Adaptive Multi-Resolution Attention with Linear Complexity

Yao Zhang*, Yunpu Ma*, Thomas Seidl* and Volker Tresp†

* Institute of Informatics, LMU Munich, Germany
† Corporate Technology, Siemens AG, Germany
yzhang@dbs.ifi.lmu.de, volker.tresp@siemens.com

Abstract—Transformers have improved the state-of-the-art across numerous tasks in sequence modeling. Besides the quadratic computational and memory complexity with respect to the sequence length, the self-attention mechanism only processes information at the same scale, i.e., all attention heads are in the same resolution, resulting in the limited power of the Transformer. To remedy this, we propose a novel and efficient structure named Adaptive Multi-Resolution Attention (AdaMRA for short), which scales linearly to sequence length in terms of time and space. Specifically, we leverage a multi-resolution multi-head attention mechanism, enabling attention heads to capture long-range contextual information in a coarse-to-fine fashion. Moreover, to capture the potential relations between query representation and clues of different attention granularities, we leave the decision of which resolution of attention to use to query, which further improves the model’s capacity compared to the vanilla Transformer. In an effort to reduce complexity, we adopt kernel attention without degrading the performance. Extensive experiments demonstrate the effectiveness and efficiency of our model by achieving state-of-the-art speed-memory-accuracy trade-off. To facilitate AdaMRA utilization by the scientific community, the implementation will be made publicly available.

I. INTRODUCTION

The recent emergence of the Transformer has drastically reshaped the landscape of natural language processing research. Transformers have demonstrated superior performance in a wide variety of tasks, such as machine translation [1], text classification [2], question answering [3], automatic speech recognition [4], image generation [5], and image captioning [6]. The key innovation in Transformers is the introduction of a multi-head self-attention mechanism, which models the pairwise interactions between all positions in the sequence, regardless of their distance from each other. This operation has been shown quite effective.

Nonetheless, despite several notable successes of Transformers, computing the attention matrix, which is their key component, also turns out to be a major efficiency bottleneck due to its quadratic time and space complexity with respect to the sequence length. Therefore, the maximum sequence length is restricted by the amount of memory available. This inherent limitation of Transformers has prevented them from being successfully applied to domains requiring longer sequence lengths, like document classification. Furthermore, building large Transformer-based models in practice takes an enormous amount of time. Although fine-tuning pre-trained Transformers is relatively inexpensive, the memory issue still restricts the scenarios in which these models can be used. Besides the computational cost, qualitative analysis of attention heads [1] suggests that heads tend to favor flatter or more peaked distributions, depending on what phenomena they capture. Thus, using an extremely long sequence may limit the power of the model.

To this end, a wide spectrum of efficient, fast Transformers has been proposed to tackle these limitations. For instance, [7]–[12] addresses the problematic complexity by limiting the number of keys that each query attends to. However, these methods either break long-term dependency or hurt time efficiency. There is also a long line of research on using dense attention matrix but defined by low-rank kernels substituting softmax [13]–[16]. Although these approaches have achieved better speed-memory-accuracy trade-off, they still suffer from the aforementioned limitations of the self-attention mechanism. Another prominent line of work is to increase the memory capacity [17]–[19]. However, these works still suffer from problematic complexity issues.

Besides the computational cost, the self-attention mechanism of all models mentioned above only processes information at the same scale, i.e., all attention heads are in the same resolution. However, inspired by the fact that the
information in most domains has a hierarchical structure, for instance, word- and sentence-level information in the domain of text/language, low- and high-level features in the domain of image, etc., we suggest that processing information in a coarse-to-fine fashion could be beneficial for capturing hierarchically structured information. We thus propose **Adaptive Multi-Resolution Attention (AdaMRA)**, a linear time and space attention approach that captures long-distance dependencies in a coarse-to-fine manner. To be more precise, unlike vanilla Transformer, which maintains a constant resolution throughout all attention heads, AdaMRA employs multi-resolution attention heads that vary in the level of abstraction. Moreover, to capture the potential relations between query representation and clues of different attention granularities, each query is routed to the corresponding attention head. Furthermore, we adopt kernel attention [13] without sacrificing performance.

