Moss: Adaptive Inductive Bias Attention for Efficient and Robust Sequence Processing

Lingkai Hu, Feng Zhan, Wenkai Huang, *Member, IEEE* and Weiming Gan

***Abstract*—Sequence processing is a fundamental research area in artificial intelligence (AI) that encompasses various tasks and applications. Existing models—such as recurrent neural networks (RNNs) and Transformers—have drawbacks such as slow computation, high complexity, and overfitting. In this paper, we propose Moss, a novel sequence processing model that leverages the adaptive inductive bias (AIB)-attention mechanism. Moss can capture positional inductive relations more robustly and efficiently than Transformers. Moreover, it enables parallel computation and can handle extremely long sequences with near-linear complexity. We conduct extensive experiments on multiple data sets and tasks and evaluate Moss using various metrics. The results demonstrate that Moss achieves state-of-the-art (SOTA) performance on several language modeling benchmarks and surpasses other models in terms of accuracy, speed, and memory usage.**

***Index Terms*—Attention mechanism, complexity, inductive bias, language modelling, sequence processing**

I. Introduction

SEQUENCE processing is a fundamental research area in artificial intelligence (AI) that encompasses tasks such as sequence modeling [1]–[5], classification [6]–[8], and translation [1], [9]–[12]. These tasks have a wide range of applications in natural language processing (NLP), speech recognition [13], music synthesis [14], chatbots [15], machine translation, and more. Numerous influential models have been proposed to address these tasks, such as recurrent neural networks (RNNs) [16] and Transformers [12].

However, these models also have inherent drawbacks. RNNs operate sequentially along the sequence order, thereby resulting in slow computation and memory loss for long sequences [16], [17]. Transformers employ a self-attention mechanism that can handle long sequences, but its time and space complexity are quadratic functions of the input sequence length, thereby requiring a substantial amount of computational resources. These limitations hinder their performance on long sequence tasks [18]–[20]. Although numerous attempts have been made to overcome these limitations [21]–[32], the effect is marginal and the challenge of long sequences remains.

Attention mechanisms are essential for numerous sequence processing models [33]–[35], such as Transformers. However, unlike models that incorporate structural priors in the input, attention mechanisms have to learn the structural relations from the data. This makes Transformer-like models prone to overfitting on small to medium-sized data sets [18], [20], [36], [37].

In this study, we propose Moss, a novel sequence processing model based on the adaptive inductive bias (AIB)-attention mechanism. Moss can effectively capture positional inductive relations with higher robustness and lower time and space complexity. In addition, It enables parallel computation and can build long-distance dependency relations with a complexity of $O(L\log_2 L)$.

The following are the main contributions of this paper:

- We introduce the AIB-attention mechanism, a novel technique that leverages positional inductive bias to enhance the learning ability of attention-based models.
- We present Moss, a new sequence processing model that incorporates the AIB-attention mechanism into its architecture. We analyze its advantages over existing models in terms of complexity, robustness, and generalization.
- We conduct extensive experiments on multiple data sets and tasks and evaluate Moss using multiple metrics, such as accuracy, calculation speed, and peak memory usage. In the character-level language modeling task, Moss easily achieved better results than GPTs of the same size in data set benchmarks, including text8 [38] and enwik8 [39]. In the enwik8 character-level language modeling task, this model exceeds the current SOTA. The results reveal that Moss outperforms other models on these metrics and can effecitvely handle long sequences.

II. Background

A. Attention mechanism

Attention mechanism is a special structure that can be embedded into machine learning models to automatically learn and compute the importance of input data for output data [40], [41]. It draws inspiration from human visual attention, which selectively focuses on parts of images or texts [42], [43]. Mathematically, the attention mechanism can be considered a function that maps a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. Moreover, the attention mechanism can enhance the efficiency and performance of models in various fields, such as NLP and image processing [12], [44]–[46].

Encoder-decoder is a common framework that incorporates the attention mechanism [10], [47], [48]. It can transform one sequence (such as an audio signal) into another sequence

Lingkai Hu, Feng Zhan, Wenkai Huang and Weiming Gan are with the School of Mechanical and Electrical Engineering, Guangzhou University, 510006 Guangzhou, China. E-mail: 2112107060@e.gzhu.edu.cn, 2007300008@e.gzhu.edu.cn, smallkat@gzhu.edu.cn, 2112107050@e.gzhu.edu.cn.

(Corresponding author: Wenkai Huang.)

TABLE I
SEVERAL PERFORMANCE INDICATORS OF DIFFERENT TYPES OF NEURAL LAYERS.

Layer Type	Complexity per Layer	Maximum Path Length	Receptive Field
Self-Attention	$O(L^2D)$	$O(1)$	$O(L)$
AIB-Attention	$O(LD\log_2 L)$	$O(1)$	$O(L)$
Fully Connected	$O(L^2D^2)$	$O(1)$	$O(L)$
Convolutional	$O(KLD^2)$	$O(\log_K L)$	$O(K)$
Recurrent	$O(LD^2)$	$O(L)$	$O(L)$

(such as a text transcript). The encoder encodes the input sequence into a semantic vector, and the decoder generates the output sequence based on this vector. The attention mechanism enables the decoder to dynamically adjust the attention weight to the encoder output, thereby improving the accuracy and fluency of speech recognition.

The self-attention mechanism is a classic implementation of the attention mechanism. It enables each element in the input sequence to interact with other elements, thereby capturing the long-distance dependency relationship within the sequence. The self-attention mechanism can also reduce the computational complexity and memory consumption compared to other attention mechanisms. The core component of the Transformer model is the self-attention mechanism. In the field of NLP, the Transformer model has achieved numerous breakthrough results in tasks such as machine translation [49], text summarization [50], question answering [51], and natural language generation [52].

