

---

# Multi Resolution Analysis (MRA) for Approximate Self-Attention

---

Zhanpeng Zeng<sup>1</sup> Sourav Pal<sup>1</sup> Jeffery Kline<sup>2</sup> Glenn Fung<sup>2</sup> Vikas Singh<sup>1</sup>

## Abstract

Transformers have emerged as a preferred model for many tasks in natural language processing and vision. Recent efforts on training and deploying Transformers more efficiently have identified many strategies to approximate the self-attention matrix, a key module in a Transformer architecture. Effective ideas include various prespecified sparsity patterns, low-rank basis expansions and combinations thereof. In this paper, we revisit classical Multiresolution Analysis (MRA) concepts such as Wavelets, whose potential value in this setting remains underexplored thus far. We show that simple approximations based on empirical feedback and design choices informed by modern hardware and implementation challenges, eventually yield a MRA-based approach for self-attention with an excellent performance profile across most criteria of interest. We undertake an extensive set of experiments and demonstrate that this multi-resolution scheme outperforms most efficient self-attention proposals and is favorable for both short and long sequences. Code is available at <https://github.com/mlpen/mra-attention>.

## 1. Introduction

The Transformer model proposed in (Vaswani et al., 2017) is the architecture of choice for a number of tasks in natural language processing (NLP) as well as vision. The “self-attention” mechanism within Transformers, and specifically, its extension referred to as Multi-Head Self-Attention, enable modeling dynamic token dependencies, and play a key role in the overall performance profile of the model. Despite these advantages, the  $O(n^2)$  complexity of the self-attention (where  $n$  is the sequence length) is a bottleneck, especially for longer sequences. Consequently, over the last year or so, extensive effort has been devoted in deriving methods that mitigate this quadratic cost – a wide choice of algorithms with linear (in the sequence length) complexity are now available (Wang et al., 2020; Choromanski et al., 2021; Zeng et al., 2021; Beltagy et al., 2020; Zaheer et al., 2020).

The aforementioned body of work on *efficient* transformers leverages the observation that the self-attention matrix has a parsimonious representation – the mechanics of how this is modeled and exploited at the algorithm level, varies from one method to the other. For instance, we may ask that self-attention has a pre-specified form of sparsity (for instance, diagonal or banded), see (Beltagy et al., 2020; Zaheer et al., 2020). Alternatively, we may model self-attention globally as a low-rank matrix, successfully utilized in (Wang et al., 2020; Xiong et al., 2021; Choromanski et al., 2021). Clearly, each modeling choice entails its own form of approximation error, which we can measure empirically on existing benchmarks. Progress has been brisk in improving these approximations: recent proposals have investigated a hybrid global+local strategy based on invoking a robust PCA style model (Chen et al., 2021) and hierarchical (or H-) matrices (Zhu & Soricut, 2021), which we will draw a contrast with.

---

<sup>1</sup>University of Wisconsin, Madison, USA <sup>2</sup>American Family Insurance, Madison, USA. Correspondence to: Zhanpeng Zeng <zzeng38@wisc.edu>.

*Proceedings of the 39<sup>th</sup> International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

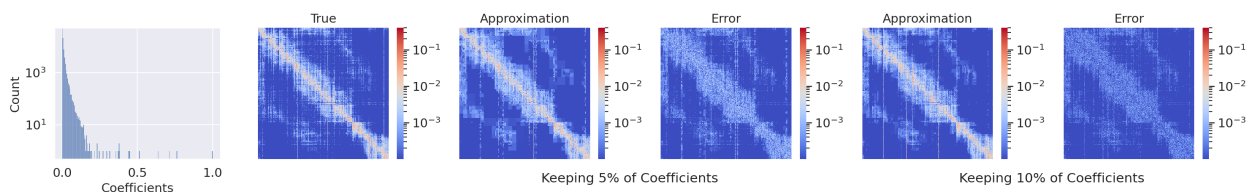


Figure 1. The left plot is the histogram of wavelet coefficients in log scale for 2D Haar basis. More than 95% of coefficients have a magnitude less than 0.005. The second plot is the true self-attention matrix  $\mathcal{A}$ . The third and fourth plots are the approximation and error by keeping top 5% of coefficients. The fifth and last plots are similar for keeping top 10% of coefficients. The errors  $\|\hat{\mathcal{A}} - \mathcal{A}\|_F$  for {MRA, low rank, sparsity} by keeping 10% of {coefficients, ranks, nonzero entries}, are 0.30, 1.24, 0.39, respectively. This shows performance benefits of MRA compared to low rank and sparsity. We discuss low rank and sparsity in more detail in §A.2.

**MRA.** Recall that the hybrid global+local intuition above has a rich classical treatment formalized under Multiresolution analysis (MRA) methods, and Wavelets are a prominent example (Mallat, 1999). While the use of wavelets for signal processing goes back at least three decades (if not longer), their use in machine learning especially for graph based datasets, as a numerical preconditioner, and even for matrix decomposition problems has seen a resurgence (Kondor et al., 2014; Ithapu et al., 2017; Gavish et al., 2010; Lee & Nadler, 2007; Hammond et al., 2011; Coifman & Maggioni, 2006). The extent to which classical MRA ideas (or even their heuristic forms) can guide efficiency in Transformers is largely unknown. Fig. 1 shows that, given a representative self-attention matrix, only 10% of the coefficients are sufficient for a high fidelity reconstruction. At a minimum, we see that the hypothesis of evaluating a MRA-based self-attention within a Transformer model may have merit.

**Contributions.** The goal of this paper is to investigate the specific modifications, adjustments and approximations needed to make the MRA idea operational in Transformers. It turns out that modulo some small compromises (on the theoretical side), MRA-based self-attention shows excellent performance across the board – it is competitive with standard self-attention and outperforms most of baselines while maintaining significantly high time and memory efficiency on both short and long sequences.

## 2. Preliminaries: Self-attention and Wavelets

We briefly review self attention and wavelet decomposition, two concepts that we will use throughout the paper.

### 2.1. Self-attention

Given embedding matrices  $Q, K, V \in \mathbb{R}^{n \times d}$  representing  $n$   $d$ -dimensional feature vectors for queries, key and values, respectively, self-attention is defined as

$$Z = \text{Softmax} \left( \underbrace{QK^T}_{\mathcal{P}} \right) V = D \underbrace{\exp(\mathcal{P})}_{\mathcal{A}} V$$

where  $D$  is a  $n \times n$  diagonal matrix that normalizes each row of the  $\mathcal{A}$  matrix such that the row entries sum up to 1. For notational simplicity, the scaling factor  $\sqrt{d}$  and linear projections applied to  $Q, K, V$  are omitted. An explicit calculation of  $\mathcal{A}$  and taking its product with  $V$  incurs a  $O(n^2)$  cost (if  $d$  is treated as fixed/constant for complexity analysis purposes), a serious resource bottleneck and the core focus of the existing literature on efficient Transformers.

**Related Work on Efficient Transformers.** A number of efficient self-attention methods have been proposed to reduce the  $O(n^2)$  cost. Much of this literature can be, roughly speaking, divided into two categories: low rank and spar-

sity. Linformer (Wang et al., 2020) shows that self-attention matrices are low rank and proposes to learn a projection – projecting the sequence length dimension to lower dimensions. Performer (Choromanski et al., 2021) and Random Feature Attention (Peng et al., 2021) view self-attention matrices as kernel matrices of infinite feature maps and propose using finite random feature maps to approximate the kernel matrices. Nyströmformer (Xiong et al., 2021) and SOFT (Lu et al., 2021) use a Nyström method for matrix approximation to approximate the self-attention matrices.

A number of methods also leverage the sparsity of self-attention matrices. By exploiting a high dependency within a local context, Longformer (Beltagy et al., 2020) proposes a sliding window attention with global attention on manually selected tokens. In addition to the sparsity used in Longformer, Big Bird (Zaheer et al., 2020) adds random sparse attention to further refine the approximation. Instead of a prespecified sparsity support, Reformer (Kitaev et al., 2020) uses hashing (LSH) to compute self-attention only within approximately nearby tokens, and YOSO (Zeng et al., 2021) uses the collision probability of LSH as attention weights and then, a LSH-based sampler to achieve linear complexity.

The discussion in (Chen et al., 2021) suggests that approximations relying solely on low rank or sparsity are limited and a hybrid model via robust PCA offers better approximation. Scatterbrain (Chen et al., 2021) uses a sparse attention + low rank attention strategy to avoid the cost of robust PCA. In §A.2, we discuss some limitations of low rank and sparsity for self-attention approximation, and show that a special form of our MRA approximation can offer a good solution for a relaxation of robust PCA.

Independent of our work, recently, H-Transformer-1D (Zhu & Soricut, 2021) proposed a hierarchical self-attention where the self-attention matrices have a low rank structure on the off-diagonal entries and attention is precisely calculated for the on-diagonal entries. This is also a form of multiresolution approximation for self-attention although the lower resolution for distant tokens may limit its ability to capture precise long range dependencies. While a prespecified structure can indeed provide an effective approximation scheme in specific settings, it would be desirable to avoid restriction to a fixed structure, if possible.

### 2.2. Wavelet Transform

A wavelet transform decomposes a signal into different scales and locations represented by a set of scaled and translated copies of a *fixed* function. This fixed function  $\varphi$  is called a mother wavelet, and the scaled and translated copies are called child wavelets specified by two factors, scale  $s$  and translation  $t$ .

$$\varphi_t^s(x) = \frac{1}{\sqrt{s}} \varphi\left(\frac{x-t}{s}\right)$$

Here,  $s$  controls the ‘‘dilation’’ or inverse of the frequency of wavelet, while  $t$  controls the location (e.g., time). These scaled/translated versions of mother wavelets play a key role in MRA. Given a choice of  $\varphi$ , the wavelet transform maps a function  $f$  to coefficients

$$\alpha_t^s = \langle f, \varphi_t^s \rangle = \int f(x) \varphi_t^s(x) dx$$

The coefficient  $\alpha_t^s$  captures the measurement of  $f$  at scale  $s$  and location  $t$ .

### 3. MRA view of Self-attention

To motivate the use of MRA, in §1, we used Fig. 1 to check how a 2D Haar wavelet basis decomposes the target matrix  $\mathcal{A}$  into terms involving different scales and translations, and terms with larger coefficients suffice for a good approximation of  $\mathcal{A}$ . But the reader will notice that the calculation of the coefficients requires access to the full matrix  $\mathcal{A}$ . Our discussion below will start from a formulation which will still need the full matrix  $\mathcal{A}$ . Later, in §4, by exploiting the locality of  $Q$  and  $K$ , we will be able to derive an approximation with reduced complexity (without access to  $\mathcal{A}$ ).

For simplicity, we assume that the sequence length  $n = 2^k$  for some integer  $k$ . Inspired by the Haar basis and its ability to adaptively approximate while preserving locality, we apply a pyramidal MRA scheme. We consider a decomposition of  $\mathcal{A}$  using a set of simpler unnormalized components  $B_{x,y}^s \in \mathbb{R}^{n \times n}$  defined as

$$(B_{x,y}^s)_{i,j} = \begin{cases} 1 & \text{if } sx - s < i \leq sx, sy - s < j \leq sy \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

for  $s \in \{1, 2, 4, \dots, n\}$  and  $x, y \in \{1, \dots, n/s\}$ . Here,  $(sx - s, sx] \times (sy - s, sy]$  is the support of  $B_{x,y}^s$ , and  $s$  represents the scale of the components, i.e., a smaller  $s$  denotes higher resolutions and vice versa. Also,  $x, y$  denote the translation of the components.

**Why not Haar basis?** The main reason for using the form in (1) instead of a 2D Haar basis directly is implementation driven, and will be discussed shortly in Remark 3.1. For the moment, we can observe that (1) is an overcomplete frame for  $\mathbb{R}^{n \times n}$ . As shown in Fig. 2, frame (1) has *one extra* scale (with support on a single entry) compared to the Haar basis (4 rows versus 3 rows). Except for this extra scale, (1) has the same support as the Haar basis at different scales. In addition, (1) provides scaled and translated copies of the ‘‘mother’’ component, similar to Haar.