We evaluate the proposed method on the Long-Range-Arena (LRA) benchmark [20] and show that AdaMRA achieves promising performance while having a linear computational complexity with respect to the sequence length. Impressively, as shown in Figure 1, the average LRA scores increased by 4.32% and 3.66% from vanilla Transformer and the previous best-performing model BigBird, respectively. In terms of time and space efficiency, AdaMRA is around 10 times faster than vanilla Transformer, while 5 times smaller in GPU running memory occupation.

II. RELATED WORK

A conceptual way of reducing the complexity of full attention is to limit the number of accessible elements to attention. [7], [10], [21], [22] achieve this by using fixed, predefined patterns, e.g., block patterns of fixed stride. Another line of work is to consider which part of the inputs should be attended to by learning to assign tokens to buckets or clusters before performing attention. [12] uses locality sensitive hashing to group together token, [11], [23] employs online $k$-means to learn the space-partitioning centroids, and [9] sort keys in a block-wise fashion. However, they lack the flexibility to look at the full sequence and thus restrict the model capacity to capture long-distance dependencies. Moreover, additional computation steps required by some approaches (e.g., LSH in [12]) might undermine their final efficiency gains. Unlike these works, our method uniquely incorporates pooling-based compression to capture the context information of different scales with a small additional computation budget while maintaining excellent performance.

A. Kernel Attention

Another method is to improve efficiency by leveraging low-rank approximations of the softmax attention. Katharopoulos et al. [13] interpret the Softmax as a kernel and approximate the attention matrix via kernel approximation. Subsequently, this strategy is also employed by [14]–[16]. In [15], the approximation of standard softmax attention is based on adapting the Nyström method, while [14], [16] leverage random feature methods to approximate the softmax function. Although these approaches have achieved better speed-memory-accuracy trade-off, the performance of these methods is still affected by the quality of approximation and the fully-connected nature of self-attention in Transformer, which, as suggested by [24], is not a good inductive bias.

B. Increasing Memory Capacity

Memory is crucial for many tasks. However, extending the memory span is computationally expensive due to the attention mechanism’s quadratic time and space complexity. Several recent works have proposed strategies to increase the memory capacity of Transformers. BP-Transformer [18] is designed to incorporate the common-sense inductive bias of the hierarchical linguistic structure within the sentence, i.e., each query attends to context information from fine-grain to coarse-grain as the relative distance increase. [19] uses some pooling operator (e.g., max pooling) to reduce the number of memories in the past, where all memories are equally compressed regardless of the content of the current query. In [17], each attention head separately learns its temporal context size from data. The works mentioned above focus on increasing memory capacity without actually changing the memory resolution. Our work differs from theirs in that we focus on capturing long-term dependencies in a multi-resolution fashion, which, in turn, indirectly reduces the model’s memory footprint.

C. Multi-scale Attention Mechanism

Transformers have been widely applied to computer vision tasks such as image classification [25]. Recently, researchers have started to adopt the multiscalpe processing, also known as 'pyramid' strategies, into the attention mechanism. For instance, [26]–[28] feed multi-scale features into an attention module, whereas [29]–[31] combine feature contexts of multiple scales inside the attention module. In contrast, a noteworthy benefit of our design is queries can choose between different attention resolutions more flexibly.

III. MODEL

A. Revisiting Self-Attention and its Linearization

The self-attention function calculates, for every token, a weighted average of the feature representations of all other tokens with a weight proportional to a normalized similarity score between representations. Formally, let $X = \{x^{(1)},...,x^{(n)}\} \in \mathbb{R}^{n \times d}$ denotes an input sequence comprising $n$ tokens of dimension $d$. Given three matrices $Q, K$, and $V$, which are linear projections of the layer’s input $X$,

$$Q = X W^Q, \quad K = X W^K, \quad V = X W^V \quad \text{(1)}$$

where $Q, K, V \in \mathbb{R}^{n \times d}$ and $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$. Following common terminology, $Q, K$, and $V$ are referred to as the queries, keys, and values, respectively. The keys are used to compute a similarity score between each item and query. Then, weight the values of each item at each query context using the normalized similarity score. The attention outputs the
weighted sum of the values by the similarity score between the queries and keys. Thus, the generalized attention function for any similarity function can be written as:

$$Attention(Q_i, K, V) = \text{Score}(Q_i, K, V) = \frac{\sum_{j=1}^{n} \text{sim}(Q_i, K_j)V_j}{\sum_{j=1}^{n} \text{sim}(Q_i, K_j)}$$ (2)

According to [1], the unified similarity function can take the form of Softmax. Therefore, the quadratic complexity emerges from the computation of the similarity score between every pair of tokens.