B. Transformers

Transformers are a popular neural network architecture that aim to solve sequence-to-sequence tasks while handling long-range dependencies with ease and are now a state-of-the-art technique in the field of NLP. A Transformer neural network can take an input sentence in the form of a sequence of vectors and convert it into a vector called an encoding and then decode it back into another sequence. An important part of the Transformer model is the attention mechanism, which allows the processing of one input word to include relevant data from certain other words, while masking the words that do not convey relevant information. Transformer neural networks have achieved remarkable results in various NLP tasks.

In the original Transformer model, the multihead attention mechanism is calculated in the following manner:

$$Q_i = XW_i^Q \tag{1}$$

$$K_i = CW_i^K \tag{2}$$

$$V_i = CW_i^V \tag{3}$$

$$head_i = Softmax\left(\frac{Q_i {K_i}^T}{\sqrt{d_k}}\right) V_i \tag{4}$$

$$Y = Attention\left(X, C\right) = Concat\left(head_1, ..., head_h\right). \tag{5}$$

It contains two inputs: the query sequence $X \in \mathbb{R}^{L\times D}$ with length L and the context sequence $C \in \mathbb{R}^{m\times D}$ with length m. The output sequence $head_i \in \mathbb{R}^{L\times d_k}$ with length L, which is the same as X. Here, D represents the encoding dimension, d_k represents the dimension of the key, h represents the number of heads, and generally $d_k = D/h$. Moreover, W_Q, W_K, $W_V \in \mathbb{R}^{D\times d_k}$ are three learnable parameters. The output $Y \in \mathbb{R}^{L\times D}$ of the attention mechanism is the concatenation of multiple heads in the last dimension. When calculating self-attention, $C = X$; when calculating cross-attention, C and X have different sources. Generally, C is the output of the Transformer encoder and X is the input of the decoder.

In the attention mechanism, each query vector needs to perform multiplication with all keys in the sequence, while each query vector in the attention matrix $Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \in \mathbb{R}^{L\times m}$ also needs to perform multiplication with all values. This results in a time and space complexity of $O\left(LmD\right)$ for Transformers. In AIB, the complexity is reduced to $O\left(LDlog_2 L\right)$, which significantly improves the computational efficiency of the neural network.

Table I summarizes the complexity, maximum path length, and receptive field of various commonly used neural network layers. From this, the performance indicators of different types of layers can be clearly compared.

C. Inductive Bias

Inductive bias refers to the necessary assumptions that a machine learning algorithm makes about the target function of a learning problem [53], [54]. In machine learning, we want to construct algorithms that can predict the output based on the given input. However, for inputs that we have not encountered before, we cannot determine the output value without any additional assumptions. Therefore, we need to make a few assumptions about the properties of the target function, and these assumptions constitute the inductive bias. Inductive bias originates from the no free lunch theorem, which states that no learning algorithm can perform well on all possible problems without any prior knowledge [55].

Inductive bias can help deep learning algorithms converge faster and generalize better, while reducing computational and parameter costs. Inductive biases can be categorized into two different groups—explicit and implicit. The former represents the assumptions that are explicitly encoded in the model architecture or the learning algorithm, while the latter represents the assumptions that are implicitly induced by factors such as initialization, optimization, and regularization [54], [56].

In deep learning, many neural network structures are designed based on explicit inductive biases. For example, convolutional neural networks (CNNs) assume local connectivity and translation invariance, recurrent neural networks (RNNs) assume temporal dependency and sequential order, and graph neural networks (GNNs) assume relational structure and permutation invariance. These structures mainly model local structural relationships, while neglecting the modeling of global relational patterns. On the other hand, neural network structures with less explicit inductive biases, such as fully connected and Transformer layers, can capture more contextual information but may have higher complexity and be more prone to overfitting [55].

In this paper, we introduce the main idea and structure of the Moss model, which is a novel sequence processing model that combines explicit and implicit inductive biases to achieve superior robustness and fitting ability. We compare the Moss model with the Transformer model theoretically and empirically, and show that the Moss model has a lower complexity and faster computation speed than that of the Transformer model, while also being able to perform parallel computation. We also apply the Moss model to several NLP tasks, and demonstrate its effectiveness and superiority over the Transformer model. We hope that our study can inspire more research on improving attention mechanism and sequence processing models.

III. Moss

In this section, we introduce our proposed Moss model. The core structure of Moss is AIB-Attention. AIB-Attention applies adaptive information induction to attention computation, thereby allowing the model to dynamically establish inductive bias relationships in the data. A notable feature of AIB-Attention is its use of a weighted attention mechanism to fuse contextual information. We demonstrate that by applying this structure, we can achieve lower time and space complexity while maintaining excellent robustness and fitting ability.

This section initiates with the introduction and analysis of an attention mechanism called prefix sum (PS)-attention. Subsequently, we present its improved version, AIB-attention, and conduct a mathematical analysis to clarify its significance. Finally, we outline the specific structure of Moss, incorporating AIB-attention.

A. *PS-Attention*

PS-Attention can be regarded as the prototype of AIB-Attention, encompassing two computational steps: 1) Calculate the attention matrix and values matrix through linear transformation. 2) Combine the attention matrix and the value matrix using a special weighted prefix sum algorithm to produce the output.

