq \frac{MC_2 \exp(C_1 - B)}{(\sum_{\ell=1}^M \exp(\langle q_{i,j}, k_{i,\ell} \rangle))^2} + \frac{C_2 \exp(C_1 - B)}{\sum_{\ell=1}^M \exp(\langle q_{i,j}, k_{i,\ell} \rangle)}.
\end{aligned}$$

Therefore, we can set $B \leq \text{poly}(N)$ to be sufficiently large such that the error is $O(\frac{1}{2^N})$. \square

D Missing Proofs in Section 5

Theorem D.1 (Theorem 5.1). *For any transformer \mathcal{T} with L layers, H attention heads in each layer, input length N , embedding dimension d , constant-size sliding window, there exists an algorithm that simulates \mathcal{T} with $O(\frac{N}{M} \cdot \frac{HL}{H'L'})$ calls to a transformer oracle with L' layers, H' attention heads in each layer, input length M and embedding dimension $O(\frac{dH'L'}{H})$ with causal masking. This result still holds if we have constant-size attention sink.*

Proof. The proof goes in the same way as Theorem B.5, where we assume without losing of generality that $H = L = H' = L' = 1$. We start with sliding window. Notice that for each query q_i ($i \geq r + 1$) we want to approximate

$$\sum_{j=i-r+1}^i \exp(\langle q_i, k_j \rangle) = \sum_{j=1}^i \exp(\langle q_i, k_j \rangle) - \sum_{j=1}^{i-r} \exp(\langle q_i, k_j \rangle),$$

and

$$\sum_{j=i-r+1}^i \exp(\langle q_i, k_j \rangle) \cdot v_j = \sum_{j=1}^i \exp(\langle q_i, k_j \rangle) \cdot v_j - \sum_{j=1}^{i-r} \exp(\langle q_i, k_j \rangle) \cdot v_j$$

given window size r . Notice that when $i \leq r$, we can approximate $\sum_{j=1}^r \exp(\langle q_i, k_j \rangle)$ and $\sum_{j=1}^r \exp(\langle q_i, k_j \rangle) \cdot v_j$ with 2 oracle calls, as shown in the proof of Theorem B.5.

We will partition $[N]$ into chunks $S'_1 = \{r + 1, \dots, M\}$, $S'_2 = \{M + 1, \dots, 2M - r\}$, \dots of size $M - r$ and approximate the terms above for all $i \in S'_t$ in each chunk using constant many oracle calls, which suffices since r is a constant. Without losing of generality we only prove our claim for S'_1 as the proof can be easily generalized to the remaining sets. Indeed, observe that each of the four terms in the right-hand-side of the equations above can be approximated with one oracle call, where the proof is shown in Claim B.6.

If we add in attention sinks, we simply need to calculate $\sum_{j=1}^s \exp(\langle q_i, k_j \rangle)$ and $\sum_{j=1}^s \exp(\langle q_i, k_j \rangle) \cdot v_j$, which can be done using Claim B.6 again. \square