In order to define an attention function, \(\text{sim}(\cdot)\) in Eq. 2 needs to be a non-negative function, which includes all kernels \(k(x, y) : \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}_+\) [13]. Given such a kernel with a feature representation \(\phi(x)\), we can rewrite the generalized attention (Eq. 2) as follows:

$$Attention(Q_i, K, V) = \frac{\sum_{j=1}^{n} \phi(Q_i)^T \phi(K_j)V_j}{\sum_{j=1}^{n} \phi(Q_i)^T \phi(K_j)}$$

$$= \frac{\phi(Q_i)^T \sum_{j=1}^{n} \phi(K_j)V_j}{\phi(Q_i)^T \sum_{j=1}^{n} \phi(K_j)}$$ (3)

\(\sum_{j=1}^{n} \phi(K_j)V_j^T\) and \(\sum_{j=1}^{n} \phi(K_j)\) could be reused for every query, therefore reducing the complexity from quadratic to linear both in terms of memory and computation.

Considering the fact that different parts of a sequence may be relevant in different ways, multi-head attention was introduced in Transformer. Assuming there are heads, this is simply the application of Eq. 2 in parallel heads, each with a different, learned linear transformation that allows specialization:

$$\text{MultiHead}(Q, K, V) = \text{Concat} (\text{Head}_1, ..., \text{Head}_H) W^O$$

$$\text{Head}_h(Q, K, V) = Attention(QW_h^Q, KW_h^K, VW_h^K)$$ (4)

where \(W_h^Q, W_h^K, W_h^K \in \mathbb{R}^{d \times d_h}, W_h^K \in \mathbb{R}^{d \times d_h}\) are the matrices that project the queries, keys, and values into the \(h\)-th subspace, respectively; \(W^O \in \mathbb{R}^{Hd_h \times d}\) is the matrix that computes a linear transformation of the heads, with, typically, \(Hd_h = Hd_k = d\).

B. Adaptive Multi-Resolution Attention

To capture hierarchically structured information effectively, we propose AdaMRA. The main idea is to employ the multi-resolution attention in a coarse-to-fine fashion and enable the query to choose between different resolutions of attention. This process is done independently for each layer, allowing queries in different layers to attend to contexts of different resolutions. We describe AdaMRA in the context of a single Transformer layer for brevity.

In AdaMRA, the input sequence \(X \in \mathbb{R}^{n \times d}\) still pass through three linear layers to form the queries \(Q \in \mathbb{R}^{n \times d}\), keys \(K \in \mathbb{R}^{n \times d}\), and values \(V \in \mathbb{R}^{n \times d}\), where \(n\) is sequence length and \(d\) is the embedding dimension. For each attention head \(h\), we define a compression rate \(c_h\), where a higher value indicates more fine-grained compressed information. To encode the context information, we produce compressed keys and values using certain compressive operations, which can be selected from \(k\)-means clustering [11], [23], projection [32], and convolution [19], etc. For the sake of computation efficiency, we employ segment means [15] to compress the original \((n \times d)\)-dimensional \(K\) and \(V\) into \((m_h \times d)\)-dimensional compressed \(\tilde{K}^h\) and \(\tilde{V}^h\), where \(h\) is the head and \(m_h = nc_h\) is the number of landmarks of head \(h\).

To be more precise, given the compression rate \(c_h\) for head \(h\), we separate the \(n\) keys/values into \(m_h\) segments. In our experiments, \(n\) is divisible by \(m\). If this is not the case in practice, we can pad inputs to a length divisible by \(m\). Note that to obtain multi-resolution attention, the compression rate of each head is different. Such a compression strategy can not only guarantee the preservation of important information but also simplify the model since \(c\) is usually a small number. More details regarding the impact of compression rate are provided in Section IV.
With compressed key $\hat{K}^h$ and value $\hat{V}^h$ available at hand, the different attention heads now have different resolutions. To capture the potential relations between query representation and clues of different attention granularities, we let query itself choose which resolution of attention to use, which is conditioned on the information encoded in query’s representation. This is accomplished by adding a Router [33] before the attention layer (see Figure 2). The router takes a token representation $q_i$ as an input and then routes this to the best-determined expert, i.e., attention head. Specifically, we adopt $F(\cdot)$, a parameterized function, for projecting query $q_i$ from $d$ dimensions to $H$ dimensions, where $H$ is the number of heads. We then normalize this value via a Softmax. Each query is routed to the head with the highest router probability $P$. In practice, we mask out the tokens that are not routed to the current head. Formally,