Let  $\mathcal{I} = \{B_{x,y}^s\}$  be a set of components for the possible scales and translations, then we decompose

$$\mathcal{A} = \sum_{B_{x,y}^s \in \mathcal{I}} \alpha_{x,y}^s B_{x,y}^s$$

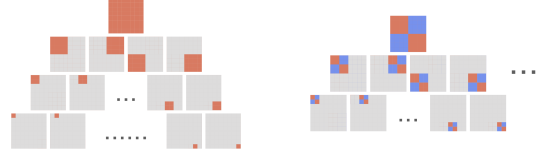


Figure 2. The left plot is the overcomplete frame defined in (1), which consists of 85 matrices for  $n = 8$ . The right plot is a 2D generalization of Haar basis, which consists of three groups of 21 self-similar matrices plus a constant matrix (all entries equal to  $\frac{1}{n}$ ). The color red, blue, and gray means positive, negative, and 0, respectively. Notice that (1) does not include negative entries: it involves more components but makes the formulation simpler.

for some set of coefficients  $\alpha_{x,y}^s$ . Since (1) is overcomplete, the coefficients  $\alpha_{x,y}^s$  are not unique. We specifically compute the coefficients  $\alpha_{x,y}^s$  as follows. Let  $E_n = \mathcal{A}$  and

$$\begin{aligned} \alpha_{x,y}^s &= \frac{1}{s^2} \langle B_{x,y}^s, E_s \rangle \\ E_{s/2} &= E_s - \sum_{B_{x,y}^s \in \mathcal{I}} \alpha_{x,y}^s B_{x,y}^s \end{aligned} \quad (2)$$

Here, the  $E_{s/2}$  denotes the residuals of the higher frequencies. At each scale  $s$ ,  $\alpha_{x,y}^s$  is the optimal solution of the least squares problem minimizing  $\|E_{s/2}\|_F^2$ . Intuitively, the approximation procedure starts from the coarsest approximation of  $\mathcal{A}$  which only consists of the lowest frequency, then the procedure refines the approximation by adding residuals of higher frequencies.

**Parsimony.** We empirically observe that the coefficients of most components are near zero, so we can represent  $\mathcal{A}$  with only a few components while maintaining the accuracy of approximation. Specifically, we can find a small subset  $\mathcal{J} \subset \mathcal{I}$ , the corresponding coefficients  $(\alpha_{x,y}^s)'$  computed following (2), and the resulting approximation

$$\hat{\mathcal{A}}^* = \sum_{B_{x,y}^s \in \mathcal{J}} (\alpha_{x,y}^s)' B_{x,y}^s \quad (3)$$

with a negligible approximation error  $\|\hat{\mathcal{A}}^* - \mathcal{A}\|_F$ .

But  $\hat{\mathcal{A}}^*$  in (3) does not suggest any interesting property of the approximation. But we can check an equivalent form of  $\hat{\mathcal{A}}^*$ . Denote the average of  $\mathcal{A}$  over the support of  $B_{x,y}^s$  to be

$$\mu_{s,x,y}^* = \frac{1}{s^2} \langle B_{x,y}^s, \exp(\mathcal{P}) \rangle \quad (4)$$

It turns out that the entries of  $\hat{\mathcal{A}}^*$  in (3) can be rewritten as

$$(\hat{\mathcal{A}}^*)_{i,j} = \mu_{s,x,y}^* \quad (5)$$

where  $(s, x, y)$  is the index of  $B_{x,y}^s \in \mathcal{J}$  that has the smallest support region and is supported on  $(i, j)$ . But if  $(i, j) \notin \text{supp}(B_{x,y}^s)$  for all  $B_{x,y}^s \in \mathcal{J}$ , then  $(\hat{\mathcal{A}}^*)_{i,j} = 0$ .

The  $(i, j)$  entry of  $\hat{\mathcal{A}}^*$  is precisely approximated by the average of  $\mathcal{A}$  over the smallest support region for  $B_{x,y}^s \in \mathcal{J}$  containing  $(i, j)$ . In other words, the procedure uses the highest resolution possible for a given  $\mathcal{J}$  as an approximation. We discuss and show how we obtain (5) from (3) in §A.3. The reader will notice that rewriting  $\hat{\mathcal{A}}^*$  as (5) is possible due to our modifications to the Haar basis in (1).

*Remark 3.1.* Consider using a Haar decomposition and let  $\mathcal{L}$  be the subset of basis with nonzero coefficients. The approximation  $(\hat{\mathcal{A}}_{\text{Haar}})_{i,j}$  depends on *all*  $\phi \in \mathcal{L}$  which are supported on  $(i, j)$ . For example, in the worst case, the coefficients of all  $\phi$  which are supported on  $(i, j)$  need to be nonzero to have  $(\hat{\mathcal{A}}_{\text{Haar}})_{i,j} = \mathcal{A}_{i,j}$ . We find that a hardware friendly and efficient approximation scheme in this case is challenging. On the other hand, when using the decomposition (2) over the overcomplete frame (1),  $(\hat{\mathcal{A}}^*)_{i,j}$  depends on only *one*  $B_{x,y}^s \in \mathcal{J}$  that has the smallest support region and is supported on  $(i, j)$ . This makes constructing the set  $\mathcal{J}$  easier and more flexible.

## 4. A Practical Approximation scheme

Given that we now understand all relevant modules, we can focus on practical considerations. Notice that each  $\mu_{s,x,y}^*$  requires averaging over  $O(s^2)$  entries of the matrix  $\mathcal{A}$ , so in the worst case, we would need access to the entire matrix  $\mathcal{A}$  to compute all  $\mu_{s,x,y}^*$  for  $B_{x,y}^s \in \mathcal{J}$ . Nonetheless, suppose that we still compute all the coefficients  $\alpha_{x,y}^s$  and then post-hoc truncate the small coefficients to construct the set  $\mathcal{J}$ . This approach will clearly be inefficient. In this section, we discuss two strategies where the main goal is efficiency.

### 4.1. Can we approximate $\mu_{s,x,y}^*$ quickly?

We first discuss calculating  $\mu_{s,x,y}^*$ . To avoid accessing the full matrix  $\mathcal{A}$ , instead of computing the average of exponential (4), we compute a lower bound (due to convexity of exponential), i.e., exponential of average (6), as an approximation.

$$\mu_{s,x,y} = \exp\left(\frac{1}{s^2} \langle B_{x,y}^s, \mathcal{P} \rangle\right) \quad (6)$$

We can verify that the expression in (6) can be computed efficiently as follows. Define  $\tilde{Q}_s, \tilde{K}_s \in \mathbb{R}^{n/s \times d}$  where  $\tilde{Q}_1 = Q, \tilde{K}_1 = K$ , and

$$\begin{aligned} (\tilde{Q}_s)_i &= \frac{1}{2}(\tilde{Q}_{s/2})_{2i-1} + \frac{1}{2}(\tilde{Q}_{s/2})_{2i} \\ (\tilde{K}_s)_i &= \frac{1}{2}(\tilde{K}_{s/2})_{2i-1} + \frac{1}{2}(\tilde{K}_{s/2})_{2i} \end{aligned} \quad (7)$$

Here,  $(Q_s)_i$  and  $(K_s)_i$  denote the  $i$ -th row of the matrix  $Q_s$  and  $K_s$ , respectively. Interestingly, (6) is simply,

$$\mu_{s,x,y} = \exp((\tilde{Q}_s)_x (\tilde{K}_s)_y^T)$$

Then, the approximation using (6) is,

$$\hat{\mathcal{A}}_{i,j} = \mu_{s,x,y}$$

where  $(s, x, y)$  is the same as (5) and  $\hat{\mathcal{A}}_{i,j} = 0$  otherwise. Each  $\mu_{s,x,y}$  only requires an inner product between one row of  $\tilde{Q}_s$  and one row of  $\tilde{K}_s$  and applying an exponential, so the cost of a single  $\mu_{s,x,y}$  is  $O(1)$  when  $\tilde{Q}_s$  and  $\tilde{K}_s$  are provided. We will discuss the overall complexity in §4.4.

While efficient, this modification will incur a small amount of error. However, by using the property of  $\mathcal{P}$  inherited from  $Q$  and  $K$ , we can quantify the error.

**Lemma 4.1.** *Assume for all  $(i_1, j_1), (i_2, j_2) \in \text{supp}(B_{x,y}^s)$ ,  $\|Q_{i_1}\|_p, \|Q_{i_2}\|_p, \|K_{j_1}\|_p, \|K_{j_2}\|_p \leq \beta_1$  and  $\|Q_{i_1} - Q_{i_2}\|_q, \|K_{j_1} - K_{j_2}\|_q \leq \beta_2$  where  $\frac{1}{p} + \frac{1}{q} = 1$ , then  $\mathcal{P}_{i,j} \in [a, a+r]$  where  $r = 2\beta_1\beta_2$  for all  $(i, j) \in \text{supp}(B_{x,y}^s)$  and some  $a$ , and*

$$0 \leq \mu_{s,x,y}^* - \mu_{s,x,y} \leq C_r \mu_{s,x,y}$$

where  $C_r = 1 + \exp(r) - 2 \exp(r/2)$ .

Lemma 4.1 suggests that the approximation error depends on the ‘‘spread’’ or numerical range  $r$  of values (range, for short) in the  $\mathcal{P}_{i,j}$  entries within a region  $\text{supp}(B_{x,y}^s)$  and  $\mu_{s,x,y}$ . If  $r$  is small or  $\mu_{s,x,y}$  is small, then the approximation error is small. The range of a region is influenced by properties of  $Q$  and  $K$ . The range  $r$  is bounded by the norm and spread of  $Q_i$  and  $K_j$  for  $(i, j) \in \text{supp}(B_{x,y}^s)$ . This relies on the locality assumption that spatially nearby tokens should also be semantically similar which commonly holds in many applications. Of course, this can be avoided if needed – it is easy to reduce the spread of  $Q_i$  and  $K_j$  in local regions simply by permuting the order of  $Q$  and  $K$ . For example, we can use Locality Sensitive Hashing (LSH) to reorder  $Q_i$  and  $K_j$  such that similar vectors are in nearby positions, e.g., see (Kitaev et al., 2020). While the range  $r$  is data/problem dependent, we can control the range by using a smaller  $s$  since the range of a smaller region will be smaller. In the extreme case, when  $s = 1$ , the range is 0. So, this offers guidance that when  $\mu_{s,x,y}$  is large, we should approximate the region at a higher resolution such that the range is smaller.

*Remark 4.2.* Observe that the numerical range, which is defined as a bound on finite differences over sets of indices, is closely related to the concept of smoothness, which is defined using finite differences amongst adjacent indices. Indeed, it is possible to adapt Lemma 4.1 and its proof to the theory of wavelets, which are useful for characterizing signal smoothness. Please see §A.5 for more details.

*Remark 4.3.* The underlying assumption of diagonal attention structure from Longformer, Big Bird, and H-Transformer-1D is that tokens are highly dependent on the nearby tokens and *only* the nearby tokens, which is more

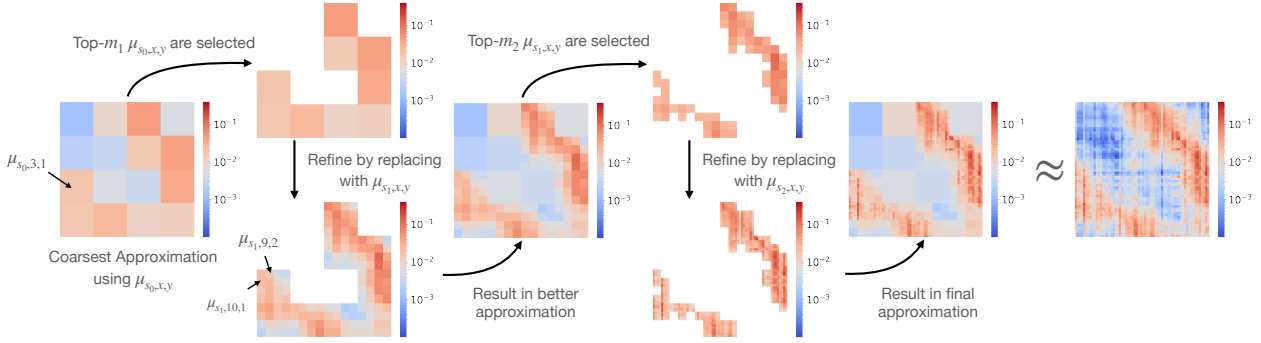


Figure 3. Illustration of our approximation scheme for  $R = \{16, 4, 1\}$ . A log scale is used for a better visualization. A linear scale visualization is shown in §A.1.

important than attention w.r.t. distant tokens. This might appear similar to the locality assumption discussed earlier, but this is incorrect. Our locality does *not* assume that semantically similar tokens must be spatially close, i.e., we allow high and precise dependence on distant tokens.

