On The Computational Complexity of Self-Attention

Feyza Duman Keles FD2135@NYU.EDU
Brooklyn, NY 10012

Pruthuvi Mahesakya Wijewardena PWIJEWARDENA@MICROSOFT.COM
Redmond, WA 98052

Chinmay Hegde CHINMAY.H@NYU.EDU
Brooklyn, NY 10012

Editors: Shipra Agrawal and Francesco Orabona

Abstract

Transformer architectures have led to remarkable progress in many state-of-art applications. However, despite their successes, modern transformers rely on the self-attention mechanism, whose time- and space-complexity is quadratic in the length of the input. Several approaches have been proposed to speed up self-attention mechanisms to achieve sub-quadratic running time; however, the large majority of these works are not accompanied by rigorous error guarantees. In this work, we establish lower bounds on the computational complexity of self-attention in a number of scenarios. We prove that the time complexity of self-attention is necessarily quadratic in the input length, unless the Strong Exponential Time Hypothesis (SETH) is false. This argument holds even if the attention computation is performed only approximately, and for a variety of attention mechanisms. As a complement to our lower bounds, we show that it is indeed possible to approximate dot-product self-attention using finite Taylor series in linear-time, at the cost of having an exponential dependence on the polynomial order.

1. Introduction

Motivation. Building upon early successes in natural language processing (Vaswani et al., 2017; Kenton and Toutanova, 2019), transformer models now form the core of virtually every state-of-the-art approach in numerous applications: computer vision and image understanding (Dosovitskiy et al., 2021; Khan et al., 2021), proteomics (Jumper et al., 2021), code synthesis (Chen et al., 2021a), and vision-language models (Radford et al., 2021; Alayrac et al., 2022). At the heart of transformer architectures is the *self-attention* mechanism, which can be viewed as a trainable "layer" that takes in as input a set of tokens (vectors), computes pairwise dot-products between (projected forms of) the tokens, performs a softmax operation to obtain non-negative weights, and produces output tokens using these weights.

Unfortunately, by virtue of its very definition, the standard form of self-attention requires pairwise token operations, and therefore incurs quadratic running time in terms of the number of input tokens. This poses serious computational challenges not just in terms of the training cost of such models, but even just for inference (forward passes) through transformer models. The community has long acknowledged this fact, and numerous approximate forms of self-attention that break this quadratic bottleneck have been proposed. Some approaches advocate reducing complexity through windowing, striding, or some other sparsification of the attention scores (Kitaev et al., 2020; Zaheer et al., 2020). Others leverage ideas from hashing (Kitaev et al., 2020; Choromanski et al., 2021). Yet others

propose kernelizing the attention operation (Katharopoulos et al., 2020; Lu et al., 2021; Chen et al., 2021b).

Empirically, all these methods certainly seem to lead to reduced running times: in some cases, they reduce the costs from quadratic to linear. However, *all* of these methods incur some form of error in the computation (compared to vanilla attention). These errors may have the undesirable (but benign) effect of drop in accuracy, or may have more dramatic effects if fed with an adversarial input. In any case, it would be useful to clearly establish rigorous guarantees on time/accuracy tradeoffs for the above methods, but these are rare. The primary theoretical question that we ask is as follows:

What are the fundamental computational tradeoffs involved in self-attention?

Our contributions. In this paper, we pursue a complementary path from most previously published work in this area. Somewhat surprisingly, we are able to establish (conditional) *quadratic lower bounds* on the running time of self-attention in a large variety of settings. This quadratic barrier holds *even* if we relax the self-attention operation and allow for additive, or multiplicative, errors in the computation. We also prove quadratic — specifically, rectangular — lower bounds even if we allow for windowing or striding. Finally, this holds even if we use kernelization (with radial basis function kernels), which to our current knowledge, achieves the Pareto time/accuracy frontier in terms of empirical performance among all fast algorithms for self-attention computation (Lu et al., 2021). In Table 1, we summarize hardness results where checkmarks indicate the proven complexities in this paper for different types of self-attention and calculation types.

Our results demonstrate that there may be a fundamental "no free lunch" phenomenon sitting here: it seems unlikely that we can get (provably) sub-quadratic algorithms for self-attention that are also (provably) near-accurate for all inputs.

Finally, while our primary contributions in this paper are mainly from the perspective of lower bounds, we also provide some upper bounds. Specifically, we show that a finite Taylor series approximation of the softmax function lends itself to an approximate form of self-attention that can be computed in linear time. However, a caveat of this result is that the running time now scales exponentially in the order of the Taylor polynomial.

Techniques. Our proofs are rather intuitive and are based on careful reductions from the Strong Exponential Time Hypothesis (SETH). SETH-based lower bounds have attracted recent (but growing) attention from the complexity theory community, and has been used to prove hardness results for edit distance (Backurs and Indyk, 2015), Frechet distance (Bringmann, 2014), dynamic programming (Bringmann and Künnemann, 2015), among many others (Bringmann, 2021). For machine learning problems, such lower bounds are less common; still, quadratic-time barriers based on SETH have been proved for kernel PCA and backpropagation through dense networks (Backurs et al., 2017), as well as nearest neighbors (Rubinstein, 2018). Our results can be viewed as an addition to this body of work.

A direct reduction from SETH is cumbersome. So instead, mirroring (Backurs and Indyk, 2015), we derive reductions from the Orthogonal Vectors Problem (OVP), which is known to require almost-quadratic time assuming SETH. As intermediate waypoints we visit two adaptations of OVP: the Thresholded Vectors Product Problem (TVPP), and the Bichromatic Hamming Close Pair (BHCP) problem. Reductions between these problems require construction of several "vector gadgets", and form the bulk of the technical difficulty of our proofs. Another subtlety lies in identifying the correct temperature scaling in the softmax in order to achieve the reductions. See Section 4 for details.

Table 1: Summary of our hardness results on self-attention

Self-Attention	Calculation Type		
	Exact	Element-wise Multiplicative Approx.	Element-wise Additive Approx.
Exponential Dot-Product	✓	✓	✓
Softmax Dot-Product	✓	✓	$\mu = O(n^{-d})$
Window Sliding	✓	✓	✓
Exponential L2-Norm	✓	✓	$\mu = O(n^{-d})$

2. Related work

Attention mechanisms and transformers. Ever since the seminal work of (Vaswani et al., 2017), transformer architectures (with self-attention layers as their primary building blocks) have become the cornerstone of state-of-the-art machine learning models. Therefore, a firm theoretical understanding about the statistical and computational tradeoffs involved in transformer-based models is of considerable interest. Transformers have already been shown to exhibit the universal approximation property (Yun et al., 2019), but lose expressivity unless the self-attention mechanism is accompanied with skip connections (Dong et al., 2021). Self-attention also exhibits undesirable Lipschitz continuity properties, but this can be fixed by pursuing kernel-like alternatives (Kim et al., 2021).

Speeding up self-attention. Our focus in this paper is on the running time of self-attention computation. It has been well-established that the (standard) definition of self-attention takes in as input a length-n sequence of tokens of size d, and requires $O(dn^2)$ time to compute the output. The quadratic dependence on n poses a challenge for very long input sequences, both from the training and testing perspectives. Therefore, several sub-quadratic methods for evaluating attention layers have been proposed. Approaches (such as the Reformer (Kitaev et al., 2020), Big Bird (Zaheer et al., 2020), Linformer (Wang et al., 2020), Longformer (Beltagy et al., 2020), or routing transformers (Roy et al., 2021)) use some combination of hashing, sparsification, or low-rank approximation to speed up the computation of the attention scores. Other approaches involve replacing the softmax-based attention with kernel approximations; cf. the work of (Katharopoulos et al., 2020), or the Nyströmformer (Xiong et al., 2021). More recent works such as the Performer (Choromanski et al., 2021), Slim (Likhosherstov et al., 2021), or RFA (Peng et al., 2021) approximate the attention computation using random projections. Methods such as SOFT (Lu et al., 2021) or the Skyformer (Chen et al., 2021b) propose to replace softmax operations with Gaussian kernels that can then be quickly evaluated.

Despite the large number (and diversity) of interesting algorithmic ideas involved in the above efforts, the vast majority of these works only focus on improvement in *running time*; but few (if any) theoretically characterize the *error* incurred by their proposed methods. Can there ever be a method that is both fast (i.e., provably with sub-quadratic running time) as well as near-accurate (i.e., with provably small additive or multiplicative error)? Our results show that this is unlikely to be the case.

Fine-grained complexity. Classical complexity theory has primarily focused on distinguishing between problems with efficient (polynomial-time) solutions versus those who don't. However, a different (and finer) picture has begun to emerge over the last decade. In particular, the focus has shifted towards precisely pinning down the *exponent*, c, of a problem that can be solved in

polynomial time $\tilde{O}(n^c)$. Many of these results are conditional, and rely on reductions from popular (but plausible) conjectures such as the Strong Exponential Time Hypothesis (SETH) (Impagliazzo and Paturi, 2001), (Impagliazzo et al., 2001). See the relevant surveys (Indyk, 2017), (Rubinstein and Williams, 2019), and (Bringmann, 2021) for a more concrete overview of the field. In particular, this approach has been shown to provide conditional lower bounds on well-known problems such as edit distance (Backurs and Indyk, 2015), Frechet distance (Bringmann, 2014), dynamic time warping (Bringmann and Künnemann, 2015), longest common subsequence (LCS) (Abboud et al., 2015), Hausdorff distance (Bringmann and Nusser, 2021), and string matching (Abboud et al., 2018). In the context of machine learning and massive data analysis, reductions from SETH have been fruitfully applied to problems such as clustering (Abboud et al., 2019), kernel PCA (Backurs et al., 2017), and approximate nearest neighbors (Rubinstein, 2018).

3. Notations and Preliminaries

An ordered finite set of n vectors in $\mathbb{R}^d$ will be denoted as an $n \times d$ matrix whose rows denote the elements of the set respectively. We use upper case characters to denote both vector sets and matrices depending on the context. We use A_i to denote the i^{th} row of matrix A, or the i^{th} element of ordered set A. We use A_{ij} to denote the element at the i^{th} row and j^{th} column of matrix A. For a positive integer $n \in \mathbb{Z}^+$, $[n]$ denotes the set of all positive integers up to n.