$$P = \text{Softmax}(F(Q)), \quad F(Q) = QW,$$

where $F(\cdot)$ is a parameterized function, $Q \in \mathbb{R}^{n \times d}$, the learnable parameter $W \in \mathbb{R}^{d \times H}$, and the router probability $P \in \mathbb{R}^{n \times H}$.

Finally, in pursuit of efficiency, we adopt kernel attention [13] while calculating attention using Eq. 3,

$$\text{Attention}(Q^h_i, \hat{K}^h, \hat{V}^h) = \frac{\sum_{j=1}^{n} \sin(Q^h_i, \hat{K}^h_j)\hat{V}^h_j}{\sum_{j=1}^{n} \sin(Q^h_i, \hat{K}^h_j)}\frac{\phi(\hat{K}^h_j)^T}{\phi(Q^h_i)^T} \frac{\sum_{j=1}^{n} \phi(\hat{K}^h_j)(\hat{V}^h_j)^T}{\phi(Q^h_i)^T} \frac{1}{\phi(\hat{K}^h_j)},$$

where $Q^h_i$ is $i$-th query that is routed to $h$-th head, $\hat{K}^h$ and $\hat{V}^h$ are the compressed query and value of $h$-th head. For our experiments, we employ ReLU as the feature function $\phi$ (see Section IV-C for feature function analysis).

As suggested by recent works on interpreting attention head roles, separate attention heads may learn to look for various relationships between tokens [34]. Thus, in practice, we use the same strategy to split the head into multiple subheads, whose resolution is the same as the original head, allowing the model to jointly attend to information at different positions from different representation subspaces. For our experiments, all attention heads have the same number of subheads. Multi-head AdaMRA is thus defined as:

$$\text{AdaMRA}(Q, K, V) = \sum_{h=1}^{H} \text{Head}_h W^O,$$

where $Q, K, V \in \mathbb{R}^{n \times d}$, $W^O \in \mathbb{R}^{d \times d}$ are learned matrices, $d_c = d/S$, $d_v = d/S$, $d_k = d$, $H$ is the number of heads, $S$ is the number of subheads, and $h$-$h$ denotes the $S$-th subhead of $h$-th head. The $s$-th subhead of $h$-th head is defined as:

$$\text{subhead}_{hs} = \text{Attention}(Q^K h^Q_i, \hat{K}^h, \hat{V}^h W^V h^V_i),$$

where $W^Q h^Q_i, W^K h^K_i, W^V h^V_i$ are the matrices that project the queries, keys and values into the $h$-th subspaces, respectively. For our experiments, we set $Sd_c = Sd_v = d$. Intuitively, one can think of $\phi(\hat{K}^h_j)^T\hat{V}^h$ as a global description/memory of the input sequence that the query will perform attention over. As discovered by previous works, global and multi-scale representations are useful. Therefore, to combine the low-level details and high-level semantics, each attention head has different memory scales, corresponding to a different semantic aspect of the entire input. For instance, coarse memory and fine-scale memory could correspond to the summary of paragraph and word representation, respectively. To further enhance feature expression ability, we leave the decision of which resolution of attention to use to query. Thus, a query can choose between the different resolutions of memory based on its own representation with more flexibility.

### C. Efficiency Advantage

We now show the efficiency advantage of AdaMRA in memory and computation. Assuming we have $H$ head with $m_h$ landmarks each, the landmark selection using segment means takes $O(n)$, where $n$ is the sequence length. The usage of kernel attention [13] eliminates the $O(n^2)$ terms from both the memory and computational complexities of the module. Instead, the computation of global description/memory ($\phi(\hat{K}^h_j)^T\hat{V}^h$) of dimensionality $d$ and new values take $O(m_h d^2)$ and $O(nd^2)$, respectively. Consequently, the total cost of AdaMRA scales as $O(Hn + \sum_{h=1}^{H} m_h d^2 + Hnd^2)$, i.e., scales linearly with respect to the sequence length $n$. In the next section, we show that a small $H$ ($H < n$) is enough to achieve good performance, which further increases the efficiency advantage of AdaMRA over vanilla Transformer.