This paragraph introduces the mechanism of the PS-attention computation. Given that a sequence $X \in \mathbb{R}^{L\times D}$ is the input to PS-attention. Then, we compute the attention matrix $A \in \mathbb{R}^{L\times D}$ and the value matrix $V \in \mathbb{R}^{L\times D}$ in the following manner:

$$A = XW^A + b_1 \tag{6}$$
$$V = XW^V + b_2, \tag{7}$$

where $W^A \in \mathbb{R}^{D\times D}$, $W^V \in \mathbb{R}^{D\times D}$, $b_1 \in \mathbb{R}^D$ and $b_2 \in \mathbb{R}^D$ are learnable parameters. Then, we calculate the attention output for each time step based on A and V. In the decoder, the i-th output can only access information from the first i inputs. Therefore,

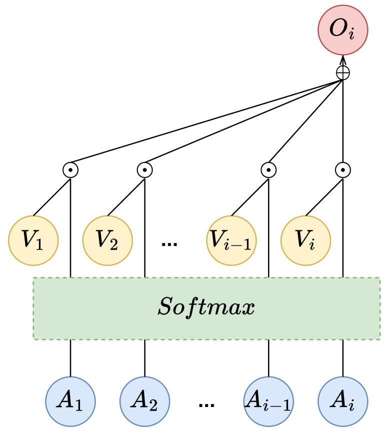

Fig. 1. The computational process of equation (8) in PS-attention.

$$O_i = Sum(Softmax\,(A\,[:i]) \odot V\,[:i]) \tag{8}$$
$$= \sum_{j=1}^{i}\left(\frac{Exp\,(A_j)}{\sum_{k=1}^{j} A_k} \odot V_j\right) \tag{9}$$
$$O = [O_1, ..., O_L] \tag{10}$$
$$Y = OW^O + b_3. \tag{11}$$

In this paper, the symbol '$\odot$' represents the element-wise product between vectors (or matrices). Where $W^O \in \mathbb{R}^{D\times D}$ and $b_3 \in \mathbb{R}^D$ are learnable parameters, $O_i \in \mathbb{R}^D$ denotes the i-th vector of the output matrix, $O \in \mathbb{R}^{L\times D}$, $A\,[:i] \in \mathbb{R}^{i\times D}$ denotes the matrix formed by the first i vectors of A, and $V\,[:i] \in \mathbb{R}^{i\times D}$ denotes the matrix formed by the first i vectors of V. $A_j \in \mathbb{R}^D$ and $A_k \in \mathbb{R}^D$ denote the j-th and k-th vectors of A, respectively. $V_j \in \mathbb{R}^D$ denotes the j-th vector of V. The computational process in (8), as illustrated in Fig. 1, reveals that the essence of A is an assessment of the significance of each time step, and based on this assessment, a weighted sum is performed on the values V at each time step to obtain the corresponding output value O_i.

In the above computation, the softmax operation is performed at each time step in (9), which would require too much computation. Therefore, the next step is to optimize it. Firstly, the calculation of (8) - (10) can be expressed as,

$$E = Exp\,(A) \tag{12}$$
$$H_i = \sum_{j=1}^{i} (E_j \odot V_j) \tag{13}$$
$$U_i = \sum_{j=1}^{i} E_j \tag{14}$$
$$H = [H_1, ..., H_L] \tag{15}$$
$$U = [U_1, ..., U_L] \tag{16}$$
$$O = \frac{H}{U}. \tag{17}$$

In this computation, $H \in \mathbb{R}^{L\times D}$ can be expressed as the prefix sum of $E\odot V$ and $U \in \mathbb{R}^{L\times D}$ as the prefix sum of $E \in \mathbb{R}^{L\times D}$.

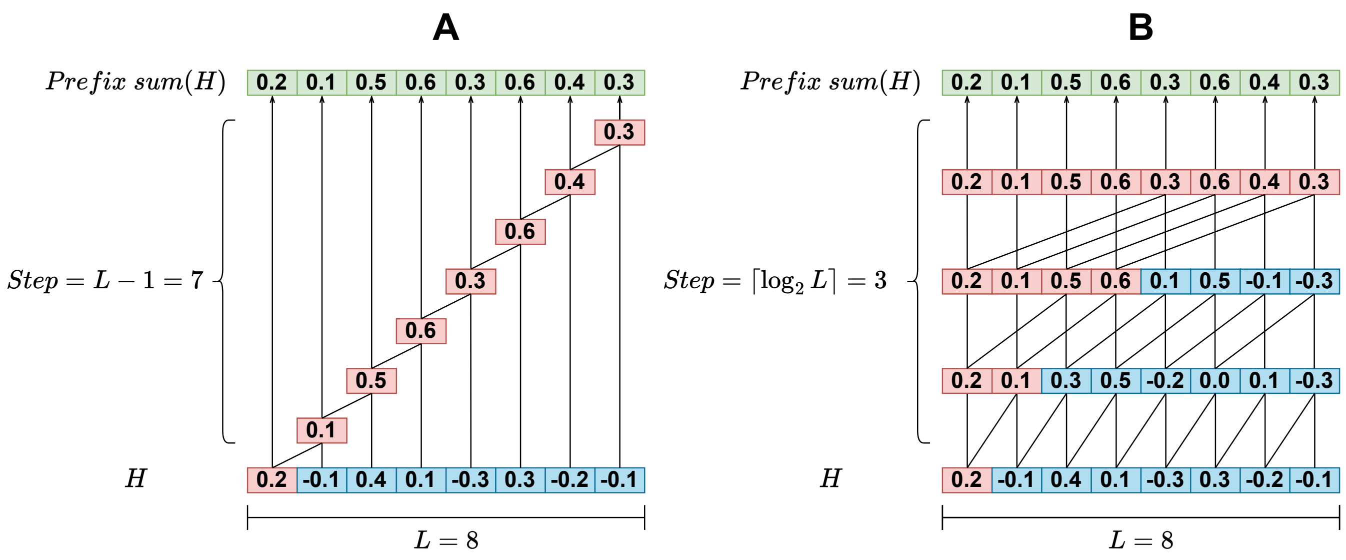

Fig. 2. Comparison of the computation process for the calculation of a sequence H of length 8 in two distinct prefix sum algorithms. (A) A commonly employed prefix sum algorithm A with a total of 7 steps. (B) The prefix sum algorithm B employed in this study, with a total of 3 steps.