3.1. Background on Self-Attention

The well-established Transformer model (Vaswani et al., 2017) is based on the multi-head attention mechanism, comprising several self-attention layers running in parallel. The canonical choice of self-attention is the *softmax dot-product self-attention*, defined as follows. For a given set of inputs written as $X \in \mathbb{R}^{n \times d}$ and trainable parameter matrices $W_q \in \mathbb{R}^{d \times d_q}, W_k \in \mathbb{R}^{d \times d_k}, W_v \in \mathbb{R}^{d \times d_v}$, this operation first calculates the *query* ($Q = XW_q$), *key* ($K = XW_k$), and *value* ($V = XW_v$) matrices respectively. We assume that $d_q = d_k$. The size of Q and K is then $n \times d_k$, while the size of V is $n \times d_v$. The softmax dot-product self-attention operation is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{1}$$

Let us consider the case where n is very large compared to d_k, d_v. By very virtue of its definition, we might expect to incur $O(n^2)$ time to compute this self-attention operation: 1) calculation of $S = QK^T/\sqrt{d_k}$ takes $O(n^2 d_k)$, 2) exponentiation and calculation of row sum of S takes $O(n^2)$ time, 3) division of each element of S with the corresponding row sum takes $O(n^2)$, and 4) multiplication of softmax(QK^T) and V takes $O(n^2 d_v)$ time. Therefore, the computational complexity of this naive approach to compute self-attention scales quadratically in n.

A generalized form of self-attention. While Eq. 1 is the typical way to define self-attention, we also consider a more general form. Let $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a function that takes two vectors in $\mathbb{R}^d$ as input. Then, the self-attention score matrix S is defined as $S_{ij} = f(Q_i, K_j)$ for all $i, j \in [n]$. Also, let $h : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$ be some kind of normalization function. Then a more abstract definition of self-attention can be expressed as:

$$\text{Attention}(Q, K, V) = h(S) \cdot V. \tag{2}$$

In particular, for the softmax dot-product self-attention, the function f is the dot-product of given vectors with normalization factor $\sqrt{d_k}$, and h is the row-wise softmax function.

In this paper, we show that there is no (provably) better algorithm than the naive $O(n^2)$ approach for calculating softmax dot-product self-attention, given hardness of SETH. We will also give similar lower bounds for various approximate forms of self-attention. Finally, we will investigate the computational complexity of generalized self-attention for more general forms of f.

3.2. SETH and OVP

Despite a remarkable amount of algorithmic effort on Boolean satisfiability (SAT) and related problems, to date no one has invented an algorithm with faster-than-exponential ($O(2^n)$) running time; indeed, there is no polynomial-time algorithm for SAT unless $P = NP$. The Strong Exponential Time Hypothesis (SETH) (Impagliazzo and Paturi, 2001; Impagliazzo et al., 2001) can be viewed as an strengthening of this statement: for every ϵ, there is no *sub-exponential* ($O(2^{n(1-\epsilon)})$) time algorithm that solves SAT.

As discussed above in Section 2, over the last decade SETH has been used to provide fine-grained lower bounds for several polynomial-time problems. Many of these results use reduction from an intermediate problem, given as follows.

Definition 1 (Orthogonal Vectors Problem (OVP)) *Two sets with cardinality $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ are given, where $a_i, b_i \in \{0, 1\}^d$ are binary vectors for all $i \in [n]$. The problem decides if there exists at least one pair of vectors $a \in A$ and $b \in B$ such that $a^T b = 0$.*

Previous work has that for all $\epsilon > 0$, there is no $O(n^{2-\epsilon})$ algorithm solves OVP for $d = \omega(\log n)$ unless SETH is false (Williams, 2005). In other words, for any $\epsilon > 0$, problem needs at least $O(n^{2-\epsilon})$ time for $d = \omega(\log n)$. In this work, we will primarily give lower bounds to the computational complexity of self-attention mechanism by showing reductions from OVP.

4. Hardness of Computing Self-Attention

4.1. Adaptations of OVP

Based on the definition of generalized self-attention (Eq. 2), we focus on two main forms of self-attention: (a) dot-product self-attention with $f(x, y) = e^{Cx^T y}$, (b) ℓ_2 self-attention (or RBF kernel self-attention) with $f(x, y) = e^{-C.\|x-y\|_2^2}$ for some temperature/scale parameter C to be specified later, where x, y are row vectors from matrices Q, K respectively. We provide hardness guarantees for a variety of self-attention mechanisms built on top of these two types of self-attention. We achieve this by deriving reductions from two fundamental problems stated in Definitions 2 and 3.

Definition 2 (Threshold Vectors Product Problem (TVPP)) *Two sets with equal cardinality $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ are given, where $a_i, b_i \in \{0, 1\}^d$ are binary vectors for all $i \in [n]$. The problem decides if there exists at least one pair $a \in A$ and $b \in B$ such that $a^T b \geq t$ for a given $t \in [n]$.*

Definition 3 (Bichromatic Hamming Close Pair Problem (BHCP)) *Two sets with equal cardinality $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ are given, where $a_i, b_i \in \{0, 1\}^d$ are binary vectors for all $i \in [n]$. The problem decides if there exists at least one pair of vectors $a \in A$ and $b \in B$ such that $\|a - b\|_2 < t$.*

Both TVPP and BHCP can be shown to be SETH-hard by showing reductions from OVP (Definition 1) as shown in Lemmas 1, 2, and 3. The hardness of BHCP is an established result (Backurs et al., 2017), but we provide an improved reduction from OVP to BHCP via a new problem: Bichromatic Hamming Far Pair problem (BHFP). In contrast to established reductions from OVP, we get rid of additional factors of d (the dimension of binary vectors) that incur during the process.

Definition 4 (Bichromatic Hamming Far Pair Problem (BHFP)) *Two sets with equivalent cardinality $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ are given, where $a_i, b_i \in \{0,1\}^d$ are binary vectors for all $i \in [n]$. The problem decides if there exists at least a pair of vectors $a \in A$ and $b \in B$ such that $\|a - b\|_2 \geq t$.*

Before discussing the hardness of computing self-attention, we state the hardness guarantees of TVPP, BHFP, and BHCP formally, with proofs deferred to Appendix A.

Lemma 1 *Assume SETH. Then for any $\epsilon > 0$, the computational complexity of TVPP is $\Omega(n^{2-\epsilon})$ for $d = \omega(\log n)$.*

Lemma 2 *Assume SETH. Then for any $\epsilon > 0$, the computational complexity of BHFP is $\Omega(n^{2-\epsilon})$ for $d = \omega(\log n)$.*

Lemma 3 *Assume SETH. Then for any $\epsilon > 0$, the computational complexity of BHCP is $\Omega(n^{2-\epsilon})$ for $d = \omega(\log n)$.*

Below, we show a series of reductions from TVPP and BHCP to several well-studied self-attention mechanisms. We mainly modify the functions $f(.)$ and $h(.)$ (see Eq. 2) to reflect each attention mechanism in our arguments. We ignore the scaling factor $1/\sqrt{d_k}$ when computing the function $f(.)$ in dot-product self-attention for easier exposition as we only require to scale every element of Q or K by $1/\sqrt{d_k}$ which takes $O(nd_k)$ time. Also, we note that several approaches for efficient transformers were developed with the particular case of $Q = K$ (Kitaev et al., 2020), and our hardness results are valid for this specific instance where $Q = K$ and by direct reduction, and hardness guarantees hold for any universal (Q, K) as well. Also, all the process is still valid for multi-head self attention which is proved in Appendix F.

4.2. Vector Gadgets for Reductions

Our arguments follow by showing reductions from TVPP and BHCP instances to self-attention instances. We define a set of vector gadgets that convert input to TVPP and BHCP into inputs to self-attention functions in the following manner.

TVPP vector gadgets. Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ given in TVPP, where $a_i, b_i \in \{0,1\}^d$ are binary vectors for all $i \in [n]$. We construct our vector gadgets in the following way. First, create the matrix $Q \in \mathbb{R}^{2n \times d}$ with its rows as $Q_i = a_i$ for all $i \in [n]$, and $Q_{n+j} = Cb_j$ for all $j \in [n]$, where $C > 0$ is a parameter which we will define in the corresponding reductions. Then we create the matrix $V \in \mathbb{R}^{2n \times 1}$ by setting first n elements to 0 and the second n elements to 1.

BHCP vector gadgets. Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ given in BHCP, where $a_i, b_i \in \{0,1\}^d$ are binary vectors for all $i \in [n]$. We construct our vector gadgets in the following way. First, create the matrix $Q \in \mathbb{R}^{2n \times d}$ with its rows as $Q_i = a_i$ for all $i \in [n]$, and $Q_{n+j} = b_j$ for all $j \in [n]$. Then we create the matrix $V \in \mathbb{R}^{2n \times 1}$ by setting first n elements to 0 and the second n elements to 1.

The entire construction process (including multiplying each a_i, b_i by C) of vector gadgets takes $O(nd)$ time. We use the above gadgets in the following reductions.

4.3. Hardness of Dot-Product Self-Attention

We begin with hardness results of dot-product self-attention (without softmax normalization) as a warm-up. In Theorem 4 we show that exact self-attention, as well as element-wise multiplicative- and additive-error approximations of self-attention, all require quadratic time, conditioned on SETH.

Theorem 4 *Assume SETH. For any $i, j \in [n]$, let $S_{ij} = f(Q_i, Q_j) = e^{Q_i^T Q_j}$ and for any matrix $M \in \mathbb{R}^{n \times n}$, let $h(M) = M$. Let $Y = h(S) \cdot V \in \mathbb{R}^{n \times d_v}$ be a self-attention mechanism. Provided $d_q = \omega(\log n)$, for any $\epsilon > 0$, computing a matrix $\hat{Y} \in \mathbb{R}^{n \times d_v}$ that satisfies any of the following conditions requires $\Omega(n^{2-\epsilon})$ time.*

1. *$\hat{Y} = Y$ (exact computation).*
2. *$|\hat{Y}_{ij} - Y_{ij}| \leq \mu |Y_{ij}|$ for all $i \in [n]$ and $j \in [d_v]$ where $0 \leq \mu < 1$ (multiplicative approximation).*
3. *$|\hat{Y}_{ij} - Y_{ij}| \leq \mu$ for all $i \in [n]$ and $j \in [d_v]$ where $0 \leq \mu$ (additive approximation).*

Proof Suppose that the matrices Q and V are constructed as TVPP vector gadgets described above in Section 4.2. With this, we have $Y = h(S) \cdot V \in \mathbb{R}^{n \times 1}$. Since Y is a vector, we slightly abuse notation and define Y_i as the i^{th} element of Y. Now consider the first n elements of Y. Since $Y = SV$, we have that

$$Y_i = \sum_{j=1}^{n} e^{C a_i^T b_j} \text{ for any } i \in [n].$$

Now we check the magnitude of $Y_i, i \in [n]$ in order to distinguish between true and false cases in TVPP. It takes $O(n)$ time to check each $Y_i, i \in [n]$. First we focus on the exact computation. Consider the following two cases.