### IV. EXPERIMENTS

In this section, we validate AdaMRA in terms of computational cost, memory consumption, and accuracy on LRA benchmark [20].

#### A. Experiment Settings

LRA is a suite of five general and challenging tasks designed to evaluate how well Transformers capture long-term dependencies from different modalities such as text, natural and synthetic images, and mathematical expressions requiring similarity, structural and visual-spatial reasoning.

- **Tasks**: Tasks used for comparison are as follows:
  1. **Long Listops**, designed to investigate the model’s capability of reasoning hierarchically structured data in a long-context scenario. We use a version of the ListOps dataset [35] of sequence lengths of up to 2K.
  2. **Byte-Level Text Classification** aims to test the models’ ability to deal with compositionality as it is required to compose characters into words into higher-level phrases. We use the IMDb reviews dataset [36] of a fixed max length of 4K.
  3. **Byte-Level Document Retrieval** uses the AAN dataset [37] to investigate a model’s ability to encode and store
### TABLE I

**EXPERIMENTAL RESULTS ON LRA BENCHMARK.** We report accuracy and relative training time/memory increase/decrease in comparison with the vanilla Transformer. The best model is in boldface, and the second-best is underlined. Results of Transformer, Reformer, Linformer, Performer, and Nyströmformer are extracted from [15]. AVG: average accuracy across all tasks. AdaMRA significantly outperforms other Transformer models across all tasks, with +4.32%, +3.66% in average accuracy against vanilla Transformer and the previous best-performing model BigBird, respectively. The most efficient model is Linear Transformer. Our model comes in a close second in terms of memory and computation.

| Model          | ListOps | Mem Time | Text | Mem Time | Retrieval | Mem Time | Image | Mem Time | Pathfinder | Mem Time | Avg  |
|----------------|---------|----------|------|----------|-----------|----------|-------|----------|------------|----------|------|
| Transformer [1]| 37.10   | 1×       | 65.02| 1×       | 79.35     | 1×       | 38.20 | 1×       | 74.16      | 1×       | 58.77|
| BigBird [10]   | 38.55   | 1.4×     | 63.90| 0.9×     | 81.50     | 1.8×     | 38.30 | 0.9×     | 74.89      | 0.9×     | 59.43|
| Reformer [12]  | 19.05   | 1.9×     | 64.88| 1.2×     | 78.64     | 2.8×     | 43.29 | 1.2×     | 69.36      | 1.2×     | 55.04|
| Linformer [32] | 37.25   | 1.9×     | 55.91| 1.2×     | 79.37     | 2.9×     | 37.84 | 1.2×     | 67.60      | 1.2×     | 55.59|
| Linear Trans. [13]| 37.35  | 2.8×     | 64.15| 1.5×     | 81.10     | 5.8×     | 38.20 | 1.5×     | 70.20      | 1.5×     | 58.20|
| Performer [14] | 18.80   | 2.7×     | 63.81| 1.5×     | 78.62     | 5.1×     | 37.07 | 1.5×     | 69.87      | 1.5×     | 53.63|
| Nyströmformer [15]| 37.15 | 1.1×     | 65.52| 0.7×     | 79.56     | 1.8×     | 41.58 | 0.7×     | 70.94      | 0.7×     | 58.95|
| Ours           | **40.40**| **2.3×** | **68.44**| **1.4×** | **84.83**| **4.5×** | **46.00**| **1.4×** | **75.77** | **1.4×** | **63.09**|

compressed representation. The model learns a similarity score between two documents. Each document has a sequence length of 4K. (4) **Image Classification** This task serves as a test of how well models are able to capture the 2D spatial relations between input pixels. We use the CIFAR-10 dataset [38] for this task. (5) **Pathfinder** In this task, we are interested in the model’s ability to capture long-range spatial dependencies. The model makes a binary decision on whether two points are connected by a path [39].

**b) Baselines and Implementation Details:** We base our evaluation on six recently proposed efficient Transformer models. Aside from the vanilla Transformer [1], we compare our model against other efficient self-attention variants, including Reformer [12], Linear Transformer [13], Performer [14], Linformer [32], Big Bird [10] and Nyströmformer [15]. The embedding dimension is 64, and the hidden dimension is 128. Our data split, preprocessing, and training procedure follow those of [15]. For all models, we use the default PyTorch implementation. We benchmark all models’ speed and memory on a Tesla T4 GPU with 16GB of memory.