H_i denotes the weighted sum of the first i vectors of V, and U_i denotes the denominator of the i-th Softmax computation.

Let H and U be temporary variables in the computation. The prefix sum calculations in (13) - (16) can be computed using the following approach:

$$H = E \odot V \tag{18}$$
$$U = E \tag{19}$$
$$For\ j = 2 : L \tag{20}$$
$$H_j += H_{j-1} \tag{21}$$
$$U_j += U_{j-1} \tag{22}$$
$$End\ For. \tag{23}$$

The above computation A has a linear complexity, but it also requires L-1 time step cyclic calculation (Fig. 2A), which will consume a lot of time. Therefore, this paper proposes the following parallel computing algorithm B as an alternative to (20) - (23) to enhance computational speed:

$$For\ j = 0 : \lceil log_2 L \rceil \tag{24}$$
$$H[2^j :] += H[: -2^j] \tag{25}$$
$$U[2^j :] += U[: -2^j] \tag{26}$$
$$End\ For. \tag{27}$$

In each iteration of the loop operation, we update the data with information from the original data, which are separated by 2^j elements. This algorithm is a strategy that reduces the total number of loop computations by increasing the single step computational load. It only performs $\lceil log_2 L \rceil$ loop computations (Fig. 2B), where $\lceil log_2 L \rceil$ represents rounding up $log_2 L$.

To exemplify the computational efficiency superiority of Prefix Sum Algorithm B, we conducted experiments comparing the time consumption and peak memory usage of both algorithms across sequences with varying lengths (Fig. 3). The

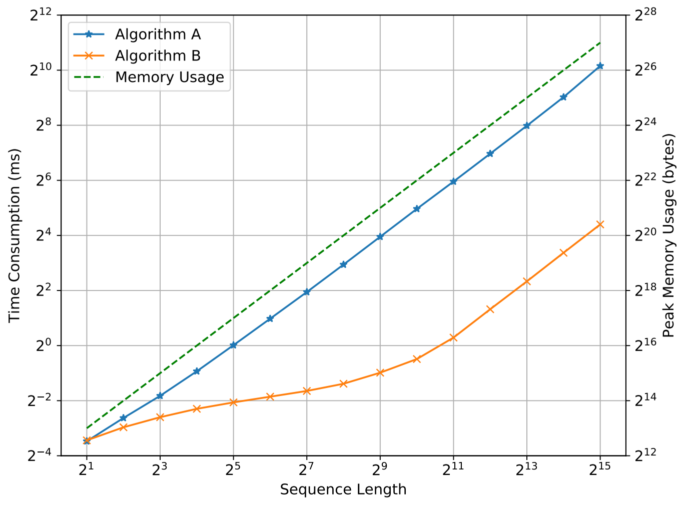

Fig. 3. Comparison of algorithm performance

findings indicate that Algorithm B exhibits considerably faster computational speed than Algorithm A, and this dissimilarity amplifies with increasing sequence length. Interestingly, both algorithms exhibit identical peak memory usage.

The preceding section addresses the specific PS-attention algorithm implemented in the decoder. Subsequent to this, an analysis is conducted on the PS-attention output to comprehend the underlying issues. The relationship between the i-th vector O_i and the $(i-1)$-th vector O_{i-1} of its output matrix

O can be expressed as follows:

$$O_i = O_{i-1} \odot \frac{U_{i-1}}{H_{i-1}} \odot \frac{H_i}{U_i} \tag{28}$$

$$= O_{i-1} \odot \frac{U_{i-1}}{U_{i-1}+E_i} \odot \frac{H_{i-1}+E_i \odot V_i}{H_{i-1}} \tag{29}$$

$$= O_{i-1} \odot \frac{1+\frac{E_i \odot V_i}{H_{i-1}}}{1+\frac{E_i}{U_{i-1}}}. \tag{30}$$

According to (13) and (14), when the index i is sufficiently large, $H_{i-1} \gg E_i \odot V_i$, and $U_{i-1} \gg E_i$. At this juncture:

$$\frac{1+\frac{E_i \odot V_i}{H_{i-1}}}{1+\frac{E_i}{U_{i-1}}} \approx 1. \tag{31}$$

Thus,

$$O_i \approx O_{i-1}. \tag{32}$$

This indicates that in PS-attention, for extended input sequences, the vectors corresponding to the latter part of the output sequence often persist at an approximate value. This poses a challenge in updating the output with new inputs in the time series, leading to a diminished fitting capability.

B. AIB-Attention

The primary limitation of PS-attention is attributed to its low inductive bias model. In (13) and (14), the function's output is a basic sum of its historical information. However, within the framework of inductive bias, the importance of inputs at various time points in the time series for the computation at the current time is evidently diverse. Consequently, in the AIB-attention introduced in this study, it becomes imperative to devise an algorithm that assigns an appropriate weight to inputs at varying distances from the output, signifying the varying importance of inputs at different distances. Ideally, it would be to replace the calculation in Fig. 1 with the form shown in Fig. 4. This can be achieved by replacing (13) and (14) with the following calculations:

$$H_i = \sum_{j=1}^{i} (a_{i-j} \odot E_j \odot V_j) \tag{33}$$

$$U_i = \sum_{j=1}^{i} (a_{i-j} \odot E_j) \tag{34}$$

$$s.t.\ a_0 = 1, a_{i-j} > 0,$$

where $a_{i-j} \in \mathbb{R}^D$ represents the scaling coefficient of the input at time step j to the output at time step i. To streamline the computation of (33) and (34), we can adopt algorithm B, as outlined in (24)-(27), and implement it as follows:

$$w = \left[w_1, ..., w_{\lceil log_2 Max(L) \rceil}\right] \tag{35}$$

$$W = Exp\left(Prefix\ sum\left(w\right)\right) \tag{36}$$

$$For\ j = 0 : \lceil log_2 L \rceil \tag{37}$$

$$H[2^j :] += H[: -2^j] \odot W_{j+1} \tag{38}$$

$$U[2^j :] += U[: -2^j] \odot W_{j+1} \tag{39}$$

$$End\ For. \tag{40}$$

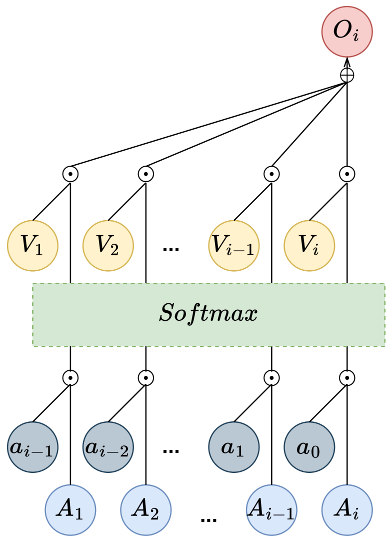

Fig. 4. The ideal computational process in AIB-attention

In the above formula, $w \in \mathbb{R}^{\lceil log_2 Max(L) \rceil \times D}$ is a trainable parameter, where each sub-vector is independent in the calculation. $W \in \mathbb{R}^{\lceil \log_2 Max(L) \rceil \times D}$ represents the weighted vector list, and $W_{j+1} \in \mathbb{R}^D$ is the j+1-th vector within it. W is computed from w, and its specific mathematical interpretation will be analyzed in the next subsection. By this method, in each iteration, H_i and U_i are weighted and added with the information of their current 2^j time step. In this manner, the attention mechanism can establish AIB, thereby scaling the reception of distant information. The algorithm we use to calculate Y in decoder is represented as Algorithm 1.

Algorithm 1 AIB-Attention in decoder

```
input : $X \in \mathbb{R}^{L \times D}$
output: $Y \in \mathbb{R}^{L \times D}$
$A, V = XW^A + b_1, XW^V + b_2$
$U = E = Exp(A)$
$H = E \odot V$
$W = Exp(Prefix\ sum(w))$
for $j \leftarrow 0$ to $\lceil \log_2 L \rceil - 1$ do
  $H[2^j :] += H[: -2^j] \odot W_{j+1}$
  $U[2^j :] += U[: -2^j] \odot W_{j+1}$
end
$O = \frac{H}{U}$
$Y = OW^O + b_3$
```

In the encoder, the output must capture the complete context. Consequently, in this research, a bidirectional information propagation strategy, akin to bidirectional RNN, was employed. It partitioning both H and U into forward and backward blocks. The computation in the forward blocks aligns with that in the decoder, involving the calculation of a weighted prefix sum. In contrast, the backward blocks switch to computing the weighted suffix sum. This approach accomplishes bidirectional information propagation, allowing the model to capture contextual information. Hence, in the AIB-attention encoder, it is only necessary to substitute equations

(37)-(40) with:

$$\overrightarrow{H} = H[:,: D|2] \tag{41}$$
$$\overleftarrow{H} = H[:, D|2 :] \tag{42}$$
$$\overrightarrow{U} = U[:,: D|2] \tag{43}$$
$$\overleftarrow{U} = U[:, D|2 :] \tag{44}$$
$$\overrightarrow{W} = W[:,: D|2] \tag{45}$$
$$\overleftarrow{W} = W[:, D|2 :] \tag{46}$$

$$For\ j = 0 : \lceil log_2 L \rceil \tag{47}$$
$$\overrightarrow{H}[2^j :] += \overrightarrow{H}[: -2^j] \odot \overrightarrow{W}_{j+1} \tag{48}$$
$$\overleftarrow{H}[: -2^j] += \overleftarrow{H}[2^j :] \odot \overleftarrow{W}_{j+1} \tag{49}$$
$$\overrightarrow{U}[2^j :] += \overrightarrow{U}[: -2^j] \odot \overrightarrow{W}_{j+1} \tag{50}$$
$$\overleftarrow{U}[: -2^j] += \overleftarrow{U}[2^j :] \odot \overleftarrow{W}_{j+1} \tag{51}$$
$$End\ For \tag{52}$$

$$H = Concat(\overrightarrow{H}, \overleftarrow{H}) \tag{53}$$
$$U = Concat(\overrightarrow{U}, \overleftarrow{U}), \tag{54}$$

where $\overrightarrow{H} \in \mathbb{R}^{L \times D|2}$ and $\overrightarrow{U} \in \mathbb{R}^{L \times D|2}$ represent the information used for the forward block computation, and $\overleftarrow{H} \in \mathbb{R}^{L \times D|2}$ and $\overleftarrow{U} \in \mathbb{R}^{L \times \frac{D}{2}}$ represent the information used for the backward block computation, where $D|2$ denotes D divided by 2. $\overrightarrow{W} \in \mathbb{R}^{\lceil \log_2 Max(L) \rceil \times D|2}$ and $\overrightarrow{W} \in \mathbb{R}^{\lceil \log_2 Max(L) \rceil \times D|2}$ are the weighted vector lists for the forward and backward blocks, respectively. The computations of the forward and backward blocks are independent, and after their completion, they are concatenated to form the new H and U.