#### 4.2. Can we construct $\mathcal{J}$ quickly?

So far, we have assumed that the set  $\mathcal{J}$  is given which is not true in practice. We now place a mild restriction on the set  $\mathcal{J}$  as a simplification heuristic. We allow each  $(i, j)$  entry of  $\hat{A}$  to be included in the support of exactly one  $B_{x,y}^s \in \mathcal{J}$ . Consider a  $B_{x,y}^s \in \mathcal{J}$ , if each entry of the support region of  $B_{x,y}^s$  is included in the support of some  $B_{x',y'}^{s'} \in \mathcal{J}$  with a smaller  $s'$ , then  $B_{x,y}^s$  can be safely removed from  $\mathcal{J}$  without affecting the approximation, by construction. This restriction allows us to avoid searching for the  $B_{x,y}^s$  with the smallest  $s$  among multiple candidates. Then, the overall approximation can be written as

$$\hat{A} = \sum_{B_{x,y}^s \in \mathcal{J}} \mu_{s,x,y} B_{x,y}^s$$

*Remark 4.4.* Under this restriction,  $\mathcal{J}$  is a subset of an orthogonal basis of  $\mathbb{R}^{n \times n}$ .

**Mechanics of constructing  $\mathcal{J}$ .** Now, we can discuss how the set  $\mathcal{J}$  is constructed. Let us first consider the approximation error  $\mathcal{E} = \|\hat{A} - \mathcal{A}\|_F^2$ , by factoring out  $\mu_{s,x,y}^2$ ,

$$\mathcal{E} = \sum_{B_{x,y}^s \in \mathcal{J}} \mu_{s,x,y}^2 \sum_{(i,j) \in \text{supp}(B_{x,y}^s)} \left( \frac{\mathcal{A}_{i,j}}{\mu_{s,x,y}} - 1 \right)^2 \quad (8)$$

Since the goal is to minimize the error  $\mathcal{E}$ , the optimal solution is to fix the computation budget  $|\mathcal{J}|$  and solve the optimization problem which minimizes  $\mathcal{E}$  over all possible  $\mathcal{J}$ . However, this might not be efficiently solvable.

Instead, we consider finding a good solution greedily. Consider the error  $\mathcal{E}$ . We can analyze the specific term to get an insight into how approximation error can be reduced. Note that

$$\log \frac{\mathcal{A}_{i,j}}{\mu_{s,x,y}} = \mathcal{P}_{i,j} - \frac{1}{s^2} \langle B_{x,y}^s, \mathcal{P} \rangle$$

#### Algorithm 1 Constructing the set $\mathcal{J}$

---

**Input:**  $R = \{s_0, s_1, \dots, s_k\}$  in descending order  
**Input:** Budget  $m_i$  for each  $s_i$  for  $i > 0$   
**Input:** Initial  $\mathcal{J}$  (empty or prespecified via priors)  
 Compute  $\mu_{s_0,x,y}$  for all possible  $x, y$  and add  $B_{x,y}^{s_0}$  to  $\mathcal{J}$   
**for**  $i = 1$  **to**  $k$  **do**  
   Pop  $m_i$  elements  $B_{x,y}^{s_{i-1}}$  with the largest  $\mu_{s_{i-1},x,y}$   
   **for each**  $B_{x,y}^{s_{i-1}}$  **do**  
     Compute  $\mu_{s_i,x',y'}$  for all  $\text{supp}(B_{x',y'}^{s_i}) \subseteq \text{supp}(B_{x,y}^{s_{i-1}})$   
     Add  $B_{x',y'}^{s_i}$  to  $\mathcal{J}$   
   **end for**  
**end for**  
**Output:**  $\mathcal{J}$

---

is the deviation of  $\mathcal{P}_{i,j}$  from the mean of the support region, so the approximation error (8) is determined by  $\mu_{s,x,y}$  and the deviation of  $\mathcal{P}_{i,j}$  within the region, which coincides with the conclusion of Lemma 4.1. Computing this deviation would incur a  $O(s^2)$  cost, so we avoid using it as a criteria for  $\mathcal{J}$  construction. We found that we can make a reasonable assumption that the deviation of  $\mathcal{P}_{i,j}$  in a support of  $B_{x,y}^s$  for the *same*  $s$  are similar, and the deviation of a region for a smaller  $s$  is smaller. Then, a sensible heuristic is to use  $\mu_{s,x,y}$  as a criteria such that if  $\mu_{s,x,y}$  is large, then we must approximate the region using a higher resolution. The approximation procedure is described in Alg. 1, and the approximation result is shown in Fig. 3. Broadly, this approximation starts with a coarse approximation of a self-attention matrix, then for regions with a large  $\mu_{s,x,y}$ , we successively refines the regions to a higher resolution.

With the approximation procedure in place, we can quantify the error of this multiresolution approximation. We only show the approximation error for  $R = \{b, 1\}$  for some  $b$ , but the analysis easily extends beyond  $R = \{b, 1\}$ .

**Proposition 4.5.** *Let  $R = \{b, 1\}$  for some  $b$  and  $\delta$  be the  $m_1$ -th largest  $\mu_{b,x,y}$ , assume for all  $(i, j) \in \text{supp}(B_{x,y}^b)$ ,  $\mathcal{P}_{i,j} \in [a, a + r]$  for some  $a$  and  $r > 0$ , then*

$$\frac{\|\hat{A} - \mathcal{A}\|_F}{\|\mathcal{A}\|_F} \leq \sqrt{\frac{(n^2 - m_1 b^2) C_{2r} \delta^2}{\sum_{i,j=1}^n \exp(2\mathcal{P}_{i,j})}}$$

where  $C_{2r} = 1 + \exp(2r) - 2\exp(r)$

**Algorithm 2** Computing  $\hat{A}V$ 


---

**Input:**  $R = \{s_0, s_1, \dots, s_k\}$  in descending order  
**Input:**  $\mathcal{J}, \hat{A}, \tilde{V}_s$   
 Initialize  $Y_{s_{-1}} \in \mathbb{R}^{1 \times d}$  to be zero matrix  
**for**  $i = 0$  **to**  $i = k$  **do**  
     Duplicate rows of  $Y_{s_{i-1}}$  to create  $Y_{s_i} \in \mathbb{R}^{n/s_i \times d}$   
     **for** each  $B_{x,y}^{s_i} \in \mathcal{J}$  **do**  
          $(Y_{s_i})_x = (Y_{s_i})_x + \mu_{s_i,x,y}(\tilde{V}_{s_i})_y$   
     **end for**  
**end for**  
**Output:**  $Y_{s_k} = \hat{A}V$

---

Proposition 4.5 again emphasizes the relation between the numerical range  $r$  of  $\mathcal{P}$  and the quality of an approximation. With some knowledge of the range  $r$  and  $\mu_{b,x,y}$ , we can control the error using an appropriate budget  $m_1$ .

*Remark 4.6.* The procedure shares some commonalities with the correction component of Geometric Multigrid methods (Saad, 2003; Hackbusch, 1985). Coarsening is similar to our low resolution approximation, but the prolongation step is different. Rather than interpolate the entire coarse grid to finer grids, our method replaces some regions of coarse grid with its higher resolution approximation.

### 4.3. How do we compute $\hat{A}V$ ?

We obtained an approximation  $\hat{A}$ , but we should not instantiate this matrix (to avoid the  $O(n^2)$  cost). So, we discuss a simple procedure for computing  $\hat{A}V$  without constructing the  $n \times n$  matrix. Define  $\tilde{V}_s \in \mathbb{R}^{n/s \times d}$  where  $\tilde{V}_1 = V$  and

$$(\tilde{V}_s)_i = \frac{1}{2}(\tilde{V}_{s/2})_{2i-1} + \frac{1}{2}(\tilde{V}_{s/2})_{2i}$$

similar to (7). Then, the steps follow Alg. 2. We again start with multiplying coarse components of  $\hat{A}$  with  $V$ , then successively add the multiplication of higher resolution components of  $\hat{A}$  and  $V$ , and finally compute  $\hat{A}V$ .

### 4.4. What is the overall complexity?

We have now described the overall procedure of our approximation approach. In this section, we analyze the complexity of our procedure. Following convention in efficient self-attention papers, we treat  $d$  as a constant and it does not influence the complexity.

We first need to compute  $\tilde{Q}_s, \tilde{K}_s, \tilde{V}_s$  for  $s \in \{2, 4, \dots, n\}$ . Since each row of  $\tilde{Q}_s, \tilde{K}_s, \tilde{V}_s \in \mathbb{R}^{n/s \times d}$  requires averaging over two rows from  $\tilde{Q}_{s/2}, \tilde{K}_{s/2}, \tilde{V}_{s/2}$ , the total cost of computing  $\tilde{Q}_s, \tilde{K}_s, \tilde{V}_s$  for all  $s$  is simply  $O(\frac{n}{2} + \frac{n}{4} + \dots + \frac{n}{n}) = O(n)$ .

Given all  $\tilde{Q}_s, \tilde{K}_s$ , in Alg. 1, there are  $O((n/s_0)^2)$  possible entries of  $\mu_{s_0,x,y}$ . And at scale  $s_i$  for  $i > 0$ , there are  $O(m_i(s_{i-1}/s_i)^2)$  entries of  $\mu_{s_i,x,y}$  since there are  $O((s_{i-1}/s_i)^2)$  number of  $B_{x',y'}^{s_i}$  satisfying

$\text{supp}(B_{x',y'}^{s_i}) \subseteq \text{supp}(B_{x,y}^{s_{i-1}})$  and there are  $m_i$  regions at scale  $s_{i-1}$  to be refined. Note that computing each  $\mu_{s,x,y}$  takes  $O(1)$ , and selecting top-k elements is linear in the input size. Therefore, the cost of constructing  $\mathcal{J}$  is  $O((n/s_0)^2 + \sum_{i=1}^k m_i(s_{i-1}/s_i)^2)$ . Once  $\mathcal{J}$  is constructed,  $\hat{A}$  is simple since  $\hat{A}_{i,j} = \mu_{s,x,y}$  for a  $B_{x,y}^s \in \mathcal{J}$ .

Finally, multiplying  $\hat{A}$  and  $V$  in Alg. 2 takes  $O(n + (n/s_0)^2 + \sum_{i=1}^k m_i(s_{i-1}/s_i)^2)$ , also. The cost of creating a  $Y_{s_i}$  is  $O(n/s_i)$ , so the cost of creating all  $Y_s$  for  $s \in \{s_0, \dots, s_k\}$  is  $O(\frac{n}{s_0} + \dots + \frac{n}{s_k}) = O(n)$ . Then, for each  $B_{x,y}^s \in \mathcal{J}$ , adding  $\mu_{s,x,y}(\tilde{V}_s)_y$  to  $(Y_s)_x$  takes  $O(1)$ . The size of  $\mathcal{J}$  is  $O((n/s_0)^2 + \sum_{i=1}^k m_i(s_{i-1}/s_i)^2)$ , so the total complexity of Alg. 2 is as stated.

Therefore, the total complexity of our approach is  $O(n + (n/s_0)^2 + \sum_{i=1}^k m_i(s_{i-1}/s_i)^2)$ . For example, when  $R = \{\sqrt{n}, 1\}$ , the complexity becomes  $O(m_1 n)$ . The parameter  $m_1$  adjusts the trade-off between approximation accuracy and runtime similar to other efficient methods, e.g., window size  $w$  in  $O(wn)$  for Longformer (Beltagy et al., 2020) and projection size  $p$  in  $O(pn)$  for Linformer (Wang et al., 2020) or Performer (Choromanski et al., 2021).

## 5. Experiments

We perform a broad set of experiments to evaluate the practical performance profile of our MRA-based self-attention module. **First**, we compare our approximation accuracy with several other baselines. Then, we evaluate our method on the RoBERTa language model pretraining (Liu et al., 2019) and downstream tasks on both short and long sequences. Finally, as is commonly reported in most evaluations of efficient self-attention methods, we discuss our evaluations on the Long Range Arena (LRA) benchmark (Tay et al., 2021). All hyperparameters are reported in §A.4.

**Overview.** Since the efficiency is a core focus of efficient self-attention methods, time and memory efficiency is taken into account when evaluating performance. Whenever possible, we include runtime and memory consumption of a single instance for each method alongside the accuracy it achieves (in each table). Since the models are exactly the same (except which self-attention module is used), we only profile the efficiency of one training step consumed by these modules. See §A.4 for more details on profiling.