### B. Performance Comparison

**a) Accuracy Comparison:** Table I reports the LRA score and efficiency improvements in comparison with the vanilla Transformer of several Transformer models. The batch size is 16.

Our model brings consistently considerable performance boosts over the baseline models to all tasks. Specifically, our model outperforms other efficient self-attention methods, with 4.32%, 4.89%, and 9.46% in average accuracy against vanilla Transformer, Linear Transformer, and Performer, respectively. Besides, our model also achieves significant performance gain compared to the previous best-performing model BigBird. This might be attributed to the fact that, in contrast to BigBird, our attention layer is able to exchange information globally on the entire sequence.

It is worth mentioning that our model boosts the score by 3.42% and 5.48% on the tasks Text (n=4K) and Retrieval (n=4K) compared with vanilla Transformer, suggesting that our model is advantageous in tasks that require large sequence length. In addition, AdaMRA outperforms vanilla Transformer by 3.3% on ListOps task, implying that AdaMRA is better at handling hierarchical data. More importantly, we notice the performance gains, especially on the Image Classification task, outperforming vanilla Transformer by 7.8%, which indicates that the inductive bias of AdaMRA plays a substantial role in this task. Thus, our model has the capability to capture 2D spatial relations.

**b) Speed and Memory Comparison:** To better illustrate the boosted efficiency, we compare Transformers with respect to their computational and memory requirements. We use the IMDb dataset with a batch size of 16 for all runs. Table II shows peak allocated GPU memory and required time of the sequence lengths \{1K, 2K, 4K\}. All compared models are of the same size as those described above.

The overall fastest models are kernel-based models (Performer and Linear Transformer). The model with the smallest memory footprint is the Linear Transformer, coming in at 1741 MB compared to 10327 MB for the vanilla Transformer at 4K. Our method comes in a close second and is almost
C. Ablation Study

As fast as the fastest one. Similar to speed, our model is also relatively compact and is almost as compact as Linear Transformer and Performer. Importantly, our model speeds up over the vanilla Transformer by about 10.4× on 4K sequence length and requires only about 20% of the memory of the vanilla Transformer at 4K. As sequence length increases, the training time speed-up and memory savings are even more dramatic. We also notice that the time and memory consumption of AdaMRA with different configurations are approximately equal.

Notably, kernel-based models are fast and compact at the cost of relatively lower quantitative performance (see Table I). In contrast, our model is competitive in both accuracy and efficiency, as Figure 1 shows. Besides, our analysis indicates that AdaMRA efficiency gains are especially notable on long sequences, suggesting that AdaMRA will be particularly useful in tasks that require large sequence length, fast training speed, and low memory footprints.

b) Architecture Design: To understand the importance of each component, we conduct ablation experiments for the AdaMRA architecture. In Table V, Rand means randomly assigning each query to attention heads; Softmax means adding a multi-resolution approach into vanilla attention mechanism, i.e., using Softmax as $\text{sim}$ in Eq. 7; ELU+1 means employing $\text{elu} + 1$ as feature function $\phi$ in Eq. 7; ReLU means using

\begin{equation}
\text{SMAT} = S_{\text{norm}} + (1 - M_{\text{norm}}) + A_{\text{norm}}
\end{equation}

where $S_{\text{norm}}, M_{\text{norm}}, A_{\text{norm}}$ are normalized speed (examples per sec), peak memory usage as well as LRA score after applying the MinMaxScaler. In this experiment, we use the IMDb dataset of the sequence length 4K with a batch size of 8 for all runs. As shown in Table III, AdaMRA consistently outperforms other variants by a large margin in terms of SMAT score.