The algorithm we use to calculate Y in encoder is represented as Algorithm 2:

Algorithm 2 AIB-Attention in encoder

```
input : $X \in \mathbb{R}^{L \times D}$
output: $Y \in \mathbb{R}^{L \times D}$
$A, V = XW^A + b_1, XW^V + b_2$
$U = E = Exp(A)$
$H = E \odot V$
$W = Exp(Prefix\ sum(w))$
$\overrightarrow{U} = U[:,: D|2]$
$\overleftarrow{U} = U[:, D|2 :]$
$\overrightarrow{H} = H[:,: D|2]$
$\overleftarrow{H} = H[:, D|2 :]$
$\overrightarrow{W} = W[:,: D|2]$
$\overleftarrow{W} = W[:, D|2 :]$
for $j \leftarrow 0$ to $\lceil \log_2 L \rceil - 1$ do
    $\overrightarrow{H}[2^j :] += \overrightarrow{H}[: -2^j] \odot \overrightarrow{W}_{j+1}$
    $\overleftarrow{H}[: -2^j] += \overleftarrow{H}[2^j :] \odot \overleftarrow{W}_{j+1}$
    $\overrightarrow{U}[2^j :] += \overrightarrow{U}[: -2^j] \odot \overrightarrow{W}_{j+1}$
    $\overleftarrow{U}[: -2^j] += \overleftarrow{U}[2^j :] \odot \overleftarrow{W}_{j+1}$
end
$U = Concat(\overrightarrow{U}, \overleftarrow{U})$
$H = Concat(\overrightarrow{H}, \overleftarrow{H})$
$O = \frac{H}{U}$
$Y = OW^O + b_3$
```

Using the above method, we have achieved AIB attention in both the encoder and decoder.

C. Mathematical Interpretation

This study achieves AIB-attention by incorporating scaling coefficients into the prefix sum algorithm. In this subsection, we will analyze the specific significance of W in the computation and explain why W is set as $Exp(Prefix\ sum(w))$ rather than other simpler forms. Due to the consistent weighting logic of AIB-attention in both the encoder and decoder, the analysis in this study focuses solely on its decoder form, omitting the analysis of its encoder form.

In the preceding subsection, the mathematical relationships delineated in (33) and (34) have been established by incorporating scaling coefficients into the prefix sum algorithm, as depicted in (35)-(40). More specifically, scaling coefficients corresponding to different distances between input and output values have been introduced to finely adjust the significance of information, thereby achieving adaptive inductive bias relationships.

All a_{i-j} in (33) and (34) can be succinctly represented utilizing vectors from W. Intuitively, when the distance is a power of two, it can be expressed as follows:

$$a_{2^i} = W_{i+1}. \tag{55}$$

When the distance is not a power of two, its scaling coefficient is the product of scaling coefficients corresponding to multiple distances that are powers of two, with the distance being the sum of their distances, as exemplified by:

$$a_7 = a_1 \odot a_2 \odot a_4 = W_1 \odot W_2 \odot W_3 \quad (56)$$

$$a_{10} = a_2 \odot a_8 = W_2 \odot W_4 \quad (57)$$

$$a_{23} = a_1 \odot a_2 \odot a_4 \odot a_{16} = W_1 \odot W_2 \odot W_3 \odot W_5. \quad (58)$$

Through the above-mentioned approach, the scaling relationships for any distance can be represented using subvectors of W.

In (36), this study employs the exponential function (Exp) to compute W. This is done to ensure that W is strictly greater than 0, mitigating the risk of vectors in U approaching or becoming zero, which could lead to training instability (as U serves as the denominator in subsequent computations).

For the scaling coefficients W, as defined in equations (35) and (36), a more straightforward approach is to set $W = Exp(w)$. Nevertheless, practical experiments revealed that this definition method might compromise the fitting capability of the neural network. We posit that during the training of AIB-attention, the weighting coefficients for longer distances should be influenced by those for shorter distances, promoting the stability of neural network optimization. For example, as W_2 increases, W_3 and W_4 should also experience an increase. This is grounded in the observation that longer-distance relationships often entail a composition of multiple shorter-distance relationships. Consequently, when changes occur in the shorter-distance relationships, a corresponding trend should manifest in the longer-distance relationships. We model the relationships among subvectors in W as follows:

$$W_1 = Exp(w_1) \quad (59)$$

$$W_{i+1} = W_i \odot Exp(w_{i+1}). \quad (60)$$

The following expressions can be derived:

$$W_i = \prod_{j=1}^{i} Exp(w_j) \quad (61)$$

$$= Exp\left(\sum_{j=1}^{i} w_j\right), \quad (62)$$

which can be succinctly written as:

$$W = Exp\left(Prefix\ sum(w)\right). \quad (63)$$

Through the aforementioned approach, the relationship between the scaling coefficients W and the trainable parameters w has been established, enabling the model to undergo training in a more stable manner.

D. Feed-Forward Networks

We implemented a feed-forward layer after the self-attention layer. This layer comprises two linear transformations with a GELU activation [57] in between. The configuration of this layer is consistent with the feedforward layer in the GPT-2 [4]:

$$FFN\left(X\right) = GELU\left(XW_1 + b_1\right)W_2 + b_2. \quad (64)$$

The dimensionality of input and output is d, and the dimensionality of the inner layer is d_{ff}.

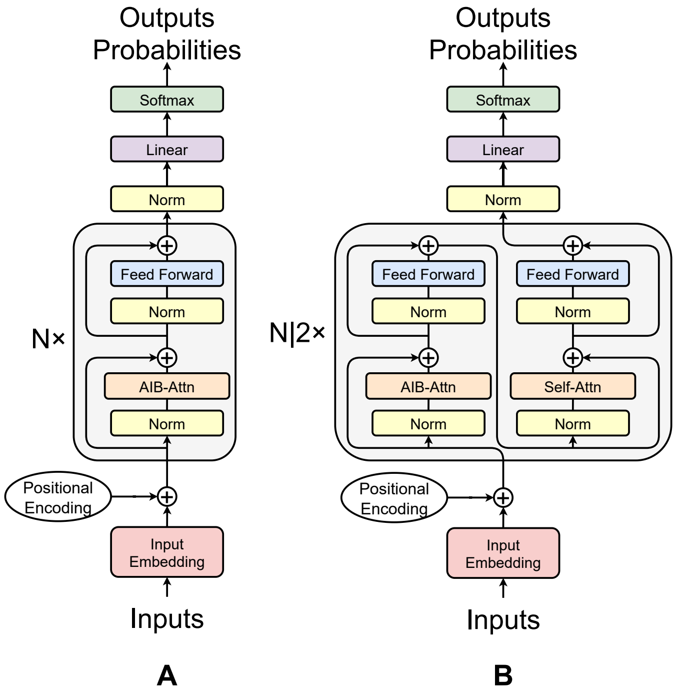

Fig. 5. The Architecture of the Moss Model.