Case 1. There are no pairs $a \in A$ and $b \in B$ with $a^T b \geq t$, that is for all $i, j \in [n]$, we have $a_i^T b_j \leq t - 1$, and $e^{C a_i^T b_j} \leq e^{C(t-1)}$. Then for all $l \in [n]$, $Y_l \leq n e^{C(t-1)} := \delta$.

Case 2. There is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, that is for some $i, j \in [n]$, we have $a_i^T b_j \geq t$, and $e^{C a_i^T b_j} \geq e^{Ct}$. Thus for some $l \in [n]$, we have $Y_l \geq e^{Ct} := \Delta$

To distinguish between the two cases, it is sufficient to have $\Delta > \delta$. But this holds when $C = 2 \log n$.

Now let us consider multiplicative approximation error. With a μ-multiplicative factor, if there are no pairs $a \in A$ and $b \in B$ with $a^T b \geq t$, then we have for all $l \in [n], \hat{Y}_l \leq (1+\mu) Y_l \leq (1+\mu) n e^{C(t-1)} := \hat{\delta}$. On the other hand, if there is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, then for some $l \in [n]$, we have $\hat{Y}_l \geq (1-\mu) Y_l \geq (1-\mu) e^{Ct} := \hat{\Delta}$. In order to distinguish between two cases, it is sufficient to have $\hat{\Delta} > \hat{\delta}$ and this inequality holds with $C = 2 \log(\frac{1+\mu}{1-\mu} n)$.

Finally we look at additive approximation error. With a μ-additive factor, if there are no pairs $a \in A$ and $b \in B$ with $a^T b \geq t$, then we have for all $l \in [n], \hat{Y}_l \leq Y_l + \mu \leq n e^{C(t-1)} + \mu := \hat{\delta}$. On the other hand, if there is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, then for some $l \in [n]$, we have

$\hat{Y}_l \geq Y_l - \mu \geq e^{Ct} - \mu := \hat{\Delta}$. In order to distinguish between two cases, it is sufficient to have $\hat{\Delta} > \hat{\delta}$ and this inequality holds with $C = 2\log(n + 2\mu)$.

Thus, if there is an algorithm for computing self-attention up to an element-wise multiplicative or additive error μ that runs in $O(n^{2-\epsilon})$ time, this entire process takes at most $O(n + nd + n^{2-\epsilon}) = O(n^{2-\epsilon})$ as long as $d = o(n^{1-\epsilon})$. Therefore, this algorithm decides if there exists at least a pair of vectors $a \in A$ and $b \in B$ such that $a^T b \geq t$ in $O(n^{2-\epsilon})$ time, which contradicts the hardness result of TVPP (Lemma 1). This completes the proof. ■

The above proof is for computing the output of self-attention. As an easy consequence, we can also show that computing the self-attention *score matrix*, S, with either exact or element-wise multiplicative/additive error requires quadratic time, conditioned on SETH. The argument follows a similar proof as Theorem 4, and is given in Appendix C in detail.

4.4. Hardness of Softmax Dot-Product Self-Attention

Now we establish hardness guarantees for computing standard softmax dot-product self-attention. The difference from vanilla self-attention is that the function $h(.)$ now normalizes input rows. Discussion of additive approximation for this part is given in Appendix G.

Theorem 5 *Assume SETH. For any $i, j \in [n]$, let $S_{ij} = f(Q_i, Q_j) = e^{Q_i^T Q_j}$ and for any matrix $M \in \mathbb{R}^{n\times n}$, let $h(M) \in \mathbb{R}^{n\times n}$ be the matrix where for all $i, j \in [n]$, $\{h(M)\}_{ij} = \frac{M_{ij}}{\sum_{k=1}^n M_{ik}}$. Let $Y = h(S) \cdot V \in \mathbb{R}^{n\times d_v}$ be a self-attention. Then provided $d_q = \omega(\log n)$, for any $\epsilon > 0$, computing a matrix $\hat{Y} \in \mathbb{R}^{n\times d_v}$ that satisfies any of the following conditions requires $\Omega(n^{2-\epsilon})$ time.*
1. $\hat{Y} = Y$ (exact).
2. $|\hat{Y}_{ij} - Y_{ij}| \leq \mu |Y_{ij}|$ for all $i \in [n]$ and $j \in [d_v]$ where $0 \leq \mu < 1$ (multiplicative approximation).

Proof The proof is technically similar to the one above. Suppose that the matrices Q and V are constructed according to TVPP vector gadgets described in Section 4.2. With this we have $Y = h(S) \cdot V \in \mathbb{R}^{n\times 1}$. Consider the first n elements of Y. Since $h(.)$ act as the row-wise softmax function, we have

$$Y_i = \sum_{j=n+1}^{2n} S_{ij} = \sum_{j=1}^{n} \frac{e^{Ca_i^T b_j}}{\sum_{k=1}^n e^{a_i^T a_k} + \sum_{k=1}^n e^{Ca_i^T b_k}} = \frac{\sum_{j=1}^n e^{Ca_i^T b_j}}{\sum_{j=1}^n e^{a_i^T a_j} + \sum_{j=1}^n e^{Ca_i^T b_j}}.$$

Again, first we focus on exact computation and consider two cases.

Case 1. There are no pairs $a \in A$ and $b \in B$ with $a^T b \geq t$, that is for all $i, j \in [n]$, we have $a_i^T b_j \leq t - 1$, and $e^{Ca_i^T b_j} \leq e^{C(t-1)}$. For a function $\frac{x}{x+y}$, the maximum value is achieved at maximum x and minimum y values. Thus, for all $l \in [n]$, $Y_l \leq \frac{ne^{C(t-1)}}{ne^{C(t-1)}+n} = \frac{e^{C(t-1)}}{e^{C(t-1)}+1} := \delta$.

Case 2. There is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, that is for some $i, j \in [n]$, we have $a_i^T b_j \geq t$. Then the row sum corresponding to that i, j pair is $\sum_{j=1}^n e^{Ca_i^T b_j} \geq e^{Ct} + (n-1)e^0 = e^{Ct} + (n-1)$. For a function $\frac{x}{x+y}$, the minimum value is achieved at minimum x and maximum y values. Thus, for some $l \in [n]$, we have $Y_l \geq \frac{e^{Ct}+(n-1)}{e^{Ct}+(n-1)+ne^d} := \Delta$ and $e^{Ca_i^T b_j} \geq e^{Ct}$.

In order to distinguish between two cases, it is sufficient to have $\Delta > \delta$ which means we require $[e^{Ct} + (n-1)] > ne^d[e^{C(t-1)}]$. This holds with $C = \log n + d$.

Next, consider multiplicative error approximation. Select same TVPP vector gadgets except this time the matrix $V \in \mathbb{R}^{2n\times 1}$ is set first n elements to 1 and the second n elements to 0. Since $|\hat{Y}_l - Y_l| \leq \mu|Y_l|$ for all $i \in [2n]$, we have $(1-\mu)Y_i \leq \hat{Y}_i \leq (1+\mu)Y_i$. Now consider the values of $\hat{Y}_i, i \in [n]$ in the following two cases.

Case 1. There are no pairs $a \in A$ and $b \in B$ with $a^T b \geq t$, that is for all $i, j \in [n]$, we have $a_i^T b_j \leq t-1$. This means that $\sum_{j=1}^{n} e^{Ca_i^T b_j} \leq ne^{C(t-1)}$. For a function $\frac{x}{x+y}$, the minimum value is achieved at the minimum x and maximum y values. Thus, for all $l \in [n]$, $Y_l \geq \frac{n}{n+ne^{C(t-1)}} = \frac{1}{e^{C(t-1)}+1}$ which means that for all $l \in [n]$, we have $\hat{Y}_l \geq (1-\mu)Y_l \geq (1-\mu)\frac{1}{e^{C(t-1)}+1} := \Delta$.

Case 2. There is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, that is for some $i, j \in [n]$, we have $a_i^T b_j \geq t$. Then the row sum corresponding to that i, j pair is $\sum_{j=1}^{n} e^{Ca_i^T b_j} \geq e^{Ct} + (n-1)e^0 = e^{Ct} + (n-1)$. For a function $\frac{x}{x+y}$, the maximum is achieved at the maximum x and minimum y values. Thus, for some $l \in [n]$, we have $Y_l \leq \frac{ne^d}{e^{Ct}+(n-1)+ne^d}$ which means that for some $l \in [n]$, we have $\hat{Y}_l \leq (1+\mu)Y_l \leq (1+\mu)\frac{ne^d}{e^{Ct}+(n-1)+ne^d} := \delta$.

In order to distinguish between the two cases, it is sufficient to have $\Delta > \delta$. This holds with $C = \log(\frac{2(1+\mu)}{1-\mu}n) + d$. Thus, if there is an algorithm for computing self-attention up to an element-wise multiplicative error μ that runs in $O(n^{2-\epsilon})$ time, this entire process takes at most $O(n + nd + n^{2-\epsilon}) = O(n^{2-\epsilon})$ as long as $d = o(n^{1-\epsilon})$, and this algorithm decides if there exists a pair of vectors $a \in A, b \in B$ such that $a^T b \geq t$ in $O(n^{2-\epsilon})$ time, contradicting TVPP hardness. ■

Remark. For the multiplicative error approximations, we can set μ to a value arbitrarily close to 1 with a dependence on n, i.e. $\mu = 1 - \Theta(1/n^x)$ for a sufficiently large(constant) order x. For example, with $x = 2$, we require $C > 2\log(\frac{2-\Theta(1/n^2)}{\Theta(1/n^2)}n) \approx 2[\log 2 + 3\log(\Theta(n))]$. Therefore, to distinguish between two cases with a μ multiplicative error approximation with $\mu = 1 - \Theta(1/n^2)$, we only require $C = O(\log(n))$. We discuss this matter in more detail in the Appendix H.

4.5. Hardness of Sliding Window Dot-Product Self-Attention

We now consider well-known less-expensive alternatives to standard self-attention. A popular example is sliding window self-attention (Beltagy et al., 2020). Here, we evaluate an element of the score matrix S_{ij} only if the difference between i and j is within a fixed window size w; else we set it to zero. This reduces the running time to $O(nw)$, which can be small if the window size is small. However, we show that such a *rectangular* complexity is unavoidable.