**Baselines.** For a rigorous comparison, we use an extensive list of baselines, including Linformer (Wang et al., 2020), Performer (Choromanski et al., 2021), Nyströmformer (Xiong et al., 2021), SOFT (Lu et al., 2021), YOSO (Zeng et al., 2021), Reformer (Kitaev et al., 2020), Longformer (Beltagy et al., 2020), Big Bird (Zaheer et al., 2020), H-Transformer-1D (Zhu & Soricut, 2021), and Scatterbrain (Chen et al., 2021). Since Nyströmformer, SOFT, and

YOSO also have a variant which involves convolution, we perform evaluations for both cases. We use our multiresolution approximation with  $R = \{32, 1\}$  for our method denoted in experiments as MRA-2. Further, we found that in tasks with limited dataset sizes, sparsity provides a regularization towards better performance. So, we include a MRA-2-s, which only computes

$$\hat{A}_1 = \sum_{B_{x,y}^1 \in \mathcal{J}} \mu_{1,x,y} B_{x,y}^1$$

after finding  $\mathcal{J}$ . We use different method-specific hyperparameters for some methods to better understand their efficiency-performance trade off. **Takeaway:** These detailed comparisons suggest that our MRA-based self-attention offers top performance and top efficiency among the baselines.

### 5.1. How good is the approximation accuracy?

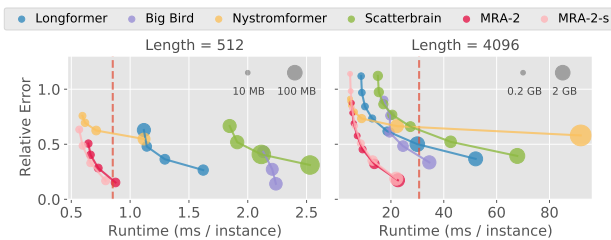


Figure 4. Approximation Error vs Runtime vs Memory. Red/dotted vertical line is the runtime of standard self-attention. Note that for any points to the right of the red vertical line, the approximation is slower than computing the true self-attention.

We show that our method give the best trade-off between approximation accuracy and efficiency by a significant margin compared to other baselines. The approximation accuracy of each method, compared to the standard self-attention, provides us a direct indication of the performance of approximation methods. To evaluate accuracy, we use 512 and 4096 length  $Q$ ,  $K$ , and  $V$  from a pretrained model and compute the relative error  $\|\hat{D}\hat{A}V - DAV\|_F / \|DAV\|_F$ . As shown in Fig. 4, our MRA-2(-s) has the lowest approximation error while maintaining the fastest runtime and smallest memory consumption by a large margin compared to other baselines in both short and long sequences. See §A.4 for more details, sequence lengths, and baselines.

Next, we evaluate the effect of the spread (or entropy) of self-attention on the approximation for different methods. The result is shown in Fig. 5. We see one limitation of low rank or sparsity-based schemes (discussed in §A.2 and Chen et al. (2021)). Our MRA-2 performs well across attention instances with different entropy settings and significantly better than Scatterbrain (Chen et al., 2021).

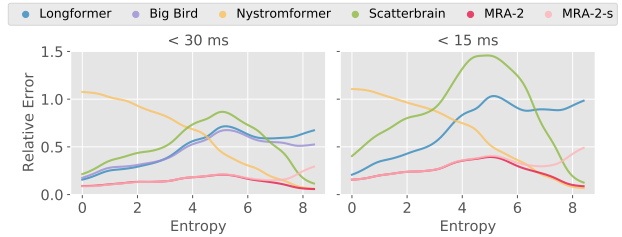


Figure 5. Entropy vs Approximation Error plots. Hyperparameters of each method is set such that the runtime is  $< 30ms$  and  $< 15ms$ , resp. The fastest setting of Big Bird is still  $> 15ms$ , so omitted from the second plot. Note that the runtime of standard self-attention is roughly 30ms.

Method	Time ms	Mem MB	MLM		MNLI	
			Before	After	m	mm
Transformer	<b>0.86</b>	71.0	<b>73.1</b>	<b>74.0</b>	<b>87.4</b>	<b>87.3</b>
Performer	1.29	62.8	6.8	63.1	32.7	33.0
Linformer	<b>0.74</b>	54.5	1.0	5.6	35.4	35.2
SOFT	<b>0.86</b>	34.0	10.9	25.0	32.7	33.0
SOFT + Conv	1.02	35.5	1.0	65.5	74.9	75.0
Nystromformer	<b>0.71</b>	34.8	17.2	68.2	35.4	35.2
Nystrom + Conv	0.88	37.2	1.4	70.9	85.1	84.6
YOSO	0.97	<b>29.8</b>	13.0	68.4	35.4	35.2
YOSO + Conv	1.20	<b>32.9</b>	3.0	69.0	83.2	83.1
Reformer	1.23	59.4	0.7	69.5	84.9	85.0
Longformer	1.30	43.3	66.0	71.2	85.6	85.4
	2.31	62.5	<b>71.9</b>	<b>73.2</b>	<b>87.0</b>	<b>87.1</b>
Big Bird	2.03	63.9	<b>71.6</b>	<b>73.3</b>	<b>87.1</b>	<b>87.0</b>
H-Transformer-1D	0.97	<b>29.3</b>	0.5	6.1	35.4	35.2
Scatterbrain	2.23	78.7	60.6	-	-	-
MRA-2	<b>0.73</b>	<b>28.1</b>	<b>68.9</b>	<b>73.1</b>	<b>86.8</b>	<b>87.1</b>
	0.86	34.3	<b>71.9</b>	<b>73.8</b>	<b>87.1</b>	<b>87.2</b>
MRA-2-s	<b>0.66</b>	<b>23.8</b>	<b>67.2</b>	72.8	<b>87.0</b>	<b>87.0</b>
	<b>0.80</b>	<b>29.1</b>	<b>71.8</b>	<b>73.8</b>	<b>87.4</b>	<b>87.4</b>

Table 1. Summary of 512 length RoBERTa-base models: runtime and memory efficiency, MLM accuracy, and MNLI accuracy. Unit for time (and memory) is ms (and MB). Before/After denotes accuracy before/after finetuning. The m/mm give the matched/mismatched MNLI. Some methods have more than one row for different model-specific hyperparameters. We divide measurements into three ranked groups for visualization (bold, normal, transparent).

### 5.2. RoBERTa Language Modeling

Here, we use RoBERTa language modeling (Liu et al., 2019) to assess the performance and efficiency trade off of our method and baselines. We use a pretrained RoBERTa-base to evaluate the compatibility of each method with the existing Transformer models and overall feasibility for direct deployment. For fair comparisons, we also check the performance of models trained from scratch. Then, MNLI (Williams et al., 2018) is used to test the model’s ability on downstream tasks. Further, we extend the 512 length models to 4096 length for a set of best performing methods and use the WikiHop (Welbl et al., 2018) task as an assessment on long sequence language models.

**Standard Sequence Length.** Since efficient self-attention approximates standard self-attention, we could simply substitute the standard self-attention of a trained model. This would allow us to minimize the training cost for new methods. To evaluate compatibility with the existing models, we use a pretrained 512 length RoBERTa-base model (Liu et al., 2019) and replace its self-attention module with efficient alternatives and measure the validation Masked Language Modeling (MLM) accuracy. Then, we check accuracy after finetuning the model on English Wikipedia and Bookcorpus (Zhu et al., 2015). Eventually, we finetune the model on the downstream task MNLI (Williams et al., 2018).

Method	Time ms	Mem MB	MLM	MNLI	
				m	mm
Transformer	<b>0.41</b>	35.47	<b>57.0</b>	72.7±0.6	73.8±0.2
Performer	0.63	31.38	48.6	69.8±0.4	70.5±0.1
Linformer	<b>0.35</b>	27.23	53.5	72.5±0.8	73.2±0.4
SOFT	0.43	17.02	42.8	63.8±2.2	64.7±2.6
SOFT + Conv	0.53	17.77	56.7	70.8±0.5	71.8±0.4
Nystromformer	<b>0.34</b>	17.40	53.1	71.4±0.6	72.0±0.3
Nystrom + Conv	0.45	18.60	<b>57.3</b>	73.0±0.4	73.9±0.6
YOSO	0.47	<b>14.91</b>	53.4	72.9±0.8	73.2±0.4
YOSO + Conv	0.58	<b>16.42</b>	<b>57.2</b>	72.5±0.4	72.9±0.5
Reformer	<b>0.39</b>	<b>16.43</b>	52.4	<b>73.7±0.4</b>	<b>74.6±0.3</b>
	0.61	29.65	55.6	<b>75.0±0.2</b>	<b>75.6±0.3</b>
Longformer	0.61	21.60	54.7	72.0±0.4	73.5±0.2
	1.10	31.44	<b>57.4</b>	<b>75.8±0.5</b>	<b>76.7±0.6</b>
Big Bird	1.02	31.91	<b>57.6</b>	<b>75.0±0.5</b>	<b>75.6±0.6</b>
H-Transformer-1D	0.47	<b>14.65</b>	43.7	62.9±2.7	63.4±3.9
Scatterbrain	1.04	78.66	20.5	42.6±8.1	43.4±9.5
MRA-2	<b>0.36</b>	<b>14.05</b>	56.4	<b>73.2±0.2</b>	<b>74.1±0.5</b>
	0.43	17.15	<b>57.3</b>	73.0±1.0	73.9±0.8
MRA-2-s	<b>0.31</b>	<b>11.93</b>	56.7	<b>73.6±1.6</b>	<b>74.3±1.1</b>
	<b>0.38</b>	<b>14.57</b>	<b>57.5</b>	<b>73.9±0.6</b>	<b>74.6±0.8</b>

Table 2. Summary of 512 length RoBERTa-small models. We also include a 95% error bar for experiments that have a small compute burden.

Only a handful of schemes including Longformer, Big Bird, and MRA-2(-s) are fully compatible with pretrained models. Scatterbrain has a reasonable accuracy without further finetuning, but the training diverges when finetuning the model. The other methods cannot get a satisfactory level of accuracy. These statements also hold for the downstream finetuning results, shown in Tab. 1. Our method has the best performance among baselines for both MLM and MNLI. Meanwhile, it has a much better time and memory efficiency.

Since many baselines are not compatible with the trained model weights (performance degrades when substituting the self-attention module), to make the comparison fair for all methods, we also evaluate models trained from scratch. Due to the large number of baselines we use, we train a small variant of RoBERTa on English Wikipedia and BookCorpus (Zhu et al., 2015) to keep the training cost reasonable. Then, we again finetune the model on downstream task (MNLI).

Method	Time (ms)	Mem (GB)	MLM	WikiHop
Transformer	30.88	3.93	<b>74.3</b>	<b>74.6</b>
Longformer	10.20	0.35	71.1	60.8
Big Bird	17.53	0.59	-	-
MRA-2	<b>7.03</b>	<b>0.28</b>	73.1	71.2
	9.25	0.38	<b>73.7</b>	<b>73.4</b>
MRA-2-s	<b>6.37</b>	<b>0.23</b>	73.0	71.8
	<b>8.62</b>	0.38	<b>73.8</b>	<b>74.1</b>

Table 3. Summary of 4096 length RoBERTa-base models. Since Big Bird is slow and we are not able to reduce its training time using multiple GPU, we cannot test Big Bird for 4096 sequence.

Results are summarized in Tab. 2. Only a few methods (including ours) achieve both good performance and efficiency.