TABLE II
HYPERPARAMETER OF THE NEURAL NETWORK IN THE LRA EXPERIMENT.

Task	N	d_{model}	d_{ff}	Batch size	Iterations	Epochs
ListOps	4	512	1024	32	5000	\
Text	4	256	1024	32	20000	\
Retrieval	4	128	512	32	5000	\
Image	1	32	64	256	\	200
Pathfinder	1	32	64	512	\	200
Path-X	1	32	64	64	\	200

E. Model Architecture

Neural sequence transduction models typically use an encoder–decoder structure. Moss also adopts this architecture, with stacked attention and point-wise fully connected layers in both the encoder and decoder. We use randomly initialized parameters as positional encoding, and the model preserves the same embedding dimension across all layers to ensure consistent residual connections. Each layer of the network contains two sub-layers. The first sub-layer is the attention layer, while the second sub-layer is a feedforward network, both of which use normalization and residual connections. This paper provides two stacking methods (Fig. 5):

A: Structure A consists of N layers, with the first sub-layer in each employing AIB-attention.,

B: Structure B also comprises N layers, where in odd layers, the first sub-layer employs AIB-attention, while in even layers, the first sub-layer employs self-attention.

This article recommends using the A structure in the encoder and the B structure in the decoder. The decoder interleaves the AIB-attention and the self-attention layer because better accuracy is achieved by this strategy than using only the AIB-attention or only the self-attention layer in our experiments. It is conjectured that the AIB-attention excels at modeling relative-position dependency whereas the self-attention layer excels at computing long-term relevance. This mixed strategy leverages the strengths of both structures.

F. Complexity Analysis

The main source of the computational cost of AIB-attention is its iterative calculation process. Its complexity can be expressed in the following manner:

$$O\left(\sum_{j=0}^{\lceil log_2 L\rceil-1} D\left(L-2^j\right)\right) \tag{65}$$

$$= O\left(D\left(L\left\lceil log_2 L\right\rceil - 2^{\lceil log_2 L\rceil}+1\right)\right) \tag{66}$$

$$\approx O\left(LDlog_2 L\right). \tag{67}$$

Hence, AIB-attention has lower computational complexity than self-attention when the sequence length L exceeds 4. Its complexity increases as $O\left(LDlog_2 L\right)$, which is significantly lower than the quadratic complexity of self-attention.

G. Initialization of Parameters

We initialize all parameters in the neural network to ensure gradient stability during training. All biases are set to zero and all weights are sampled from normal distributions, with zero mean and specific standard deviations. For position encoding, word embedding parameters, W^A, W^V, and W_1, we use $\frac{1}{\sqrt{D}}$. For w, W^O, and W_2, we use 1, $\sqrt{\frac{1-\frac{2}{D}}{2ND}}$, and $1.7047\sqrt{\frac{1-\frac{2}{D}}{2Nd_{ff}}}$, respectively.

IV. Experiments

In the previous section, we described the encoder and decoder architectures based on AIB-attention. In this section, we assess the performance of these architectures on several long-sequence tasks using the Long Range Arena (LRA) [36], wikitext-103 [58], text8 [38], and enwik8 [39] data sets. We implemented our neural network models in PyTorch and used Adam [59] as the optimizer. In addition, we trained and tested our models on a single NVIDIA GeForce RTX 3090 GPU.

A. Long-range Modeling on LRA

We evaluated the encoder performance of our model on the LRA data set, a systematic benchmark for efficient sequence processing models. The LRA data set measures model quality under long-sequence scenarios across six tasks and various data types, such as text, natural images, synthetic images, and mathematical expressions. The sequence lengths in the LRA data set vary from 1K to 16K, requiring different reasoning abilities, such as similarity, structural, and visual-spatial. The LRA data set can help comprehensively examine the generalization ability, computational efficiency, memory usage, and other aspects of sequence processing models.

This model's attention calculation transmits information in a directional manner. Therefore, the classifier built by this model cannot use the common [CLS] method [60]. Instead, it uses the average of the output layer of the neural network as the output.

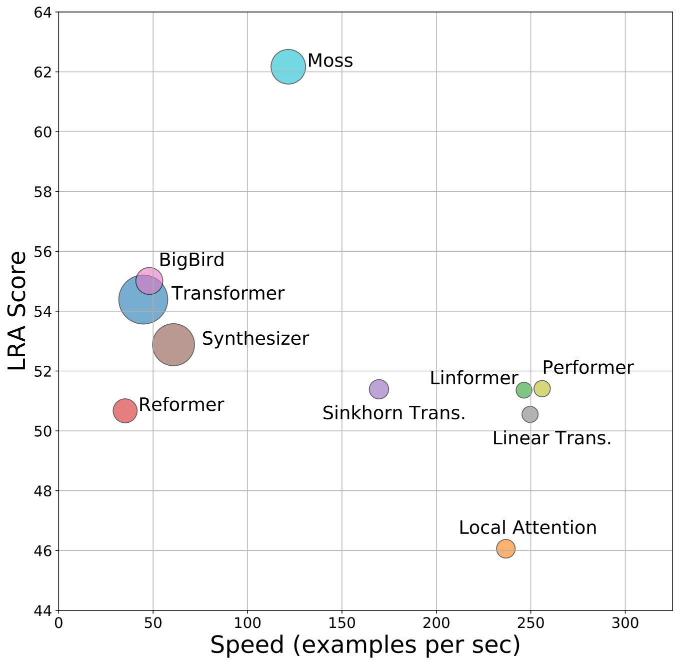

Fig. 6. Performance (y axis), speed (x axis), and memory footprint (size of the circles) of different models.