Theorem 6 *Assume SETH. Let $f : \{\mathbb{R}^d, \mathbb{R}^d\} \longrightarrow \mathbb{R}$ as $f(x, y) = e^{x^T y}$. For set Q of vectors $Q_1, \ldots, Q_n$, we define the matrix $S \in \mathbb{R}^{n\times n}$ as*

$$(S)_{ij} = \begin{cases} f(Q_i, Q_j) & |i-j| \leq w/2 \\ 0 & \textit{otherwise} \end{cases}$$

Also for any matrix $M \in \mathbb{R}^{n\times n}$, let $h(M) = M$. Let $Y = h(S) \cdot V \in \mathbb{R}^{n\times d_v}$ be a self-attention. Then for any $\epsilon > 0$, computing a matrix $\hat{Y} \in \mathbb{R}^{n\times d_v}$ that satisfies any of the following conditions requires $\Omega(nw^{1-\epsilon})$ time when $d_q = \omega(\log w)$ and $w = \omega(d_q)$.
1. $\hat{Y} = Y$ (exact).

2. $|\hat{Y}_{ij} - Y_{ij}| \leq \mu |Y_{ij}|$ *for all* $i \in [n]$ *and* $j \in [d_v]$ *where* $0 \leq \mu < 1$ *(multiplicative approximation).*
3. $|\hat{Y}_{ij} - Y_{ij}| \leq \mu$ *for all* $i \in [n]$ *and* $j \in [d_v]$ *where* $0 \leq \mu$ *(additive approximation).*

Proof See Appendix B for a full proof. First, a TVPP problem with sets size $k = \sqrt{nw}$ are constructed. Then sliding-window self attention is computed for all the points in order, so that for each TVPP problem, all the pairs are in the window. An appropriate value matrix V is constructed for selection of this pairs. If the overall self attention can be calculated in $O((nw)^{1-\epsilon})$ time, then TVPP problem can be solved in $O(k^{2-\epsilon})$ which contradicts Lemma 1. ■

4.6. Hardness of ℓ_2- Self-Attention

We now establish hardness guarantees for computing ℓ_2-self-attention, which replaces a softmax with an RBF kernel and is the core idea underlying both SOFT (Lu et al., 2021) and Skyformer (Chen et al., 2021b), the current state-of-the-art in fast self-attention operations. In our argument, we adopt a similar proof technique employed in (Backurs et al., 2017), who establish quadratic hardness of kernel PCA assuming SETH. However, our proof involves a different chain of reductions: OVP $\rightarrow$ BHFP $\rightarrow$ BHCP $\rightarrow$ kernel computation. Discussion of additivite approximation for this part is given in Appendix G.

Theorem 7 *Assume SETH. For any* $i, j \in [n]$*, let* $S_{ij} = f(Q_i, Q_j) = e^{C.\|Q_i - Q_j\|_2^2}$ *where* C *is a parameter and for any matrix* $M \in \mathbb{R}^{n \times n}$*, let* $h(M) = M$*. Let* $Y = h(S).V \in \mathbb{R}^{n \times d_v}$ *be a self-attention. Then for any* $\epsilon > 0$*, computing a matrix* $\hat{Y} \in \mathbb{R}^{n \times d_v}$ *that satisfies any of the following conditions requires* $\Omega(n^{2-\epsilon})$ *time when* $d_q = \omega(\log n)$*.*
1. $\hat{Y} = Y$ *(exact).*
2. $|\hat{Y}_{ij} - Y_{ij}| \leq \mu |Y_{ij}|$ *for all* $i \in [n]$ *and* $j \in [d_v]$ *where* $0 \leq \mu < 1$ *(multiplicative approximation).*

Proof (Sketch). The Q and V matrices and constructed according to BHCP vector gadgets described in Section 4.2. With these, we define $Y = h(S) \cdot V \in \mathbb{R}^{n \times 1}$. Considering the first n elements of Y and a suitable selection of C, our goal is to make the two cases (whether there is a solution for BHCP or not) distinguishable. Thus, if there is an algorithm for computing self-attention up to an element-wise multiplicative error μ that runs in $O(n^{2-\epsilon})$ time, this algorithm decides if there exists a solution or not for BHCP in $O(n^{2-\epsilon})$ time, which contradicts BHCP hardness (Lemma 3). ■

5. Polynomial Approximations of Self-Attention

We have shown (conditional) hardness results for a number of well-known self-attention mechanisms. We conclude with some upper bounds. Given the query Q, key K, and value V matrices, we show that when $f(Q_i, K_j)$ is a polynomial of order p (p is an integer ≥ 0) of $Q_i^T K_j$ and $h(.)$ is either the identity function or row-wise normalization, one can compute self-attention in linear time. However, now the time complexity scales exponentially with p. As a special case of this, we show that one can approximate dot-product softmax self-attention in linear time in n by using finite Taylor series approximation: $e^x \approx \sum_{k=0}^{p} \frac{x^k}{k!}$. In what follows we use the fact that $d_q = d_k$ and use d_q in both places. Recall the dimensions of matrices $Q \in \mathbb{R}^{n \times d_q}, K \in \mathbb{R}^{n \times d_q}, V \in \mathbb{R}^{n \times d_v}$.

Lemma 8 shows that the product SV(without the row-wise normalization) can be computed in linear time n. Lemma 9 shows that the denominator for row-wise normalization can be computed in

linear time in n. Once we have the denominator for each row, $h(S) \cdot V$ can be computed by dividing each row of SV by the corresponding denominator.

Lemma 8 *Let p be an integer ≥ 0 and let S_{ij} be $C \cdot (Q_i^T K_j)^p$ where $i, j \in [n]$ and C is a constant. Then SV can be computed in $O(nd_q^p d_v)$ time.*

Lemma 9 *Let p be an integer ≥ 0. Then for all $i \in [n]$, $\sum_{\hat{j}=1}^{n} C \cdot (Q_i^T K_{\hat{j}})^p, \hat{j} \in [n]$ where C is a constant can be computed in $O(nd_q^p)$ time.*

Proof sketch of Lemma 8 and 9. When $f(Q_i, V_j) = Q_i^T V_j$, we directly multiply matrices K^T and V to obtain $K^T V$ in $O(d_q n d_v)$ time, then Q and $K^T V$ in $O(nd_q d_v)$ gives the desired matrix SV in $O(n)$ time. The sum of a row of S can be simply computed by storing the sum of vectors $K_j, j \in [n]$ in memory in $O(nd_q)$ time and reusing this for each row i to compute $\sum_{j=1}^{n} Q_i^T K_j = Q_i^T \sum_{j=1}^{n} K_j$ in $O(nd_q)$ time. This idea can be extended to $(Q_i^T K_j)^p$.

Theorem 10 *Let p be an integer ≥ 0. If $S_{ij} = f(Q_i, K_j)$ is a polynomial function of order p of $Q_i^T K_j$ and $h(.)$ performs row-wise normalization, then $h(S) \cdot V$ can be computed in $O(nd_q^p d_v)$ time.*

Proof The result is implied by Lemma 8 and Lemma 9. Let $x_{ij} = Q_i^T K_j; i, j \in [n]$. Define the polynomial function of order p as $\sum_{z=0}^{p} c_z x_{ij}^z$ where $c_z, z \in \{0, \ldots, p\}$ are constants. Let $S^{(1)}, \ldots, S^{(p)} \in \mathbb{R}^{n \times n}$ where $S_{ij}^{(z)} = c_z x_{ij}^z$. Now we can write $S_{ij} = \sum_{z=0}^{p} S_{ij}^{(z)}$, thus $SV = \sum_{z=0}^{p} S_{ij}^{(z)} V$. By Lemma 8, each term of this summation can be computed in $O(nd_q^z d_v)$ time. Therefore the overall time complexity of computing SV is $O(nd_q^p d_v)$ (considering the largest exponent of d_q is when $z = p$).

Let s_i be the sum of the elements of i^{th} row of S. What remains is computing $h(S).V$. This can be computed indirectly by first computing SV and the dividing each row i of SV by s_i. If we have precomputed each $s_i, i \in [n]$, then this process takes $O(nd_v)$ time, since we are dividing each element of $SV (\in \mathbb{R}^{n \times d_v})$ by a scaler. What remains to show is that computing $s_i, \forall i \in [n]$ takes $O(npd_q^p)$ time. Observe that

$$s_i = \sum_{\hat{j}=1}^{n} S_{ij} = \sum_{\hat{j}=1}^{n} \sum_{z=0}^{p} S_{i\hat{j}}^{(z)} = \sum_{z=0}^{p} \sum_{\hat{j}=1}^{n} S_{i\hat{j}}^{(z)} = \sum_{z=0}^{p} \sum_{\hat{j}=1}^{n} c_z x_{i\hat{j}}^z$$

From Lemma 9, each term of the outer summation over z can be computed $O(nd_q^z)$ time for all $i \in [n]$ and overall time complexity is $O(nd_q^p)$(taking the largest exponent of d_q similarly). ■

Corollary 11 *Let p be an non-negative integer. The p^{th} order polynomial approximation of dot-product softmax self-attention using matrices Q, K, V with finite Taylor series can be computed in $O(nd_q^p d_v)$ time.*

Proof The result follows from replacing constants $c_z = \frac{1}{z!}$ for $z = 0, \ldots, p$ in Theorem 10. ■

6. Conclusions

In this paper we investigate fundamental bounds on the computational complexity of self-attention. We examine numerous state-of-the-art self-attention mechanisms, and prove quadratic (or rectangular) lower bounds assuming the Strong Exponential Time Hypothesis (SETH). Even though a large number of recent works have proposed fast approximations to self-attention, our results imply that it may be difficult to both overcome the quadratic runtime barrier while still retaining high accuracy. On the positive side, we show that linear-time computation is possible if we choose the score computation function in the form of a polynomial.

Our work leaves open several directions. At a high level our theorems establish a result between 'exponential' and 'polynomial' forms of self-attention, but having a clearer picture of the landscape may be helpful. Moreover, our results are for worst-case inputs; similar hardness results on average-case inputs is an interesting direction. Finally, we leave the door open for the possibility of randomized algorithms that achieve sub-quadratic complexity and are correct with high probability.

Acknowledgments

This work was supported in part by the National Scienc Foundation (under grants CCF-2005804 and CCF-1801495) and USDA/NIFA (under grant 2021-67021-35329).