Method	Time (ms)	Mem (GB)	MLM	WikiHop
Transformer	15.36	1.96	<b>55.8</b>	<b>54.6±1.6</b>
Performer	5.13	0.24	23.2	43.7±0.6
Linformer	<b>2.85</b>	0.21	13.8	11.0±0.4
SOFT	<b>2.46</b>	<b>0.11</b>	25.9	14.0±8.6
	5.92	0.24	31.0	12.1±1.9
SOFT + Conv	<b>3.33</b>	<b>0.11</b>	52.8	30.8±29.3
Nystromformer	<b>2.38</b>	<b>0.11</b>	34.7	44.0±0.2
	4.34	0.27	46.8	46.0±0.8
Nystrom + Conv	<b>3.23</b>	<b>0.12</b>	53.1	<b>54.6±0.8</b>
YOSO	4.15	<b>0.12</b>	47.8	52.4±0.1
	5.07	0.17	49.9	52.8±0.5
YOSO + Conv	5.45	0.13	<b>55.1</b>	<b>53.2±0.7</b>
Reformer	5.04	0.24	52.2	<b>53.7±0.9</b>
Longformer	4.88	0.17	52.4	52.3±0.7
Big Bird	8.68	0.29	<b>54.4</b>	<b>54.3±0.7</b>
H-Transformer-1D	3.93	<b>0.12</b>	41.1	43.7±0.7
Scatterbrain	8.83	0.31	35.8	12.1±0.9
MRA-2	<b>3.43</b>	0.14	<b>54.2</b>	52.6±0.9
	4.52	0.19	<b>55.2</b>	<b>54.0±0.9</b>
MRA-2-s	<b>3.12</b>	<b>0.12</b>	<b>53.8</b>	51.8±0.9
	4.13	0.19	<b>55.1</b>	<b>53.6±0.8</b>

Table 4. Summary of 4096 length RoBERTa-small models.

**Longer Sequences.** To evaluate the performance of our MRA-2(-s) on longer sequences, we extend the 512 length models to 4096 length. We extend the positional embedding and further train the models on English Wikipedia, Bookcorpus (Zhu et al., 2015), one third of Stories dataset (Trinh & Le, 2018), and one third of RealNews dataset (Zellers et al., 2019) following (Beltagy et al., 2020). Then, the 4096 length models are finetuned on WikiHop dataset (Welbl et al., 2018) to assess the performance of these models on downstream tasks. The results are summarized in Tab. 3 for base models and Tab. 4 for small models. Our MRA-2 is again one of the top performing methods with high efficiency among baselines. Note that the difference in WikiHop performance of Longformer (Beltagy et al., 2020) from the original paper is due to a much larger window size which has an even slower runtime. Linformer (Wang et al., 2020) does not seem to be able to adapt the weights from its 512 length model to a 4096 model. It is interesting that the convolution in Nystromformer (Xiong et al., 2021) seems



to play an important role in boosting performance.

### 5.3. Long Range Arena

Method	Listops	Text	Retrieval	Image	Pathfinder	Avg
Transformer	37.1±0.4	<b>65.2±0.6</b>	79.6±1.7	38.5±0.7	<b>72.8±1.1</b>	58.7±0.3
Performer	36.7±0.2	65.2±0.9	79.5±1.4	38.6±0.7	71.4±0.7	58.3±0.1
Linformer	<b>37.4±0.3</b>	57.0±1.1	78.4±0.1	38.1±0.3	67.2±0.1	55.6±0.3
SOFT	36.3±1.4	65.2±0.0	<b>83.3±1.0</b>	35.3±1.3	67.7±1.1	57.5±0.5
SOFT + Conv	37.1±0.4	65.2±0.4	<b>82.9±0.0</b>	37.1±4.7	68.1±0.4	58.1±0.9
Nystromformer	24.7±17.5	<b>65.7±0.1</b>	<b>80.2±0.3</b>	38.8±2.9	<b>73.1±0.1</b>	56.5±2.8
Nystrom + Conv	30.6±8.9	<b>65.7±0.2</b>	78.9±1.2	<b>43.2±3.4</b>	69.1±1.0	57.5±1.5
YOSO	37.0±0.3	63.1±0.2	78.3±0.7	40.8±0.8	<b>72.9±0.6</b>	58.4±0.3
YOSO + Conv	<b>37.2±0.5</b>	64.9±1.2	78.5±0.9	<b>44.6±0.7</b>	69.5±3.5	<b>59.0±1.1</b>
Reformer	18.9±2.4	64.9±0.4	78.2±1.6	<b>42.4±0.4</b>	68.9±1.1	54.7±0.2
Longformer	37.2±0.3	64.1±0.1	79.7±1.1	<b>42.6±0.1</b>	70.7±0.8	<b>58.9±0.1</b>
Big Bird	<b>37.4±0.3</b>	64.3±1.1	79.9±0.1	40.9±1.1	<b>72.6±0.7</b>	<b>59.0±0.3</b>
H-Transformer-ID	30.4±8.8	<b>66.0±0.2</b>	<b>80.1±0.4</b>	<b>42.1±0.8</b>	70.7±0.1	57.8±1.8
Scatterbrain	<b>37.5±0.1</b>	64.4±0.3	79.6±0.1	38.0±0.9	54.8±7.8	54.9±1.4
MRA-2	37.2±0.3	<b>65.4±0.1</b>	79.6±0.6	39.5±0.9	<b>73.6±0.4</b>	<b>59.0±0.3</b>
MRA-2-s	<b>37.4±0.5</b>	64.3±0.8	<b>80.3±0.1</b>	41.1±0.4	<b>73.8±0.6</b>	<b>59.4±0.2</b>

Table 5. Test set accuracy of LRA tasks. Since the benchmark consist of multiple tasks with different sequence length, we do not include the efficiency components in the table.

The Long Range Arena (LRA) (Tay et al., 2021) has been proposed to provide a lightweight benchmark to quickly compare the capability of long sequence modeling for Transformers. Due to a consistency issue and code compatibility of official LRA benchmark (see Issue-34, Issue-35, and Lee-Thorp et al. (2021)), we use the LRA code provided by (Xiong et al., 2021) and follow exactly the same hyperparameter setting. The results are shown in Tab. 5. Our method has the best performance compared to others.

**Caveats.** A reader may ask why Longformer, Big Bird, and MRA-2-s perform better than standard Transformers (Vaswani et al., 2017) despite being approximations. The performance difference is most obvious on the image task. We also found that Longformer with a smaller local attention window (256, 128, 64) tends to offer better performance (39.52, 42.11, 42.71, respectively) on the image task. One reason is that standard self-attention needs larger datasets to compensate for its lack of locality bias (Xu et al., 2021; d’Ascoli et al., 2021). Hence, due to the small datasets (i.e., its lightweight nature), the LRA accuracy metrics should be interpreted with caution.

**ImageNet.** To test the performance on large datasets, we use ImageNet (Russakovsky et al., 2015) as a large scale alternative to CIFAR-10 (Krizhevsky et al., 2009) used in image task of LRA. Further, data augmentation is used to increase the dataset size. Like LRA, we focus on small models and use a 4-layer Transformer (see §A.4 for more details). Model specific hyperparameters are the same as the ones used on LRA. The results are shown in Tab. 6. MRA-

Method	Time (ms)	Mem (MB)	Top-1	Top-5
Transformer	1.24	45.5	48.7	73.7
Reformer	1.14	19.1	39.6	65.5
Longformer	1.12	13.7	<b>49.1</b>	73.9
H-Transformer-ID	1.03	<b>9.8</b>	48.7	<b>73.9</b>
MRA-2	<b>1.00</b>	11.8	48.9	73.6
MRA-2-s	<b>0.98</b>	<b>9.7</b>	<b>49.2</b>	<b>73.9</b>

Table 6. Summary of ImageNet results trained on 4-layer Transformers. We reports both top-1 and top-5 accuracy.

2-s is the top performing approach. Standard self-attention and MRA-2 can clearly perform better on a large dataset.

## 6. Conclusion

We show that Multiresolution analysis (MRA) provides fresh ideas for efficiently approximating self-attention, which subsumes many piecemeal approaches in the literature. We expect that exploiting the links to MRA will allow leveraging a vast body of technical results developed over many decades. But we show that there are tangible practical benefits available immediately. When some consideration is given to which design choices or heuristics for a MRA-based self-attention scheme will interface well with mature software stacks and modern hardware, we obtain a procedure with strong advantages across both *performance/accuracy and efficiency*. Further, our implementation can be directly plugged into existing Transformers, a feature missing in some existing efficient transformer implementations. We show use cases on longer sequence tasks and in resource limited setting but believe that various other applications of Transformers will also benefit in the short term. Finally, we should note the lack of integrated software support for MRA as well as our specialized model in current deep learning libraries. Overcoming this limitation required implementing custom CUDA kernels for some generic block sparsity operators. Therefore, extending our algorithm for other use cases may involve reimplementing the kernel. We hope that with broader use of MRA-based methods, the software support will improve thereby reducing this implementation barrier.

## Acknowledgments

This work was supported by the UW AmFam Data Science Institute through funds from American Family Insurance. VS was supported by NIH RF1 AG059312. We thank Sathya Ravi for discussions regarding multigrid methods, Karu Sankaralingam for suggestions regarding hardware support for sparsity/block sparsity, and Pranav Pulijala for integrating our algorithm within the HuggingFace library.

## References

- Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.
- Candès, E. J., Li, X., Ma, Y., and Wright, J. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3): 1–37, 2011.
- Chen, B., Dao, T., Winsor, E., Song, Z., Rudra, A., and Ré, C. Scatterbrain: Unifying sparse and low-rank attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- Choromanski, K. M., Likhoshesterov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with performers. In *International Conference on Learning Representations (ICLR)*, 2021.
- Coifman, R. R. and Maggioni, M. Diffusion wavelets. *Applied and computational harmonic analysis*, 21(1):53–94, 2006.
- d’Ascoli, S., Touvron, H., Leavitt, M., Morcos, A., Biroli, G., and Sagun, L. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning (ICML)*, 2021.
- Daubechies, I. *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1992.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.
- Donoho, D. L. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.
- Donoho, D. L. and Johnstone, I. M. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995.
- Gavish, M., Nadler, B., and Coifman, R. R. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. In *The International Conference on Machine Learning (ICML)*, 2010.
- Hackbusch, W. *Multi-Grid Methods and Applications*, volume 4. Springer Science & Business Media, 01 1985.
- Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.
- Hu, S. Relations of the nuclear norm of a tensor and its matrix flattenings. *Linear Algebra and its Applications*, 478:188–199, 2015.
- Ithapu, V. K., Kondor, R., Johnson, S. C., and Singh, V. The incremental multiresolution matrix factorization algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *International Conference on Learning Representations (ICLR)*, 2020.
- Kondor, R., Teneva, N., and Garg, V. Multiresolution matrix factorization. In *International Conference on Machine Learning (ICML)*, 2014.
- Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. Revealing the dark secrets of bert. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.
- Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.
- Lee, A. B. and Nadler, B. Treelets: A tool for dimensionality reduction and multi-scale analysis of unstructured data. In *International Conference on Artificial Intelligence and Statistics*, 2007.
- Lee-Thorp, J., Ainslie, J., Eckstein, I., and Ontanon, S. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- Lu, J., Yao, J., Zhang, J., Zhu, X., Xu, H., Gao, W., Xu, C., Xiang, T., and Zhang, L. Soft: Softmax-free transformer with linear complexity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- Mallat, S. *A Wavelet Tour of Signal Processing*. Elsevier, 1999.
- Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N., and Kong, L. Random feature attention. In *International Conference on Learning Representations (ICLR)*, 2021.
- Peng, X., Lu, C., Yi, Z., and Tang, H. Connections between nuclear-norm and frobenius-norm-based representations. *IEEE Transactions on Neural Networks and Learning Systems*, 29(1):218–224, 2018.

- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. In *International Journal of Computer Vision (IJCV)*, 2015.
- Saad, Y. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.
- Simic, S. Jensen’s inequality and new entropy bounds. *Applied Mathematics Letters*, 22(8):1262–1265, 2009.
- Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations (ICLR)*, 2021.
- Trinh, T. H. and Le, Q. V. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*, 2018.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- Wang, S., Li, B., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.
- Welbl, J., Stenetorp, P., and Riedel, S. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018.
- Williams, A., Nangia, N., and Bowman, S. A broad-coverage challenge corpus for sentence understanding through inference. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
- Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.
- Xu, Y., Zhang, Q., Zhang, J., and Tao, D. Vitae: Vision transformer advanced by exploring intrinsic inductive bias. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., and Choi, Y. Defending against neural fake news. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- Zeng, Z., Xiong, Y., Ravi, S., Acharya, S., Fung, G. M., and Singh, V. You only sample (almost) once: Linear cost self-attention via bernoulli sampling. In *International Conference on Machine Learning (ICML)*, 2021.
- Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *International Conference on Computer Vision (ICCV)*, 2015.
- Zhu, Z. and Soricut, R. H-transformer-1D: Fast one-dimensional hierarchical attention for sequences. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

## A. Appendix

The appendix includes more details regarding the formulation, analysis, and experiments.