C. Ablation Study

TABLE II

Comparison of training time and peak memory consumption on various input sequence lengths. We report relative training time & memory increase/decrease in comparison with the vanilla Transformer in brackets. The best model is in boldface, and the second-best is underlined. Performer-32 denotes Performer self-attention module using a feature map of 32 dimensions. Nyströmformer-32 denotes Nyströmformer self-attention module using 32 landmarks. Ours-(c,1) denotes AdaMRA self-attention module with four attention heads using a compression rate of c, on each head. AdaMRA offers favorable memory and time efficiency over standard self-attention and is almost as fast and compact as kernel-based Transformer (Linear Transformer and Performer).

| Model                  | Running time (ms) | Peak Memory Usage (MB) |
|------------------------|-------------------|------------------------|
|                        | 1K                | 2K                     | 4K                     | 1K                | 2K                     | 4K                     |
| Transformer [1]        | 82(1×)            | 272(1×)                | 1007(1×)               | 1713(1×)          | 3829(1×)               | 10327(1×)              |
| BigBird [10]           | 104(0.79×)        | 211(1.3×)              | 420(2.4×)              | 1815(0.9×)        | 2835(1.4×)              | 4921(2.1×)             |
| Reformer [12]          | 53(1.5×)          | 103(2.6×)              | 200(5.0×)              | 1467(1.2×)        | 2007(1.9×)              | 3125(3.3×)             |
| Linformer [32]         | 37(2.2×)          | 71(3.8×)               | 139(7.2×)              | 1499(1.1×)        | 1978(1.9×)              | 3035(3.4×)             |
| Linear Transformer [13]| 22(3.7×)          | 38(7.1×)               | 69(14.6×)              | 1113(1.5×)        | 1353(2.8×)              | 1741(5.9×)             |
| Performer-32 [14]     | 30(2.7×)          | 54(5.0×)               | 100(10.7×)             | 1171(1.5×)        | 1439(2.7×)              | 1917(5.4×)             |
| Nyströmformer-32 [15] | 76(1.1×)          | 160(1.7×)              | 233(4.3×)              | 2627(0.6×)        | 3381(1.1×)              | 4685(2.2×)             |
| Ours-(y,4,y,16,y,6)   | 33(2.4×)          | 54(5.0×)               | 98(10.3×)              | 1239(1.4×)        | 1581(2.4×)              | 2201(4.7×)             |
| Ours-(y,6,y,16,y,6)   | 32(2.6×)          | 55(5.1×)               | 96(10.5×)              | 1201(1.4×)        | 1565(2.4×)              | 2177(4.9×)             |

a) Compression Rate: To understand the impact of compression rate $c$, we conduct ablation experiments on Byte-Level Text Classification and Image Classification. We experiment with various numbers of attention heads and vary the compression rate $c_h$ of each head. We use the same $c_h$ across all layers. As indicated by the results in Table IV, the choice of compression rate is crucial for the final performance. However, compared to the vanilla Transformer, all configurations achieve consistent improvement on both tasks (see Table I).

Besides, there are a few things to notice: i) Model 5 outperforms Model 4&6, and Model 3 outperforms Model 2, indicating a benefit in using a moderate compression rate, and using an extremely low compression rate cause a significant performance drop. We speculate that using an extremely low compression rate might lose too much information. ii) Model 3 outperforms Model 4-8, which means AdaMRA does not perform better as the number of heads $H$ increases. This indicates that having multiple attention heads is effective, but a too large number of heads hurts. iii) When we use a relatively low compression rate, the resulting model’s (Model 3&5) performance already outperforms all other Transformer models. This suggests that we can decrease the compression rate to a certain extent, which further increases the efficiency advantage of AdaMRA over the vanilla Transformer. iv) One can notice a significant accuracy drop when using the single-resolution (Model 9-11), which indicates that a multi-resolution attention head is beneficial for capturing hierarchically structured information.
TABLE III
COMPARISON OF SMAT score (higher is better). The value normalized with MinMaxScaler is in brackets. The best model is in boldface, and the second-best is underlined. AdaMRA significantly outperforms other Transformer models.

| Model              | Speed (0.00) | Peak Memory Usage (MB) | LRA Score | SMAT Score |
|--------------------|--------------|------------------------|-----------|------------|
| Transformer [1]    | 14.5         | 6645 (1.00)            | 58.77 (0.54) | 0.54       |
| BigBird [10]       | 36.4 (0.12)  | 2917 (0.30)            | 59.43 (0.61) | 1.43       |
| Reformer [12] [13] | 80.0 (0.35)  | 2023 (0.13)            | 55.04 (0.15) | 1.37       |
| Linformer [32]     | 111.1 (0.52) | 2003 (0.12)            | 55.90 (0.21) | 1.61       |
| Linear Transformer | 200.0 (1.00) | 1353 (0.00)            | 58.20 (0.48) | 2.48       |
| Performer-32 [14] | 142.9 (0.69) | 1439 (0.02)            | 53.63 (0.00) | 1.67       |
| Nyströmformer-32 [15] | 57.1 (0.23) | 2687 (0.25)            | 58.95 (0.56) | 1.54       |
| Ours               | 160.0 (0.78) | 1475 (0.02)            | 63.09 (1.00) | 2.76       |