References

Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for lcs and other sequence similarity measures. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 59–78. IEEE, 2015.

Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. *SIAM Journal on Computing*, 47(6):2527–2555, 2018.

Amir Abboud, Vincent Cohen-Addad, and Hussein Houdrougé. Subquadratic high-dimensional hierarchical clustering. *Advances in Neural Information Processing Systems*, 32, 2019.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless seth is false). In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 51–58, 2015.

Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. On the fine-grained complexity of empirical risk minimization: Kernel methods and neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020.

Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless seth fails. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 661–670. IEEE, 2014.

Karl Bringmann. Fine-grained complexity theory: Conditional lower bounds for computational geometry. In *Conference on Computability in Europe*, pages 60–70. Springer, 2021.

Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 79–97. IEEE, 2015.

Karl Bringmann and André Nusser. Translating hausdorff is hard: Fine-grained lower bounds for hausdorff distance under translation. In *37th International Symposium on Computational Geometry*, 2021.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *CoRR*, 2021a.

Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. Skyformer: Remodel self-attention with gaussian kernel and nystr\" om method. *Advances in Neural Information Processing Systems*, 34, 2021b.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pages 2793–2803. PMLR, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. ISSN 0022-0000.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

Piotr Indyk. Beyond p vs. np: quadratic-time hardness for big data problems. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 1–1, 2017.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys (CSUR)*, 2021.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pages 5562–5571. PMLR, 2021.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

Valerii Likhosherstov, Krzysztof M Choromanski, Jared Quincy Davis, Xingyou Song, and Adrian Weller. Sub-linear memory: How to make performers slim. *Advances in Neural Information Processing Systems*, 34, 2021.

Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing XU, Tao Xiang, and Li Zhang. Soft: Softmax-free transformer with linear complexity. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21297–21309. Curran Associates, Inc., 2021.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th annual ACM SIGACT symposium on theory of computing*, pages 1260–1268, 2018.

Aviad Rubinstein and Virginia Vassilevska Williams. Seth vs approximation. *ACM SIGACT News*, 50(4):57–76, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148, 2021.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2019.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

Appendix A. Proofs for Hardness of TVPP, BHCP and BHFP

Lemma 1. *Assume SETH. Then for any $\epsilon > 0$, the computational complexity of TVPP is $\Omega(n^{2-\epsilon})$ for $d = \omega(\log n)$.*

Proof Consider the two sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ given in OVP, where $a_i, b_i \in \{0, 1\}^d$ are binary vectors for all $i \in [n]$. For any vector $a_i \in A$, let us define $\bar{a_i} \in \mathbb{R}^{2d}$ as the concatenation of vector a_i and vector $(\mathbf{1} - a_i)$, where $\mathbf{1} := (1, 1, \ldots, 1) \in \mathbb{R}^d$. Then we have $\|\bar{a_i}\|_1 = \|a_i\|_1 + \|\mathbf{1} - a_i\|_1 = d$. Now define $\bar{A}$ as the set of $\bar{a_i}$s. For any vector $b_j \in B$, let us define $\bar{b_j} \in \mathbb{R}^{2d}$ as the concatenation of vector $(\mathbf{1} - b_j)$ and vector $\mathbf{1}$. Now define $\bar{B}$ as the set of $\bar{b_j}$. Because the overall dimensions of A, B and $\bar{A}, \bar{B}$ are nd and $2nd$ respectively, this process takes $O(nd)$ time. Now for any $i, j \in [n]$ we have

$$\bar{a_i}^T \bar{b_j} = {a_i}^T(\mathbf{1} - b_j) + (\mathbf{1} - a_i)^T \mathbf{1} = \|a_i\|_1 - a_i^T b_j + \|\mathbf{1} - a_i\|_1 = d - a_i^T b_j.$$

$\bar{a_i}^T \bar{b_j} = d$ if and only if ${a_i}^T b_j = 0$. Now if we run the algorithm for TVPP with the threshold $t = d$ on the sets $\bar{A}$ and $\bar{B}$ to find a pair $\bar{a} \in \bar{A}$ and $\bar{b} \in \bar{B}$ that satisfies $\bar{a}^T \bar{b} \geq d$, then we can conclude that there is a pair $a \in A$ and $b \in B$ that satisfies $a^T b = 0$.

Thus, if there is an algorithm for TVPP that runs in $O(n^{2-\epsilon})$ time, this entire process takes at most $O(nd + n^{2-\epsilon}) = O(n^{2-\epsilon})$ as long as $d = o(n^{1-\epsilon})$, and this algorithm decides if there exists at least a pair of vectors $a \in A$ and $b \in B$ such that $a^T b = 0$ in $O(n^{2-\epsilon})$ time, which contradicts the hardness result of OVP. As a result, for at least one t, TVPP problem cannot be solved in $O(n^{2-\epsilon})$ time. (We note that for $t = 1$, this problem have a linear time algorithm. Selecting rows of matrix Q as elements of set A, rows of matrix K as elements of set B, and rows of matrix V as 1, then calculating $Y = QK^T V$ takes linear time on n, by firstly calculating $K^T V$, then calculating $Q(K^T V)$. If there is any positive value of Y, then we can conclude that there is a pair $a \in A$ and $b \in B$ that satisfies $a^T b \geq 1$.)

■

Lemma 2. *Assume SETH. Then for any $\epsilon > 0$, the computational complexity of BHFP is $\Omega(n^{2-\epsilon})$ for $d = \omega(\log n)$.*

Proof Consider the two sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ given in OVP, where $a_i, b_i \in \{0, 1\}^d$ are binary vectors for all $i \in [n]$.

For any vector $a_i \in A$, let us define $\bar{a_i} \in \mathbb{R}^{3d}$ as the concatenation of vector a_i, vector $(\mathbf{1} - a_i)$ and vector $\mathbf{0}$ where $\mathbf{1} := (1, 1, \ldots, 1) \in \mathbb{R}^d$ and $\mathbf{0} := (0, 0, \ldots, 0) \in \mathbb{R}^d$. Then we have $\bar{a_i}^T \bar{a_i} = a_i{}^T a_i + (\mathbf{1} - a_i)^T (\mathbf{1} - a_i) = d$. Now define $\bar{A}$ as the set of $\bar{a_i}$s. For any vector $b_j \in B$, let us define $\bar{b_j} \in \mathbb{R}^{3d}$ as the concatenation of vector b_j, vector $\mathbf{0}$, and vector $(\mathbf{1} - b_j)$. Then we have $\bar{b_j}^T \bar{b_j} = b_j{}^T b_j + (\mathbf{1} - b_j)^T (\mathbf{1} - b_j) = d$. Now define $\bar{B}$ as the set of $\bar{b_j}$. Because the overall dimensions of A, B and $\bar{A}, \bar{B}$ are nd and $3nd$ respectively, this process takes $O(nd)$ time. Now for any $i, j \in [n]$ we have

$$\bar{a_i}^T \bar{b_j} = a_i{}^T b_j + (\mathbf{1} - a_i)^T \mathbf{0} + \mathbf{0}^T (\mathbf{1} - b_j) = a_i{}^T b_j.$$

The squared ℓ_2 distance between $\bar{a_i}$ and $\bar{b_j}$ for any $i, j \in [n]$ is

$$\|\bar{a_i} - \bar{b_j}\|_2^2 = (\bar{a_i} - \bar{b_j})^T (\bar{a_i} - \bar{b_j}) = \bar{a_i}^T \bar{a_i} + \bar{b_j}^T \bar{b_j} - 2\bar{a_i}^T \bar{b_j} = d + d - 2\bar{a_i}^T \bar{b_j} = 2d - 2a_i{}^T b_j.$$

$\|\bar{a_i} - \bar{b_j}\|_2^2 = 2d$ if and only if $a_i{}^T b_j = 0$. Now if we run the algorithm for BHFP with the threshold $t = \sqrt{2d}$ on the sets $\bar{A}$ and $\bar{B}$ to find a pair $\bar{a} \in \bar{A}$ and $\bar{b} \in \bar{B}$ that satisfy $\|\bar{a} - \bar{b}\|_2 \geq \sqrt{2d}$, then we can conclude that there is a pair $a \in A$ and $b \in B$ that satisfies $a^T b = 0$.

Thus, if there is an algorithm for BHFP that runs in $O(n^{2-\epsilon})$ time, this entire process takes at most $O(nd + n^{2-\epsilon}) = O(n^{2-\epsilon})$ as long as $d = o(n^{1-\epsilon})$, and this algorithm decides if there exists at least a pair of vectors $a \in A$ and $b \in B$ such that $a^T b = 0$ in $O(n^{2-\epsilon})$ time, which contradicts the hardness result of OVP. As a result, for at least one t, BHFP problem cannot be solved in $O(n^{2-\epsilon})$ time.

■

Lemma 3. *Assume SETH. Then for any $\epsilon > 0$, the computational complexity of BHCP is $\Omega(n^{2-\epsilon})$ for $d = \omega(\log n)$.*

Proof Consider the two sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ given in BHFP, where $a_i, b_i \in \{0, 1\}^d$ are binary vectors for all $i \in [n]$. Let t, t' be the given threshold for BHFP and BHCP respectively.

For any vector $b_j \in B$, let us define $\bar{b_j} \in \mathbb{R}^d$ as $(\mathbf{1} - b_j)$, where $\mathbf{1} := (1, 1, \ldots, 1) \in \mathbb{R}^d$. Now define $\bar{B}$ as the set of $\bar{b_j}$. Because the overall dimension of B and $\bar{B}$ is nd, this process takes $O(nd)$ time. The squared ℓ_2 distance between a_i and $\bar{b_j}$ for any $i, j \in [n]$ is

$$\|a_i - \bar{b_j}\|_2^2 = \|a_i - (\mathbf{1} - b_j)\|_2^2 = d - \|a_i - b_j\|_2^2.$$

$\|a_i - \bar{b_j}\|_2^2 < d - t^2 + 1$ if and only if $\|a_i - b_j\|_2 \geq t$. Now if we run the algorithm for BHCP with the threshold $t' = \sqrt{d - t^2 + 1}$ on the sets A and $\bar{B}$ to find a pair $a \in \bar{A}$ and $\bar{b} \in \bar{B}$ that satisfy $\|a - \bar{b}\|_2 < \sqrt{d - t^2 + 1}$, then we can conclude that there is a pair $a \in A$ and $b \in B$ that satisfies $\|a - b\|_2 \geq t$.