### A.1. Visualizing Approximation Procedure in Linear Scale

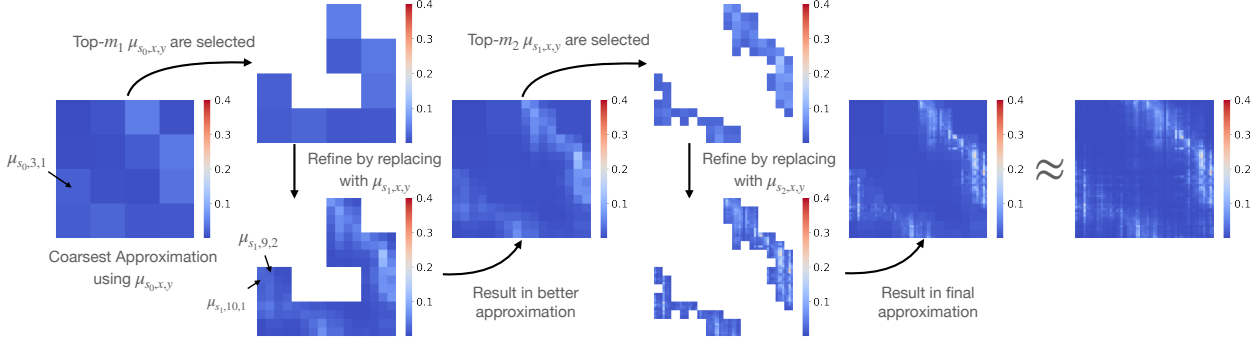


Figure 6. Illustration of approximation scheme for  $R = \{16, 4, 1\}$  in linear scale.

We also show a visualization of our approximation procedure using a linear scale in Fig. 6. This figure gives a better illustration of how approximation quality increases as approximation procedure proceeds.

### A.2. Link to Sparsity and Low Rank

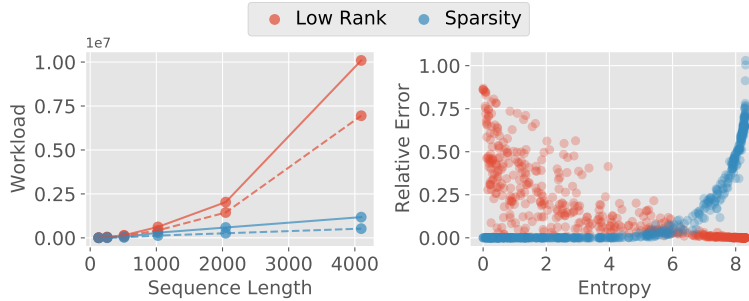


Figure 7. The solid and dashed lines in the left plot is the theoretical workload, without considering the overhead, necessary to get relative error less than 0.05 and 0.1, respectively, for different sequence length. Ideally, the workload should be linear to sequence length. The right plot is the approximation error vs entropy of self-attention matrices, similar to (Chen et al., 2021), at 1/4 of workload for standard self-attention (keeping 25% of rank and nonzero entries for low rank and sparsity, respectively). Entropy of softmax is used as a proxy for the spread of attention. Relative error is defines as  $\|\hat{D}\hat{A}\hat{V} - DAV\|_F / \|DAV\|_F$

Low rank and sparsity are two popular directions for efficient self-attention. To explore the potential of these two types of approximations, we set aside the efficiency consideration and use the best possible methods for each type of approximation. Specifically, subject to  $\|\hat{A} - \mathcal{A}\|_F \leq \epsilon$ , we use sparse approximation which minimizes  $\|\hat{A}\|_0$  by finding support on the largest entries of  $\mathcal{A}$ , and low rank approximation which minimizes  $\text{rank}(\hat{A})$  via truncated SVD. As shown in Fig. 7, these two types of methods are limited for approximating self-attention. The low rank method requires superlinear cost to maintain the approximation accuracy and fails when the entropy of self-attention is smaller. In many cases, sparse approximation is sufficient, but in some cases when the self-attention matrices are less sparse and have larger entropy, the sparse approximation would fail as well. This motivates the use of sparse + low rank approximation. It can be achieved via robust PCA which decomposes approximation to  $\hat{A} = S + L$  by solving an optimization objective  $\|S\|_0 + \lambda \text{rank}(L)$ . A convex relaxation  $\|S\|_1 + \lambda \|L\|_*$  (Candès et al., 2011) is used to make the optimization tractable, but the cost of finding a good solution is still more than  $O(n^2)$ , which is not suitable for efficient self-attention. Scatterbrain (Chen et al., 2021) proposes to combine an existing sparse attention with an existing low rank attention to obtain a sparse + low rank approximation and avoid the expensive cost of robust PCA.

Interestingly, a special form of our work offers an alternative to Scatterbrain’s approach for sparse + low rank approximation. When  $R = \{b, 1\}$  for some  $b$ , our MRA-2 can be viewed as a sparse + low rank approximation. Specifically, let

$$\hat{\mathcal{A}}_r = \sum_{B_{x,y}^b \in \mathcal{J}} \mu_{b,x,y} B_{x,y}^b$$

for a resolution  $b$ , then  $\hat{\mathcal{A}}_1 + \hat{\mathcal{A}}_b$  serves as a reasonably good solution for a relaxed version of sparse and low rank decomposition.

Let us consider an alternative relaxation of robust PCA objective,

$$\|S\|_0 + \lambda \|L\|_F \quad (9)$$

Note that  $\|L\|_F$  is easier to compute. And we have  $\|L\|_* \leq \sqrt{n} \|L\|_F$  (Hu, 2015). In fact, Peng et al. (2018) shows that solutions obtained by minimizing  $\|L\|_*$  and  $\|L\|_F$  are two solutions of a low rank recovery problem and are identical in some situations. The optimal solution for objective (9) can be easily obtained. For  $\epsilon = 0$ , there exists a  $m$  such that the optimal  $S$  has support on the  $m$  largest entries of  $\mathcal{A}$ , and  $L = \mathcal{A} - S$ . However, for practical use, the recovered sparsity cannot be efficiently used on a GPU due to scattered support. Further, the complexity is  $O(n^2)$  since we found that we still need to find the largest entries of  $\mathcal{A}$ . Suppose we restrict  $S$  to be a block sparse matrix, namely, supported on a subset of  $\{\text{supp}(B_{x,y}^b)\}_{x,y}$ , then a GPU can exploit this block sparsity structure to significantly accelerate computation. Further, the optimal  $S$  is supported on the regions with the largest  $\mu'_{b,x,y}$  defined as

$$\mu'_{b,x,y} = \frac{1}{b^2} \sum_{(i,j) \in \text{supp}(B_{x,y}^b)} \exp(2\mathcal{P}_{i,j}) \quad (10)$$

which is similar to (4). As a result, similar to approximating (4) with (6), we use the lower bound  $\mu_{b,x,y}^2$  as an proxy for (10). Then, the cost of locating support blocks is only  $O(n^2/b^2)$ . Consequently, the resulting solution  $S$  is supported on the regions with the largest  $\mu_{b,x,y}$ . Note that this  $S$  is exactly  $\hat{\mathcal{A}}_1$  with an appropriate budget  $m$ . And the  $\hat{\mathcal{A}}_b$  is a reasonable solution for  $L$  since  $\|\hat{\mathcal{A}}_b\|_F$  is small and  $\text{rank}(\hat{\mathcal{A}}_b) \leq n/b$ .

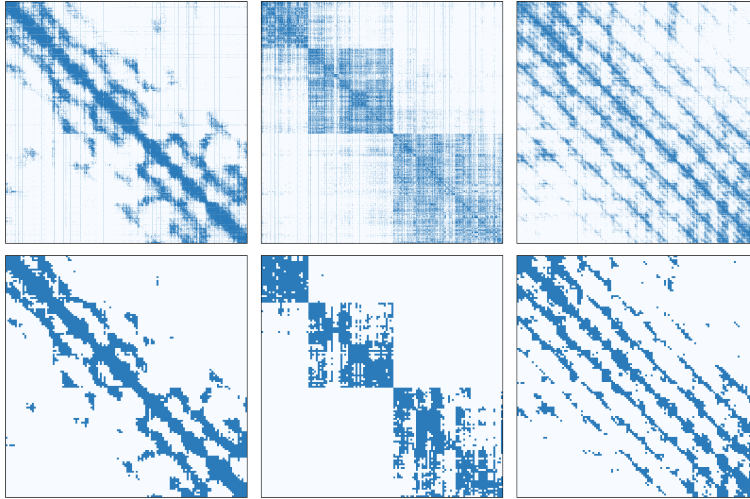


Figure 8. The top 3 plots are the optimal supports for 3 typical self-attention matrices at 80% sparsity. The bottom 3 plots are supports found via our MRA-2 at 80% sparsity.

We empirically evaluate the quality of this sparse solution. (Kovaleva et al., 2019) showed that the BERT model (Devlin et al., 2019) has multiple self-attention patterns capturing different semantic information. We investigate the types of possible self-attention of a pretrained RoBERTa model (Liu et al., 2019). And we show the optimal sparsity supports for self-attention matrices generated from RoBERTa-base with 4096 sequence in the top plots of Fig. 8. Our MRA-2, as shown in the bottom plots of Fig. 8, is able to find a reasonably good sparse solution for (9). Notice that while many self-attention matrices tend out to be diagonally banded matrices, which Longformer and Big Bird can approximate well, they are not the only possible structure. Diagonally banded structure is not sufficient to approximate the last two self-attention patterns well.

### A.3. Analysis

In this section, we discuss in more detail some manipulations described in the main paper.

**Observation A.1.** We can re-write  $\hat{\mathcal{A}}^* = \sum_{B_{x,y}^s \in \mathcal{J}} \alpha_{x,y}^s B_{x,y}^s$  as  $(\hat{\mathcal{A}}^*)_{i,j} = \mu_{s,x,y}^*$  where  $(s, x, y)$  is the index of  $B_{x,y}^s \in \mathcal{J}$  that has the smallest support region and is supported on  $(i, j)$ .

*Proof.* We describe the details next. First, notice that at each scale  $s$ ,

$$E_{s/2} = E_s - \sum_{B_{x,y}^s \in \mathcal{J}} \alpha_{x,y}^s B_{x,y}^s$$

If  $(i, j) \notin \text{supp}(B_{x,y}^s)$  for all  $B_{x,y}^s \in \mathcal{J}$ , then  $(\sum_{B_{x,y}^s \in \mathcal{J}} \alpha_{x,y}^s B_{x,y}^s)_{i,j} = 0$  and thus  $(E_{s/2})_{i,j} = (E_s)_{i,j}$ . Further, at each scale  $s$ , the supports of  $B_{x,y}^s$  are disjoint, and there is exactly one  $B_{x,y}^s$  whose support includes  $(i, j)$ . Thus, if  $(E_{s/2})_{i,j} = (E_s)_{i,j}$ , then  $(E_{s/2})_{i',j'} = (E_s)_{i',j'}$  for all  $(i', j') \in \text{supp}(B_{x,y}^s)$ . Also, for scale  $s' < s$ , if  $B_{x',y'}^{s'}$  is supported on  $(i, j)$ , then  $\text{supp}(B_{x',y'}^{s'}) \subseteq \text{supp}(B_{x,y}^s)$ . These observations are useful for the derivation below.