TABLE IV
ABLATION STUDY OF AdaMRA ON THE TEXT CLASSIFICATION AND IMAGE CLASSIFICATION.

| ID | Compression Rate | Acc (Img) | Acc (Txt) |
|----|------------------|-----------|-----------|
| 1  | (V, U2)          | 45.3      | 66.8      |
| 2  | (V, V16, U2)     | 45.5      | 66.3      |
| 3  | (V, U16)         | 46.0      | 68.4      |
| 4  | (V, V, V16)      | 44.9      | 67.8      |
| 5  | (V, V, V16, U2)  | 45.2      | 68.2      |
| 6  | (V, V16, U2, V4) | 44.0      | 67.3      |
| 7  | (V1, V, V16, U2, V4) | 44.9 | 67.6 |
| 8  | (V2, V, V16, V12, U28) | 45.0 | 65.7 |
| 9  | (1, 1, 1)        | 36.31     | 63.82     |
| 10 | (V2, V, V)       | 38.98     | 64.11     |
| 11 | (V, V, V)        | 42.38     | 63.38     |

TABLE V
ABLATION STUDY FOR ARCHITECTURE

| Model  | Img  | Txt  |
|--------|------|------|
| Transformer | 38.20 | 65.02 |
| Rand   | 41.40 | 64.64 |
| Softmax | 41.19 | 65.56 |
| ELU+1  | 43.73 | 66.33 |
| ReLU   | 46.00 | 68.44 |

ReLU as ϕ (AdaMRA). As indicated by the results, the privilege of AdaMRA comes from the learned routing and multi-resolution attention simultaneously. Using ReLU as the feature function is also advantageous, without which we only have small gains over the vanilla model. The multi-resolution method is also compatible with other attention mechanisms (e.g., vanilla attention) to a certain extent, and we leave it for future work.

V. CONCLUSION

Transformers are notoriously slow to train and deploy because of their quadratic time and space complexity. In this paper, we propose a novel and efficient structure AdaMRA. We see the benefits of this approach in the domain of text, images, mathematical expressions, etc. In particular, we have shown that our model achieves a state-of-the-art speed-memory-accuracy trade-off. The main limitation of this work is the additional hyperparameters (number of Heads H and the compression rate of each head c_k). However, we empirically show that multiple configurations work fairly well. More importantly, the scalability of AdaMRA enables application in tasks that require working with large inputs, fast training speed, or low memory footprints.

ACKNOWLEDGEMENT

This work has been funded by the German Federal Ministry of Education and Research and the Bavarian State Ministry for Science and the Arts. The authors of this work take full responsibility for its content.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547de6e91fbd05c1c4a845aa-Paper.pdf

[2] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers, I. Gurevych and Y. Miyao, Eds. Association for Computational Linguistics, 2018, pp. 328–339. [Online]. Available: https://aclanthology.org/P18-1031/1

[3] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “SQuAD: 100,000+ questions for machine comprehension of text,” in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. [Online]. Available: https://aclanthology.org/P16-1264

[4] L. Dong, S. Xu, and B. Xu, “Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition,” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 5884–5888.

[5] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, “Image transformer,” in International Conference on Machine Learning. PMLR, 2018, pp. 4055–4064.

[6] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in International conference on machine learning. PMLR, 2015, pp. 2048–2057.

[7] I. Beltagy, M. E. Peters, and A. Cohan, “Longformer: The long-document transformer,” arXiv preprint arXiv:2004.05150, 2020.

[8] C. Wang, Z. Ye, A. Zhang, Z. Zhang, and A. J. Smola, “Transformer on a diet,” arXiv preprint arXiv:2002.06170, 2020.

[9] Y. Tay, D. Bahri, L. Yang, D. Metzler, and D.-C. Juan, “Sparse sinkhorn attention,” in International Conference on Machine Learning. PMLR, 2020, pp. 9438–9447.