Thus, if there is an algorithm for BHCP that runs in $O(n^{2-\epsilon})$ time, this entire process takes at most $O(nd + n^{2-\epsilon}) = O(n^{2-\epsilon})$ as long as $d = o(n^{1-\epsilon})$, and this algorithm decides if there exists at least a pair of vectors $a \in A$ and $b \in B$ such that $\|a - b\|_2 \geq t$ in $O(n^{2-\epsilon})$ time, which contradicts

the hardness result of BHFP. As a result, for at least one t', BHCP problem cannot be solved in $O(n^{2-\epsilon})$ time. ■

Appendix B. Proof for Hardness of Sliding Window Dot-Product Self-Attention

Theorem 3. *Assume SETH. Let $f : \{\mathbb{R}^d, \mathbb{R}^d\} \longrightarrow \mathbb{R}$ as $f(x, y) = e^{x^T y}$. For set Q of vectors $Q_1, \ldots, Q_n$, we define the matrix $S \in \mathbb{R}^{n \times n}$ as*

$$(S)_{ij} = \begin{cases} f(Q_i, Q_j) & |i - j| \leq w/2 \\ 0 & \textit{otherwise} \end{cases}$$

Also for any matrix $M \in \mathbb{R}^{n \times n}$, let $h(M) = M$. Let $Y = h(S) \cdot V \in \mathbb{R}^{n \times d_v}$ be a self-attention. Then for any $\epsilon > 0$, computing a matrix $\hat{Y} \in \mathbb{R}^{n \times d_v}$ that satisfies any of the following conditions requires $\Omega((nw)^{1-\epsilon})$ time when $d_q = \omega(\log \sqrt{nw})$ and $w = \omega(d_q)$.

1. *$\hat{Y} = Y$(exact).*
2. *$|\hat{Y}_{ij} - Y_{ij}| \leq \mu |Y_{ij}|$ for all $i \in [n]$ and $j \in [d_v]$ where $0 \leq \mu < 1$(multiplicative approximation).*
3. *$|\hat{Y}_{ij} - Y_{ij}| \leq \mu$ for all $i \in [n]$ and $j \in [d_v]$ where $0 \leq \mu$(additive approximation).*

Proof Consider the two sets $A = \{a_1, \ldots, a_k\}$ and $B = \{b_1, \ldots, b_k\}$ given in TVPP, where $k = \sqrt{nw}$ and $a_i, b_i \in \{0, 1\}^d$ are binary vectors for all $i \in [k]$. The problem is to decide if there exists at least a pair $a \in A$ and $b \in B$ that satisfies $a^T b \geq t$ for a given $t \in [d]$.

We construct our matrices Q and V in the following way. Firstly, define the function $(\alpha \pmod k) := m$ if $\alpha \equiv m \pmod k$ and $m \in [k]$. Create the matrix $Q \in \mathbb{R}^{3n \times d}$ with its rows with even indices as $Q_{2\alpha} = Cb_{(\alpha \pmod k)}$, and its rows with odd indices as $Q_{2\alpha-1} = a_{(\alpha \pmod k)+w\lfloor \frac{2\alpha-1}{2k} \rfloor}$. Thus, this process takes $O(nd)$ time. By this construction, each (a_i, b_j) pair is found at a distance w.

Also, select V as concatenation of vector $(1, 0, \ldots, 1, 0) \in \mathbb{R}^{3n}$.

With this we have $Y = h(S) \cdot V \in \mathbb{R}^{3n \times 1}$. Since Y is a vector, we abuse the notation again and define Y_i as the i^{th} element of Y.

Now consider the first n even rows of Y. Because odd rows of matrix Q is from set A, and even rows of matrix Q is from set B, the even rows of Y becomes the summation of the exponential of the C times of the dot products of w different (a_i, b_j) pairs by the definition of vector V.

Also, each (a_i, b_j) pair is found at a distance w, so that the exponential of the C times dot products of all pairs appear in the sliding window attention score matrix and contributes Y_{2l} for at least an $l \in [n]$ value.

Now we check the magnitudes of Y_{2l} for $l \in [n]$ in order to distinguish between true and false cases in TVPP. It takes $O(n)$ time to check these n values. First, we focus on the exact computation. Consider the following two cases.

Case 1. There are no pairs $a \in A$ and $b \in B$ with $(a)^T b \geq t$, that is for all $i, j \in [k]$, we have $(a_i)^T b_j \leq t - 1$, and $e^{C(a_i)^T b_j} \leq e^{C(t-1)}$. Then for all $l \in [n]$, we have $Y_{2l} \leq we^{C(t-1)} := \delta$.

Case 2. There is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, that is for some $i, j \in [k]$, we have $(a_i)^T b_j \geq t$, and $e^{C(a_i)^T b_j} \geq e^{Ct}$. Because any pair appears in some element of Y, we have $Y_{2l} \geq e^{Ct} := \Delta$ for some odd $l \in [n]$.

In order to distinguish between two cases, it is sufficient to have $\Delta > \delta$. This holds with $C = 2 \log w$.

Now let us consider multiplicative approximation error. With a μ-multiplicative factor, if there are no pairs $a \in A$ and $b \in B$ with $a^T b \geq t$, then we have for all $l \in [n], \hat{Y}_{2l} \leq (1+\mu) Y_{2l} \leq (1+\mu) w e^{C(t-1)} := \hat{\delta}$. On the other hand, if there is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, then for some $l \in [n]$, we have $\hat{Y}_{2l} \geq (1-\mu) Y_{2l} \geq (1-\mu) e^{Ct} := \hat{\Delta}$. In order to distinguish between two cases, it is sufficient to have $\hat{\Delta} > \hat{\delta}$ and this inequality holds with $C = 2 \log(\frac{1+\mu}{1-\mu} w)$.

Finally we look at additive approximation error. With a μ-additive factor, if there are no pairs $a \in A$ and $b \in B$ with $a^T b \geq t$, then we have for all $l \in [n], \hat{Y}_{2l} \leq Y_{2l} + \mu \leq w e^{C(t-1)} + \mu := \hat{\delta}$. On the other hand, if there is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, then for some $l \in [n]$, we have $\hat{Y}_{2l} \geq Y_{2l} - \mu \geq e^{Ct} - \mu := \hat{\Delta}$. In order to distinguish between two cases, it is sufficient to have $\hat{\Delta} > \hat{\delta}$ and this inequality holds with $C = 2 \log(w + 2\mu)$.

Thus, if there is an algorithm for computing self-attention up to an element-wise multiplicative or additive error μ that runs in $O(k^{2-\epsilon})$ time, this entire process takes at most $O(nd + k^{2-\epsilon}) = O(\sqrt{nw}^{2-\epsilon}) = O((nw)^{1-\epsilon})$ as long as $d = o(w^{1-\epsilon})$. Therefore, this algorithm decides if there exists at least a pair of vectors $a \in A$ and $b \in B$ such that $a^T b \geq t$ in $O((nw)^{1-\epsilon})$ time, which contradicts the hardness result of TVPP (Lemma 1). This completes the proof.

■

A similar proof also applies for dilated sliding window (Beltagy et al., 2020), where the self-attention score is calculated as Theorem 6. Also, when the self-attention score is the softmax dot product (where softmax is only applied to the window size in each row), one can prove $O(nw^{1-\epsilon})$ complexity by following the proof of Theorem 5.

Appendix C. Proofs for Hardness of Self-Attention Score Matrix S Approximation

Theorem 12 *Assume SETH. For any $i, j \in [n]$, let $S_{ij} = f(Q_i, Q_j) = e^{Q_i^T Q_j}$ and for any matrix $M \in \mathbb{R}^{n \times n}$, let $h(M) = M$. $\hat{S} \in \mathbb{R}^{n \times n}$ satisfies any of the following conditions:*

1. *$|\hat{S}_{ij} - S_{ij}| \leq \mu |S_{ij}|$ for all $i \in [n]$ and $j \in [d_v]$ where $0 \leq \mu < 1$(multiplicative approximation).*
2. *$|\hat{S}_{ij} - S_{ij}| \leq \mu$ for all $i, j \in [n]$, where $0 \leq \mu$(additive approximation).*

Let $Y = h(\hat{S}).V \in \mathbb{R}^{n \times d_v}$ be a self-attention. Then for any $\epsilon > 0$, computing self-attention $Y \in \mathbb{R}^{n \times d_v}$ requires $\Omega(n^{2-\epsilon})$ time when $d_q = \omega(\log n)$.

Proof Consider two sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ given in TVPP, where $a_i, b_i \in \{0,1\}^d$ are binary vectors for all $i \in [n]$. The problem is to decide if there exists at least a pair $a \in A$ and $b \in B$ that satisfies $a^T b \geq t$ for a given $t \in [n]$.

Suppose that the matrices Q and V are constructed according to TVPP vector gadgets described in the section 4.2. With this we have $Y = h(\hat{S}).V \in \mathbb{R}^{n \times 1}$. Since Y is a vector, we abuse the notation again and define Y_i as the i^{th} element of Y. Now consider the first n elements of Y. Since $Y = SV$, we have that

$$Y_i = \sum_{j=1}^{n} \hat{S}_{ij} \text{ for any } i \in [n].$$

Now we check the magnitude of $Y_i, i \in [n]$ in order to distinguish between true and false cases in TVPP. It takes $O(n)$ time to check each $Y_i, i \in [n]$. First we focus on the multiplicative error approximation. Consider the following two cases.

Case 1. There are no pairs $a \in A$ and $b \in B$ with $a^T b \geq t$, that is for all $i, j \in [n]$, we have $a_i^T b_j \leq t-1$, and $S_{ij} = e^{Ca_i^T b_j} \leq e^{C(t-1)}$, so $\hat{S}_{ij} \leq (1+\mu)S_{ij} = (1+\mu)e^{Ca_i^T b_j} \leq (1+\mu)e^{C(t-1)}$. Then for all $l \in [n]$, $Y_l \leq n(1+\mu)e^{C(t-1)} := \delta$.

Case 2. There is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, that is for some $i, j \in [n]$, we have $a_i^T b_j \geq t$, and $S_{ij} = e^{Ca_i^T b_j} \geq e^{Ct}$, so $\hat{S}_{ij} \geq (1-\mu)S_{ij} = (1-\mu)e^{Ca_i^T b_j} \geq (1-\mu)e^{Ct}$. Then for some $l \in [n]$, we have $Y_l \geq (1-\mu)e^{Ct} + n - 1 := \Delta$

In order to distinguish between two cases, it is sufficient to have $\Delta > \delta$. This holds with $C = 2\log(\frac{1+\mu}{1-\mu}n)$.