Consider approximation on the  $(i, j)$  entry of  $\hat{\mathcal{A}}^*$ , and let  $B_{x_1,y_1}^{s_1}, \dots, B_{x_k,y_k}^{s_k} \in \mathcal{J}$  be all components that are supported on  $(i, j)$  and  $s_1 < \dots < s_k$ . Then,  $E_n = \mathcal{A}$ . Since  $B_{x_k,y_k}^{s_k}$  has the largest  $s_k$  and is supported on  $(i, j)$ , by the above observations, for all  $(i', j') \in \text{supp}(B_{x_k,y_k}^{s_k})$ ,

$$(E_{s_k})_{i',j'} = (E_{2s_k})_{i',j'} = \dots = (E_n)_{i',j'} = \mathcal{A}_{i',j'}$$

Therefore,

$$\alpha_{x_k,y_k}^{s_k} = \frac{1}{s_k^2} \langle B_{x_k,y_k}^{s_k}, E_{s_k} \rangle = \frac{1}{s_k^2} \langle B_{x_k,y_k}^{s_k}, \mathcal{A} \rangle = \mu_{s_k,x_k,y_k}^*$$

Then,

$$(E_{s_k/2})_{i',j'} = (E_{s_k})_{i',j'} - \left( \sum_{B_{x_k,y_k}^{s_k} \in \mathcal{J}} (\alpha_{x_k,y_k}^{s_k})' B_{x_k,y_k}^{s_k} \right)_{i',j'} = \mathcal{A}_{i',j'} - \mu_{s_k,x_k,y_k}^*$$

Assume for all  $(i', j') \in \text{supp}(B_{x_{m+1},y_{m+1}}^{s_{m+1}})$ ,

$$(E_{s_{m+1}/2})_{i',j'} = \mathcal{A}_{i',j'} - \mu_{s_{m+1},x_{m+1},y_{m+1}}^*$$

Then, again by the above observations,

$$(E_{s_m})_{i',j'} = \dots = (E_{s_{m+1}/2})_{i',j'} = \mathcal{A}_{i',j'} - \mu_{s_{m+1},x_{m+1},y_{m+1}}^*$$

Therefore,

$$\begin{aligned} \alpha_{x_m,y_m}^{s_m} &= \frac{1}{s_m^2} \langle B_{x_m,y_m}^{s_m}, E_{s_m} \rangle \\ &= \frac{1}{s_m^2} \sum_{(i'',j'') \in \text{supp}(B_{x_m,y_m}^{s_m})} (\mathcal{A}_{i'',j''} - \mu_{s_{m+1},x_{m+1},y_{m+1}}^*) \\ &= \frac{1}{s_m^2} \sum_{(i'',j'') \in \text{supp}(B_{x_m,y_m}^{s_m})} \mathcal{A}_{i'',j''} - \mu_{s_{m+1},x_{m+1},y_{m+1}}^* \\ &= \mu_{s_m,x_m,y_m}^* - \mu_{s_{m+1},x_{m+1},y_{m+1}}^* \end{aligned}$$

Then,

$$\begin{aligned} (E_{s_m/2})_{i',j'} &= (E_{s_m})_{i',j'} - \left( \sum_{B_{x_m,y_m}^{s_m} \in \mathcal{J}} (\alpha_{x_m,y_m}^{s_m})' B_{x_m,y_m}^{s_m} \right)_{i',j'} \\ &= \mathcal{A}_{i,j} - \mu_{s_{m+1},x_{m+1},y_{m+1}}^* - (\mu_{s_m,x_m,y_m}^* - \mu_{s_{m+1},x_{m+1},y_{m+1}}^*) \\ &= \mathcal{A}_{i,j} - \mu_{s_m,x_m,y_m}^* \end{aligned}$$

By induction, for all  $(i', j') \in \text{supp}(B_{x_m, y_m}^{s_m})$ ,

$$(E_{s_m/2})_{i', j'} = \mathcal{A}_{i', j'} - \mu_{s_m, x_m, y_m}^*$$

holds for all  $s_m$ . Further, we know for  $m < k$ ,

$$\begin{aligned} \alpha_{x_k, y_k}^{s_k} &= \mu_{s_k, x_k, y_k}^* \\ \alpha_{x_m, y_m}^{s_m} &= \mu_{s_m, x_m, y_m}^* - \mu_{s_{m+1}, x_{m+1}, y_{m+1}}^* \end{aligned}$$

Finally,

$$(\hat{\mathcal{A}}^*)_{i, j} = \left( \sum_{B_{x, y}^s \in \mathcal{J}} \alpha_{x, y}^s B_{x, y}^s \right)_{i, j} = \mu_{s_k, x_k, y_k}^* + \sum_{m=1}^{k-1} (\mu_{s_m, x_m, y_m}^* - \mu_{s_{m+1}, x_{m+1}, y_{m+1}}^*) = \mu_{s_1, x_1, y_1}^*$$

And,  $(s_1, x_1, y_1)$  is the index of  $B_{x_1, y_1}^{s_1} \in \mathcal{J}$  that has the smallest support region and is supported on  $(i, j)$ . This completes the proof.  $\square$

**Lemma A.2.** Assume for all  $(i_1, j_1), (i_2, j_2) \in \text{supp}(B_{x, y}^s)$ ,  $\|Q_{i_1}\|_p, \|Q_{i_2}\|_p, \|K_{j_1}\|_p, \|K_{j_2}\|_p \leq \beta_1$  and  $\|Q_{i_1} - Q_{i_2}\|_q, \|K_{j_1} - K_{j_2}\|_q \leq \beta_2$  where  $\frac{1}{p} + \frac{1}{q} = 1$ , then  $\mathcal{P}_{i, j} \in [a, a + r]$  where  $r = 2\beta_1\beta_2$  for all  $(i, j) \in \text{supp}(B_{x, y}^s)$  and some  $a$ , and

$$0 \leq \mu_{s, x, y}^* - \mu_{s, x, y} \leq C_r \mu_{s, x, y}$$

where  $C_r = 1 + \exp(r) - 2 \exp(r/2)$ .

*Proof.*

$$\begin{aligned} \mathcal{P}_{i_1, j_1} - \mathcal{P}_{i_2, j_2} &= Q_{i_1} K_{j_1}^T - Q_{i_2} K_{j_2}^T \\ &= (Q_{i_1} - Q_{i_2} + Q_{i_2}) K_{j_1}^T - Q_{i_2} K_{j_2}^T \\ &= (Q_{i_1} - Q_{i_2}) K_{j_1}^T + Q_{i_2} (K_{j_1} - K_{j_2})^T \end{aligned}$$

Then by Hölder's inequality,

$$\mathcal{P}_{i_1, j_1} - \mathcal{P}_{i_2, j_2} \leq \|Q_{i_1} - Q_{i_2}\|_q \|K_{j_1}\|_p + \|Q_{i_2}\|_p \|K_{j_1} - K_{j_2}\|_q \leq 2\beta_1\beta_2$$

Therefore,  $\mathcal{P}_{i, j} \in [a, a + r]$  where  $r = 2\beta_1\beta_2$  and  $a = \min_{(i', j') \in \text{supp}(B_{x, y}^s)} \mathcal{P}_{i', j'}$ . Then, by Jensen's inequality,

$$\mu_{s, x, y} = \exp\left(\frac{1}{s^2} \sum_{(i, j) \in \text{supp}(B_{x, y}^s)} \mathcal{P}_{i, j}\right) \leq \frac{1}{s^2} \sum_{(i, j) \in \text{supp}(B_{x, y}^s)} \exp(\mathcal{P}_{i, j}) = \mu_{s, x, y}^*$$

Also, by Simic (2009), for a convex function  $f$  and  $x_i \in [a, b]$  for all  $x_i$ ,

$$\frac{1}{k} \sum_{i=1}^k f(x_i) - f\left(\frac{1}{k} \sum_{i=1}^k x_i\right) \leq f(a) + f(b) - 2f\left(\frac{a+b}{2}\right)$$

Therefore,

$$\mu_{s, x, y}^* - \mu_{s, x, y} \leq \exp(a) + \exp(a + r) - 2 \exp\left(a + \frac{r}{2}\right) \leq (1 + \exp(r) - 2 \exp\left(\frac{r}{2}\right)) \mu_{s, x, y} \quad (11)$$

Denote  $C_r = 1 + \exp(r) - 2 \exp\left(\frac{r}{2}\right)$ .  $\square$

**Proposition A.3.** Let  $R = \{b, 1\}$  for some  $b$  and  $\delta$  be the  $m_1$ -th largest  $\mu_{b,x,y}$ , assume for all  $(i, j) \in \text{supp}(B_{x,y}^b)$ ,  $\mathcal{P}_{i,j} \in [a, a + r]$  for some  $a$  and  $r > 0$ , then

$$\frac{\|\hat{\mathcal{A}} - \mathcal{A}\|_F}{\|\mathcal{A}\|_F} \leq \sqrt{\frac{(n^2 - m_1 b^2) C_{2r} \delta^2}{\sum_{i,j=1}^n \exp(2\mathcal{P}_{i,j})}}$$

where  $C_{2r} = 1 + \exp(2r) - 2 \exp(r)$

*Proof.* Similar to (11), for  $\mu'_{s,x,y}$  defined in (10)

$$\mu'_{s,x,y} - \mu_{s,x,y}^2 \leq \exp(2a) + \exp(2a + 2r) - 2 \exp(2a + r) \leq (1 + \exp(2r) - 2 \exp(r)) \mu_{s,x,y}^2$$

Denote  $C_{2r} = 1 + \exp(2r) - 2 \exp(r)$ , Then, the error by approximating a region  $\text{supp}(B_{x,y}^s)$  with lower resolution  $\mu_{s,x,y}$  is

$$\begin{aligned} \sum_{(i,j) \in \text{supp}(B_{x,y}^s)} (\mu_{s,x,y} - \exp(\mathcal{P}_{i,j}))^2 &= \sum_{(i,j) \in \text{supp}(B_{x,y}^s)} (\mu_{s,x,y}^2 + \exp(2\mathcal{P}_{i,j}) - 2\mu_{s,x,y} \exp(\mathcal{P}_{i,j})) \\ &= s^2 \mu_{s,x,y}^2 + s^2 \mu'_{s,x,y} - 2s^2 \mu_{s,x,y} \mu_{s,x,y}^* \\ &\leq s^2 \mu_{s,x,y}^2 + s^2 \mu'_{s,x,y} - 2s^2 \mu_{s,x,y}^2 \\ &\leq C_{2r} s^2 \mu_{s,x,y}^2 \end{aligned}$$

Since  $\mu_{1,x,y} = \exp(\mathcal{P}_{x,y})$ , the approximation error only comes from terms  $B_{x,y}^s \in \mathcal{J}$  for  $s > 1$ . Therefore,

$$\begin{aligned} \|\hat{\mathcal{A}} - \mathcal{A}\|_F^2 &= \sum_{B_{x,y}^b \in \mathcal{J}} \sum_{(i,j) \in \text{supp}(B_{x,y}^b)} (\mu_{b,x,y} - \exp(\mathcal{P}_{i,j}))^2 \\ &= \sum_{B_{x,y}^b \in \mathcal{J}} C_{2r} b^2 \mu_{b,x,y}^2 \\ &\leq \sum_{B_{x,y}^b \in \mathcal{J}} C_{2r} b^2 \delta^2 \\ &= (n^2 - m_1 b^2) C_{2r} \delta^2 \end{aligned}$$

Then, the relative error is

$$\frac{\|\hat{\mathcal{A}} - \mathcal{A}\|_F}{\|\mathcal{A}\|_F} \leq \sqrt{\frac{(n^2 - m_1 b^2) C_{2r} \delta^2}{\sum_{i,j=1}^n \exp(2\mathcal{P}_{i,j})}}$$

□

#### A.4. Experiments

**Implementations.** We use Reformer, Longformer, and Big Bird implementations from HuggingFace (Wolf et al., 2020). Nystromformer, SOFT, and Scatterbrain implementations are from their official Github repo. Linformer implementation is from Nystromformer's repo. Performer and H-Transformer-1D implementations are from Lucidrains repo.

**Efficiency.** We provide more details about the efficiency shown in the main text. The efficiency is measured on a single Nvidia RTX 3090. We use the largest possible batch size for each method and average the measurements over multiple steps and batch sizes to get an accurate measurement of runtime and memory consumption of a single instance. We show the efficiency and error for sequence length 256, 512, 1024, 2048, and 4096. The results are shown in Tab. 7

**FLOPS vs Runtime.** Existing automated profilers only support a limited set of PyTorch operators like Conv2D and BatchNorm (e.g., ptflops, thop, deepspeed, or PyTorch profiler cannot be used for all operators). So, runtime appeared a good indicator of efficiency.

**Hyperparameters.** In this section, we provide more details about the experiments. We run all experiments on a  $8 \times$  Nvidia RTX 3090 server. The hyperparameters of each experiments are summarized in Tab. 8.



Multi Resolution Analysis (MRA) for Approximate Self-Attention

Sequence Length Method	256			512			1024			2048			4096		
	Time	Memory	Error	Time	Memory	Error	Time	Memory	Error	Time	Memory	Error	Time	Memory	Error
Transformer	0.33	41.2	0.00	0.85	72.3	0.00	2.53	272.2	0.00	8.66	1085.8	0.00	30.67	4579.7	0.00
Nystromformer	0.31	11.4	0.70	0.59	22.4	0.76	1.16	45.3	0.83	2.29	95.1	0.88	4.50	209.8	0.91
	0.33	13.8	0.62	0.62	25.8	0.69	1.18	50.7	0.77	2.31	104.2	0.84	4.62	226.7	0.89
	0.41	21.6	0.55	0.71	35.4	0.62	1.31	64.0	0.71	2.52	125.1	0.78	4.96	262.6	0.85
	-	-	-	1.12	66.3	0.55	1.79	104.4	0.63	3.17	188.2	0.72	6.17	345.3	0.79
	-	-	-	-	-	-	3.70	238.6	0.55	5.50	375.3	0.65	8.92	627.8	0.73
	-	-	-	-	-	-	-	-	-	16.40	947.1	0.57	22.28	1573.1	0.66
	-	-	-	-	-	-	-	-	-	-	-	-	91.82	4018.8	0.58
Linformer	0.26	10.5	8.04	0.51	21.1	11.45	1.02	42.7	18.33	2.06	93.1	28.29	4.10	206.4	43.60
	0.26	11.5	5.42	0.51	23.0	8.27	1.02	47.2	12.45	2.07	99.3	19.97	4.15	218.9	31.21
	0.27	13.1	3.61	0.54	26.2	5.91	1.06	53.4	9.12	2.17	111.5	14.20	4.36	243.3	22.57
	0.30	16.9	2.56	0.60	33.3	4.02	1.20	67.2	6.26	2.40	138.9	9.85	4.82	297.8	15.91
	0.37	28.3	1.66	0.74	55.8	2.71	1.47	112.7	4.33	2.94	234.6	6.74	5.90	510.0	11.05
	-	-	-	1.02	100.8	1.86	2.03	204.0	3.03	4.07	427.1	4.80	8.12	938.8	7.62
	-	-	-	-	-	-	3.15	387.8	1.98	6.31	816.5	3.09	12.74	1814.4	5.18
	-	-	-	-	-	-	-	-	-	11.15	1613.4	2.21	22.74	3637.9	3.54
	-	-	-	-	-	-	-	-	-	-	-	-	43.32	7572.3	2.44
	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Performer	0.31	11.8	0.82	0.62	23.8	0.84	1.25	48.7	0.85	2.47	102.4	0.89	5.02	224.9	0.93
	0.33	13.1	0.81	0.67	26.2	0.83	1.33	53.4	0.85	2.66	111.6	0.88	5.33	243.4	0.92
	0.37	15.6	0.81	0.75	31.0	0.83	1.50	62.7	0.85	3.00	129.9	0.88	5.98	279.7	0.92
	0.51	20.7	0.80	1.03	40.6	0.83	1.92	81.2	0.84	3.70	332.9	0.88	7.40	704.4	0.92
	0.68	32.9	0.80	1.35	64.1	0.82	2.59	128.4	0.84	5.18	264.9	0.88	11.05	568.3	0.92
	-	-	-	2.02	115.7	0.82	4.05	232.0	0.84	8.12	479.9	0.88	16.45	1037.0	0.92
	-	-	-	-	-	-	7.01	439.0	0.85	14.10	909.9	0.88	28.66	1974.1	0.92
	-	-	-	-	-	-	-	-	-	26.17	1769.9	0.88	53.64	3848.7	0.92
	-	-	-	-	-	-	-	-	-	-	-	-	105.43	7597.5	0.93
Longformer	0.56	19.9	0.55	1.12	80.1	0.63	2.22	84.0	0.76	4.39	178.1	0.91	8.74	396.6	1.12
	0.57	20.8	0.40	1.14	42.1	0.48	2.29	86.6	0.62	4.54	183.2	0.77	9.01	407.2	0.97
	0.64	22.1	0.29	1.30	44.6	0.36	2.60	91.6	0.50	5.08	193.0	0.64	10.18	426.8	0.84
	-	-	-	1.62	49.8	0.27	3.26	101.6	0.40	6.47	212.8	0.54	12.91	466.1	0.73
	-	-	-	-	-	-	4.63	130.4	0.29	9.26	271.4	0.43	18.56	666.9	0.62
	-	-	-	-	-	-	-	-	-	14.69	456.8	0.33	29.98	1989.6	0.50
	-	-	-	-	-	-	-	-	-	-	-	-	52.06	1820.5	0.37
Big Bird	1.02	30.5	0.34	2.13	63.7	0.43	4.12	132.1	0.55	10.28	276.3	0.73	17.36	595.3	0.90
	1.05	30.5	0.18	2.21	66.5	0.27	4.49	140.2	0.41	9.89	295.4	0.57	19.69	720.0	0.76
	-	-	-	2.24	73.2	0.14	4.70	167.0	0.29	10.91	367.1	0.43	19.49	828.7	0.61
	-	-	-	-	-	-	5.53	203.7	0.17	12.59	466.7	0.32	24.57	1054.3	0.48
	-	-	-	-	-	-	-	-	-	16.93	666.9	0.18	34.53	1630.3	0.34
Scatterbrain	0.96	40.1	0.60	1.85	80.0	0.67	3.84	161.7	0.79	7.66	332.7	0.93	15.06	786.6	1.12
	0.95	40.1	0.45	1.91	81.4	0.52	3.81	167.0	0.65	7.66	353.6	0.79	15.42	788.2	0.97
	1.05	40.1	0.33	2.12	160.0	0.41	4.21	161.7	0.53	8.50	629.2	0.67	17.14	788.2	0.86
	-	-	-	2.53	160.0	0.31	5.07	167.0	0.44	10.15	629.2	0.57	20.52	788.2	0.77
	-	-	-	-	-	-	6.73	198.6	0.33	13.53	416.8	0.47	27.36	914.6	0.66
	-	-	-	-	-	-	-	-	-	20.11	585.4	0.36	42.60	1251.7	0.52
	-	-	-	-	-	-	-	-	-	-	-	-	67.81	2091.2	0.39
MRA-2	0.32	12.6	0.44	0.65	25.5	0.51	1.28	52.2	0.61	2.57	109.6	0.72	5.18	240.3	0.87
	0.33	13.0	0.34	0.66	26.2	0.40	1.34	53.8	0.50	2.68	112.9	0.63	5.41	246.9	0.78
	0.37	14.2	0.21	0.73	28.7	0.28	1.47	58.9	0.40	2.95	123.3	0.53	5.94	268.9	0.69
	-	-	-	0.88	34.9	0.15	1.75	71.3	0.28	3.50	148.1	0.41	7.23	318.5	0.58
	-	-	-	-	-	-	2.30	100.2	0.16	4.61	210.7	0.29	9.28	462.6	0.45
MRA-2-s	-	-	-	-	-	-	-	-	-	6.81	377.0	0.16	13.74	836.1	0.32
	-	-	-	-	-	-	-	-	-	-	-	-	22.68	1583.3	0.17
	0.28	10.9	0.54	0.57	22.1	0.63	1.13	45.5	0.77	2.26	96.1	0.93	4.55	212.5	1.14
	0.30	11.3	0.40	0.60	22.9	0.49	1.19	47.1	0.61	2.38	99.4	0.78	4.78	219.1	0.98
	0.33	12.1	0.23	0.66	24.5	0.33	1.32	50.2	0.47	2.64	105.6	0.63	5.27	231.5	0.83
	-	-	-	0.79	29.8	0.17	1.60	60.8	0.31	3.19	126.8	0.47	6.36	274.0	0.67
	-	-	-	-	-	-	2.15	99.8	0.17	4.31	209.9	0.32	8.61	460.7	0.51
-	-	-	-	-	-	-	-	-	6.50	376.2	0.17	13.08	834.2	0.35	
-	-	-	-	-	-	-	-	-	-	-	-	22.03	1581.2	0.19	

Table 7. Efficiency vs Approximation for approximation methods. The units for time and memory are ms and MB respectively. Error is the relative error used in the main text.

Experiment Task Sequence Length	RoBERTa-base				RoBERTa-small				ImageNet 1024
	MLM		MNLI	WikiHop	MLM		MNLI	WikiHop	
	512	4096	512	4096	512	4096	512	4096	
Num of Layers	12	12	12	12	4	4	4	4	4
Embedding Dim	768	768	768	768	128	128	128	128	128
Transformer Dim	768	768	768	768	384	384	384	384	128
Hidden Dim	3072	3072	3072	3072	1536	1536	1536	1536	512
Num of Heads	12	12	12	12	6	6	6	6	2
Head Dim	64	64	64	64	64	64	64	64	64
Batch Size	512	64	32	32	512	64	32	32	256
Learning Rate	5e-5	5e-5	3e-5	5e-5	1e-4	5e-5	3e-5	5e-5	5e-4
Num of Steps	20K	75K	4 epochs	15 epochs	150K	75K	10 epochs	25 epochs	300 epochs

Table 8. Hyperparameters for all experiments.

### A.5. Wavelets

Multiresolution analysis reorganizes a signal into different strata or resolutions, where the lower resolutions contain information summarizing global features of the signal, and higher ones capture fine-grained details of the signal. These strata are constructed iteratively, and we briefly describe the construction. For simplicity, we will focus on signals that are 1-dimensional (e.g., time series), but all of the following extends to signals of arbitrary dimension.

Construction of the filters  $L$  and  $H$ , which has been the subject of extensive research (Daubechies, 1992), guarantees that the wavelet analysis operator  $W : L_0 \rightarrow (L_N, H_N, H_{N-1}, \dots, H_1)$  is a linear isometry. That is,

$$\|L_0\|^2 = \|W(L_0)\|^2 = \|L_N\|^2 + \sum_{0 < k \leq N} \|H_k\|^2.$$

Thus, the reorganization of a signal under the action of  $W$  has a linear inverse,  $W^{-1} = W^*$ , which is the adjoint of  $W$ .

A wavelet decomposition allows one to alter values within particular resolution in an attempt to suppress information that we wish to ignore (Donoho, 1995). For example, if we wish to ignore fine-grained changes in the data, we would apply a threshold to the higher level, and replace small values in them with zero, and then reconstruct a new signal that has had the higher frequency information purged from it.

There is a tight connection between a signal's smoothness and the energy captured by the coefficients in  $H_k$  (Donoho & Johnstone, 1995). Informally, most of the energy of a smooth function is captured in the low frequency wavelet coefficients, and the energy captured by the high-frequency coefficients decreases as  $k$  decreases. Smoothness can be characterized by the rate of decrease. A reasonable heuristic is that smooth functions have sparse representations, and most energy lies within the lower strata of a wavelet decomposition. However, the smoothness of the wavelet basis also has a role here. The Haar wavelet system is not continuous, and one consequence of this is that energy of smooth functions leaks into higher frequency strata. The result is that wavelet decompositions based on the Haar system are typically not as sparse as decompositions based on more sophisticated wavelet systems.

Nevertheless, the Haar system is useful for its simplicity. Using the Haar system, we can bound the high frequency wavelet coefficients of  $\mathcal{P} = QK^T$  in terms of the corresponding wavelet coefficients of  $Q$  and  $K$ . The filters  $L$  and  $H$  of the Haar system are  $L := (2^{-1/2}, 2^{-1/2})$  and  $H := (2^{-1/2}, -2^{-1/2})$ . Column  $j$  of  $\mathcal{P}$  convolved with  $H$  has entry  $i$  equal to  $2^{-1/2}(\mathcal{P}_{i,j} - \mathcal{P}_{i+1,j})$ . If we denote the downsampling operation by down then entry  $i$  of  $H_1^{\mathcal{P},j} := \text{down}(\mathcal{P}_{\cdot,j} * H)$  is  $2^{-1/2}(\mathcal{P}_{2i,j} - \mathcal{P}_{2i+1,j})$ . Hence,

$$\begin{aligned} \left\| H_1^{\mathcal{P},j} \right\|_1 &= \sum_i \left| 2^{-1/2}(\mathcal{P}_{2i,j} - \mathcal{P}_{2i+1,j}) \right| = \sum_i \left| 2^{-1/2}(Q_{2i} - Q_{2i+1})K_j^T \right| \\ &\leq \sum_i \left\| 2^{-1/2}(Q_{2i} - Q_{2i+1}) \right\|_p \|K_j\|_q = \left( \sum_i \left\| (H_1^Q)_{i,\cdot} \right\|_p \right) \|K_j\|_q \end{aligned}$$

where  $(H_1^Q)_{i,\cdot} \in \mathbb{R}^d$  and  $1/p + 1/q = 1$ . Informally, the smoothness of the columns of  $\mathcal{P}$  is controlled by the smoothness of the columns of  $Q$ . A nearly identical statement can be made about the smoothness of the rows of  $\mathcal{P}$  and the smoothness of the rows of  $K^T$ . The same argument extends to the other strata of the wavelet decomposition. It is natural to compare this estimate to the bounds used in Lemma 4.1.