Now let us look at the additive error approximation. With a μ additive factor, if there are no pairs $a \in A$ and $b \in B$ with $a^T b \geq t$, then we have for all $l \in [n], Y_l \leq ne^{C(t-1)} + n\mu := \hat{\delta}$. On the other hand, if there is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, then for some $l \in [n]$, we have $Y_l \geq e^{Ct} + n - 1 - \mu := \hat{\Delta}$. In order to distinguish between two cases, it is sufficient to have $\hat{\Delta} > \hat{\delta}$ and this inequality holds with $C = 2\log(n + 2\mu)$.

Thus, if there is an algorithm for computing self-attention up to an element-wise multiplicative or additive error μ that runs in $O(n^{2-\epsilon})$ time, this entire process takes at most $O(n^{2-\epsilon})$, and this algorithm decides if there exists at least a pair of vectors $a \in A$ and $b \in B$ such that $a^T b \geq t$ in $O(n^{2-\epsilon})$ time, which contradicts the hardness result of TVPP (Lemma 1). This completes the proof. ■

Similar proofs also work to show the quadratic complexity of multiplicative approximation to the S of softmax dot-product self-attention, and ℓ_2-self-attention directly from Theorem 5 and Theorem 7. And also by Theorem 6, one can show the quadratic complexity of additive and multiplicative approximation to the S of sliding window dot-product self-attention.

Appendix D. Proofs of Lemmas for Polynomial Approximations of Self-Attention

Lemma 4. *Let p be an integer ≥ 0 and let S_{ij} be $C \cdot (Q_i^T K_j)^p$ where $i, j \in [n]$ and C is a constant. Then SV can be computed in $O(nd_q^p d_v)$ time.*

Proof We omit the constant C in this proof since we can multiply any of the matrices Q, K, V by C in $O(nd_q)$ or $O(nd_v)$ time before the rest of the computation.

When $p = 1$, $SV = QK^T V$ thus SV can be trivially computed by first computing $K^T V$ by multiplying K^T and V with $O(d_q n d_v)$ time, then by multiplying Q and $K^T V$ with $O(nd_q d_v)$ time.

For $p \geq 2$, Consider a fixed (i, j) pair. Now

$$\begin{aligned} S_{ij} = (Q_i^T K_j)^p = \Big(\sum_{r=1}^{d_q} x_r y_r\Big)^p = \Big(x_1 y_1 + \cdots + x_{d_q} y_{d_q}\Big)^p = \sum_{r_1,\ldots,r_p=1}^{d_q} x_{r_1} y_{r_1} \ldots x_{r_p} y_{r_p} \\ = \sum_{r_1,\ldots,r_p=1}^{d_q} (x_{r_1} \ldots x_{r_p})(y_{r_1} \ldots y_{r_p}) \end{aligned} \tag{3}$$

where $x = Q_i, y = K_j$ and x_r, y_r denote the r^{th} element of the corresponding vectors. Given a vector $v \in \mathbb{R}^{d_q}$, let us define a function $\alpha : \mathbb{R}^{d_q} \to \mathbb{R}^{d_q^p}$ where elements of $\alpha(v)$ are computed

by element-wise multiplication $v_{r_1} \dots v_{r_p}, r_1, \dots, r_p \in [d_q]$(ordered p-permutations of d_q with replacement). With this, let us define $\hat{Q} \in \mathbb{R}^{n \times d_q^p}$(and $\hat{K}$) where each row i is computed with $\alpha(Q_i)$(respectively $\alpha(K_j)$ for $\hat{K}$). $\hat{Q}, \hat{K}$ matrices can be computed in $O(nd_q^p)$ time. From Eq.3, it is evident that $(Q_i^T K_j)^p = \hat{Q}_i^T \hat{K}_j$. Now, similar to the case $p = 1$ one can compute SV by first computing $\hat{K}^T V$ by multiplying $\hat{K}^T$ and V with $O(d_q^p n d_v)$ time, then by multiplying $\hat{Q}$ and $\hat{K}^T V$ with $O(nd_q^p d_v)$ time. This completes the proof.

■

Lemma 5. *Let p be an integer ≥ 0. Then for all $i \in [n]$, $\sum_{\hat{j}=1}^{n} C \cdot (Q_i^T K_{\hat{j}})^p, \hat{j} \in [n]$ where C is a constant can be computed in $O(nd_q^p)$ time.*

Proof We omit the constant C in this proof since we can multiply any of the matrices Q, K by C in $O(nd_q)$ time before the rest of the computation.

Let $\hat{Q}, \hat{K} \in \mathbb{R}^{n \times d_q^p}$ be the matrices computed by applying $\alpha(.)$ on rows of Q, V as stated in the proof of Lemma 8(from Eq. 3). The time complexity of computing $\hat{Q}, \hat{K}$ is $O(nd_q^p)$.

Now we have $\sum_{\hat{j}=1}^{n} (Q_i^T K_{\hat{j}})^p = \sum_{\hat{j}=1}^{n} \hat{Q}_i{}^T \hat{K}\hat{j} = \hat{Q}_i{}^T \sum_{\hat{j}=1}^{n} \hat{K}\hat{j}$. Let $A = \sum_{\hat{j}=1}^{n} \hat{K}\hat{j}$ and A can be computed in $O(nd_q^p)$ time(taking the summation of n d_q^p size vectors). We store the value of A in the memory and reuse it. One can simply compute the inner product of $\hat{Q}_i{}^T$ and A in $O(d_q^p)$ time per i. For all $i \in [n]$, this takes $O(nd_q^p)$ time which gives the desired time complexity.

■

Appendix E. Proof for Hardness of ℓ_2-Self-Attention

Theorem 4. *Assume SETH. For any $i, j \in [n]$, let $S_{ij} = f(Q_i, Q_j) = e^{C.\|Q_i - Q_j\|_2^2}$ where C is a parameter and for any matrix $M \in \mathbb{R}^{n \times n}$, let $h(M) = M$. Let $Y = h(S).V \in \mathbb{R}^{n \times d_v}$ be a self-attention. Then for any $\epsilon > 0$, computing a matrix $\hat{Y} \in \mathbb{R}^{n \times d_v}$ that satisfies any of the following conditions requires $\Omega(n^{2-\epsilon})$ time when $d_q = \omega(\log n)$.*

1. $\hat{Y} = Y$(exact).

2. $|\hat{Y}_{ij} - Y_{ij}| \leq \mu |Y_{ij}|$ for all $i \in [n]$ and $j \in [d_v]$ where $0 \leq \mu < 1$(multiplicative approximation).

Proof Consider two sets $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_n\}$ given in BHCP, where $a_i, b_i \in \{0, 1\}^d$ are binary vectors for all $i \in [n]$. The problem is to decide if there exists at least a pair $a \in A$ and $b \in B$ that satisfies $\|a - b\|_2^2 < t$ for a given $t \in [n]$.

Suppose that the matrices Q and V are constructed according to BHCP vector gadgets described in the section 4.2. With this we have $Y = h(S).V \in \mathbb{R}^{n \times 1}$. Since Y is a vector, we abuse the notation again and define Y_i as the i^{th} element of Y. Now consider the first n elements of Y. Since $Y = SV$, we have that

$$Y_i = \sum_{j=1}^{n} e^{-C\|a_i - b_j\|_2^2} \text{ for any } i \in [n].$$

Now we check the magnitude of $Y_i, \in [n]$ in order to distinguish between true and false cases in BHCP. It takes $O(n)$ time to check each $Y_i, i \in [n]$. First we focus on the exact computation. Consider the following two cases.

Case 1. There are no pairs $a \in A$ and $b \in B$ with $\|a_i - b_j\|_2^2 \leq t$, that is for all $i, j \in [n]$, we have $\|a_i - b_j\|_2^2 \geq t$, and $e^{-C\|a_i - b_j\|_2^2} \leq e^{-Ct}$. Then for all $l \in [n]$, $Y_l \leq ne^{-Ct} := \delta$.

Case 2. There is a pair $a \in A$ and $b \in B$ with $\|a - b\|_2^2 < t$, that is for some $i, j \in [n]$, we have $\|a_i - b_j\|_2^2 \leq t - 1$, and $e^{-C\|a_i - b_j\|_2^2} \geq e^{-C(t-1)}$. Thus for some $l \in [n]$, we have $Y_l \geq e^{-C(t-1)} := \Delta$

In order to distinguish between two cases, it is sufficient to have $\Delta > \delta$. This holds with $C = 2 \log n$.

Now let us look at the multiplicative error approximation. With a μ multiplicative factor, if there are no pairs $a \in A$ and $b \in B$ with $a^T b \geq t$, then we have for all $l \in [n], \hat{Y}_l \leq (1 + \mu)Y_l \leq (1 + \mu)ne^{-Ct} := \hat{\delta}$. On the other hand, if there is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, then for some $l \in [n]$, we have $\hat{Y}_l \geq (1 - \mu)Y_l \geq (1 - \mu)e^{-C(t-1)} := \hat{\Delta}$. In order to distinguish between two cases, it is sufficient to have $\hat{\Delta} > \hat{\delta}$ and this inequality holds with $C = 2 \log(\frac{1+\mu}{1-\mu} n)$.

Thus, if there is an algorithm for computing self-attention up to an element-wise multiplicative error μ that runs in $O(n^{2-\epsilon})$ time, this entire process takes at most $O(n + nd + n^{2-\epsilon}) = O(n^{2-\epsilon})$ as long as $d = o(n^{1-\epsilon})$, and this algorithm decides if there exists at least a pair of vectors $a \in A$ and $b \in B$ such that $\|a_i - b_j\|_2^2 \geq t$ in $O(n^{2-\epsilon})$ time, which contradicts the hardness result of BHCP (Lemma 3). This completes the proof. ■

Here in the proof we set $C = 2 \log(\frac{2(1+\mu)}{1-\mu} n)$ however, the parameter C in the RBF kernel is predefined, therefore the hardness result is valid only conditioned on this specific C. We can bypass this by simply defining $C = \log(\frac{2(1+\mu)}{1-\mu} n) = \alpha\beta$, then setting α to the predefined constant of the RBF function and solving this equation for β. After this, all that remains is modifying the BHCP vector gadget(see section 4.2) by multiplying each vector $Q_i, i \in [2n]$ by the scaler $\sqrt{\beta}$ in $O(n)$ time, and we obtain the desired hardness result for general RBF kernel.

Appendix F. Multi-Head Self-Attention

Lemma 6. k parallel OVP (or TVPP, or BHCP, or BHFP) problems (each has two sets with n binary vectors) require $O(kn^{2-\epsilon})$ time for any ϵ.

Proof Suppose there is an algorithm for k parallel OVP (TVPP, BHCP, BHFP) problems better than $O(kn^{2-\epsilon})$ time.

Say A_i and B_i are the sets of binary vectors with size n for any $i \in [k]$.

For each $t \in [k]$, look at these k parallel OVP (TVPP, BHCP, BHFP) problems:

$$(A_1, B_t), (A_2, B_{t+1}), \cdots, (A_k, B_{t+k-1}), \text{where } B_{l+k} = B_l$$

Because of the assumption, there is an algorithm better than $O(kn^{2-\epsilon})$ time.

So that, there is an $O(k \times kn^{2-\epsilon})$-time algorithm that solves OVP problem of ($A = A_1 \cup \cdots A_k$, $B = B_1 \cup \cdots B_k$). In other words for the OVP (TVPP, BHCP, BHFP) problem with sets of size nk binary vectors has an algorithm in $O((kn)^{2-\delta})$-time (by selecting $\delta = \epsilon/2$ and $k < n$). This contradicts SETH. As a result, k parallel OVP (TVPP, BHCP, BHFP) problems require $O(kn^{2-\epsilon})$ time for any ϵ. ■

This lemma proves that the direct sum of computational complexity for OVP (TVPP, BHCP, BHFP) problems is valid.

Appendix G. Discussion of Additive Approximation

This part depends on for a given C value, and the selected δ and Δ values, the difference $\Delta - \delta$ is positive. So that, this allows us to select an additive error μ.

The following theorem shows that the we cannot reach better elementwise additive approximation than $e^{-2d\log(n+2)} = (n+2)^{-2d}$ for the ℓ_2 distance self-attention. So this element-wise additive approximation error order is $O(n^{-d})$.

Theorem 5. Assume SETH. For any $i, j \in [n]$, let $S_{ij} = f(Q_i, Q_j) = e^{-C\|Q_i - Q_j\|_2^2}$, where C is a parameter and for any matrix $M \in R^{n\times n}$. Let $h(M) = M$ and $Y = h(S)$. $V \in \mathbb{R}^{n\times d_v}$ be self-attention. Then for any $\epsilon > 0$, computing a matrix $\hat{Y} \in R^{n\times d_v}$ that satisfies the following condition requires $\Omega(n^{2-\epsilon})$ time when $d_q = \omega(\log n)$: $|\hat{Y}_{ij} - Y_{ij}| \leq \mu$ for all $i \in [n]$ and $j \in [d_v]$, where $0 \leq \mu \leq e^{-2d\log(n+2)} = (n+2)^{-2d}$ (additive approximation)

Proof Consider two sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ given in BHCP, where $a_i, b_i \in \{0,1\}^d$ are binary vectors for all $i \in [n]$. The problem is to decide if there exists at least a pair $a \in A$ and $b \in B$ that satisfies $\|a - b\|_2^2 < t$ for a given $t \in [n]$.

Suppose that the matrices Q and V are constructed according to BHCP vector gadgets described in the section 4.2. With this we have $Y = h(S) \cdot V \in R^{n\times 1}$. Since Y is a vector, we abuse the notation again and define Y_i as the ith element of Y. Now consider the first n elements of Y. Since $Y = SV$, we have that $Y_i = \sum_{j=1}^{n} e^{-C\|a_i - b_j\|_2^2}$ for any $i, j \in [n]$.

Case 1. There are no pairs $a \in A$ and $b \in B$ that satisfies $\|a - b\|_2^2 < t$, that is for all $i, j \in [n]$, we have $\|a - b\|_2^2 \geq t$, and $e^{-C\|a_i - b_j\|_2^2} \leq e^{-Ct}$. Then for all $l \in [n]$, $Y_l \leq ne^{-Ct}$. It is given that $|\hat{Y}_l - Y_l \leq \mu$, so $\hat{Y}_l \leq Y_l + \mu \leq ne^{-Ct} + \mu \leq ne^{-Ct} + e^{-2d\log(n+2)} =: \delta$.

Case 2. There is a pair $a \in A$ and $b \in B$ with $\|a - b\|_2^2 < t$, that is for some $i, j \in [n]$, we have $\|a - b\|_2^2 \leq t - 1$, and $e^{-C\|a_i - b_j\|_2^2} \geq e^{-C(t-1)}$. Thus for some $l \in [n]$, $Y_l \geq ne^{-C(t-1)}$. It is given that $|\hat{Y}_l - Y_l \leq \mu$, so $\hat{Y}_l \geq Y_l - \mu \leq ne^{-Ct} - \mu \geq ne^{-Ct} - e^{-2d\log(n+2)} =: \Delta$.

In order to distinguish between two cases, it is sufficient to have $\Delta > \delta$. This holds with $C = 2\log(n+2)$.

Thus, if there is an algorithm for computing self-attention up to an element-wise additive error $\mu \leq e^{-2d\log(n+2)}$ that runs in $O(n^{2-\epsilon})$ time, this entire process takes at most $O(n + nd + n^{2-\epsilon}) = O(n^{2-\epsilon})$ as long $d = O(n^{1-\epsilon})$, and this algorithm decides if there exists at least a pair of vectors $a \in A$ and $b \in B$ with $\|a - b\|_2^2 < t$ in $O(n^{2-\epsilon})$ time, which contradicts the hardness result of BHCP (Lemma 3). This completes the proof. ■

The following theorem shows that the we cannot reach better elementwise additive approximation than $e^{-3d\log(n)-3d^2} = n^{-3d} \cdot e^{-3d^2}$ for the softmax dot-product self-attention. This element-wise additive approximation error order is $O(n^{-d})$.

Theorem 6. Assume SETH. For any $i, j \in [n]$, let $S_{ij} = f(Q_i, Q_j) = e^{Q_i^T Q_j}$, and for any matrix $M \in \mathbb{R}^{n\times n}$, let $h(M) \in \mathbb{R}^{n\times n}$ be the matrix where for all $i, j \in [n]$, $\{h(M)\}_{ij} = \frac{M_{ij}}{\sum_{k_1}^{n} M_{ik}}$. Let $Y = h(S) \cdot V \in \mathbb{R}^{n\times d_v}$ be self-attention. Then provided $d_q = \omega(\log n)$, for any $\epsilon > 0$, computing a matrix $\hat{Y} = \mathbb{R}^{n\times d_v}$ that satisfies the following condition requires $\Omega(n^{2-\epsilon})$ time:

$|\hat{Y}_{ij} - Y_{ij}| \leq \mu$ for all $i \in [n]$ and $j \in [d_v]$, where $0 \leq \mu \leq e^{-3d\log(n)-3d^2}$ (additive approximation)

Proof The proof is technically similar. Suppose that the matrices Q and V are constructed according to TVPP vector gadgets described in Section 4.2. With this we have $Y = h(S) \cdot V \in \mathbb{R}^{n \times 1}$. Consider the first n elements of Y. Since $h(\cdot)$ act as the row-wise softmax function, we have

$$Y_i = \sum_{j=n+1}^{2n} S_{ij} = \sum_{j=1}^{n} \frac{e^{Ca_i^T b_j}}{\sum_{k=1}^{n} e^{Ca_i^T a_k} + \sum_{k=1}^{n} e^{Ca_i^T b_k}}$$
$$= \frac{\sum_{j=1}^{n} e^{Ca_i^T b_j}}{\sum_{k=1}^{n} e^{Ca_i^T a_k} + \sum_{k=1}^{n} e^{Ca_i^T b_k}}$$

Case 1. There are no pairs $a \in A$ and $b \in B$ that satisfies $a^T b \geq t$, that is for all $i, j \in [n]$, we have $a^T b \leq t - 1$, and $\sum_{j=1}^{n} e^{Ca_i^T b_j} \leq n e^{C(t-1)}$. For a function $\frac{x}{x+y}$, the maximum value is achieved at maximum x and minimum y values. Thus, for all $l \in [n]$, $Y_l \leq \frac{n e^{C(t-1)}}{n e^{C(t-1)} + n} = \frac{e^{C(t-1)}}{e^{C(t-1)} + 1}$. It is given that $|\hat{Y}_l - Y_l| \leq \mu$, so $\hat{Y}_l \leq Y_l + \mu \leq \frac{e^{C(t-1)}}{e^{C(t-1)}+1} + \mu \leq \frac{e^{C(t-1)}}{e^{C(t-1)}+1} + e^{-3d\log(n)-3d^2} =: \delta$.

Case 2. There is a pair $a \in A$ and $b \in B$ with $a^T b \geq t$, that is for some $i, j \in [n]$, we have $a^T b \geq t$. Then the row sum corresponding to that i, j pair is $\sum_{j=1}^{n} e^{Ca_i^T b_j} \geq e^{Ct} + (n-1)e^0 = e^{Ct} + n - 1$. For a function $\frac{x}{x+y}$, the maximum value is achieved at minimum x and maximum y values. Thus, for some $l \in [n]$, we have $Y_l \geq \frac{e^{Ct}+(n-1)}{e^{Ct}+(n-1)+ne^d}$. It is given that $|\hat{Y}_l - Y_l| \leq \mu$, so $\hat{Y}_l \geq Y_l - \mu \geq \frac{e^{Ct}+(n-1)}{e^{Ct}+(n-1)+ne^d} - \mu \geq \frac{e^{Ct}+(n-1)}{e^{Ct}+(n-1)+ne^d} - e^{-3d\log(n)-3d^2} =: \Delta$.

■

Appendix H. Discussion of Remark in Section 4.4

In our results, we show the quadratic hardness of multiplicative error approximations self-attention matrix elements for dot-product softmax self-attention mechanism. One assumption we make on the approximation factor μ is that $\mu < 1$. Consider the value of the parameter $C = \log(\frac{2(1+\mu)}{1-\mu}n) + d$ in the proof of theorem 5. Notice that when $\mu = 1$, C is not defined. In fact, having $\mu = 1$ implies that the $|\hat{Y}_l - Y_l| = |Y_l|$ for all $l \in [n]$, therefore one can approximate every entry of $\hat{Y}$ by 0 in $O(n)$ time while satisfying this condition. However, as mentioned in the remark, one can set μ close to 1 by setting it as $1 - \frac{1}{n^x}$ for a constant x, while maintaining the condition on C in the same order.