To ensure a fair comparison of different models, we follow the same model configuration as [36], including the structure and number of iterations. Table II summarizes the neural network hyperparameters used in each task. Table III reports the accuracy scores of various tasks in the experiment. Despite the limited number of iterations that prevent the model from converging in these tasks, it still surpasses all other Transformer-based models and baseline methods in terms of average accuracy.

This model outperforms some Transformer-based sequence processing models in terms of computation speed and memory usage. Table IV presents the comparison of speed and memory usage among various models with different input lengths (1K, 2K, 3K, and 4K). Following the experimental setting of [36], we evaluate all models on a byte-level classification task with the same batch size.

Fig. 2 depicts the trade-off among memory usage, computation speed, and performance for different models. Most models that reduce the complexity of the Transformer model sacrifice its performance. However, our model outperforms them with lower time and space complexity.

A common drawback of most Transformer models is their weak generalization ability. In contrast, our model establishes an attention mechanism based on adaptive inductive bias, which enables the model to learn position-dependent inductive relations. This endows it with superior robustness. Table V presents the training and testing accuracy of various models on the image classification benchmark. It is evident that all Transformer-like models suffer from severe overfitting, with a large gap between the two scores, despite using strong regularization mechanisms. Without applying any regularization

TABLE III
EXPERIMENTAL RESULTS OF LONG-RANGE ARENA BENCHMARK.

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Transformer	36.37	64.27	57.46	42.44	71.40	FAIL	54.39
Local Attention [23]	15.82	52.98	53.39	41.46	66.63	FAIL	46.06
Sparse Trans [24]	17.07	63.58	59.59	44.24	71.71	FAIL	51.24
Longformer [25]	35.63	62.85	56.89	42.22	69.71	FAIL	53.46
Linformer [26]	35.70	53.94	52.27	38.56	76.34	FAIL	51.36
Reformer [27]	37.27	56.10	53.40	38.07	68.50	FAIL	50.67
Sinkhorn Trans [28]	33.67	61.20	53.83	41.23	67.45	FAIL	51.39
Synthesizer [29]	36.99	61.68	54.67	41.61	69.45	FAIL	52.88
BigBird [30]	36.05	64.02	59.29	40.83	74.87	FAIL	55.01
Linear Trans [31]	16.13	65.90	53.09	42.34	75.30	FAIL	50.55
Performer [32]	18.01	65.40	53.82	42.77	77.05	FAIL	51.41
Our Work	39.48	70.54	63.92	50.30	84.76	63.99	62.17

TABLE IV
COMPARISON OF SPEED AND MEMORY USAGE AMONG VARIOUS MODELS WITH DIFFERENT INPUT LENGTHS.

Model	Steps per second				Peak Memory Usage (GB)			
	1K	2K	3K	4K	1K	2K	3K	4K
Transformer	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Local Attention	1.1	1.7	3.2	5.3	0.49	0.29	0.19	0.14
Linformer	1.2	1.9	3.7	5.5	0.44	0.21	0.18	0.10
Reformer	0.5	0.4	0.7	0.8	0.56	0.37	0.28	0.24
Sinkhorn Trans	1.1	1.6	2.9	3.8	0.55	0.31	0.21	0.16
Synthesizer	1.1	1.2	2.9	1.4	0.76	0.75	0.74	0.74
BigBird	0.9	0.8	1.2	1.1	0.91	0.56	0.40	0.30
Linear Trans	1.1	1.9	3.7	5.6	0.44	0.22	0.15	0.11
Performer	1.2	1.9	3.8	5.7	0.44	0.22	0.15	0.11
Our Work	1.0	1.6	2.2	2.7	1.24	0.87	0.63	0.50

TABLE V
TEST AND TRAIN ACCURACY OF DIFFERENT MODELS ON THE IMAGE CLASSIFIICATION TASK.

Model	Test Accuracy	Train Accuracy
Transformer	42.44	69.45
Local Attention	41.46	63.19
Linformer	38.56	97.23
Reformer	38.07	68.45
Sinkhorn Trans	41.23	69.21
Synthesizer	41.61	97.31
BigBird	40.83	71.49
Linear Trans	42.34	65.61
Performer	42.77	73.90
Our Work	50.30	52.88

algorithm, our model achieves lower training accuracy than all Transformer models, but higher testing accuracy than all Transformer models and is very close to the training accuracy level. This demonstrates that our model is much more robust than Transformer models.

B. Natural Language Modelling

We tested the decoder of our model on various natural language data sets, including WikiText-103, Text8, and enwik8, without any additional training data. The results indicate that our model significantly outperforms the GPT-2 [4] model with the same structural parameters. Moreover, our model outperforms the current SOTA in character-level language modeling task of enwik8.

*1) **Language Modelling on WikiText-103**:* In this section, we compare Moss with GPT-2's small and medium models. GPT-2 is a Transformer-based language model that has been trained unsupervised on a large corpus of English text and can generate coherent and fluent text. We use WikiText-103—a large-scale English text data set with 1.03 million tokens extracted from Wikipedia—as our evaluation benchmark. We follow the same text encoding method and training settings as in the GPT-2 paper. The input sequence consists of 1024 consecutive tokens randomly cropped from the document, and the batch size is 1. In addition, we use the Adam optimizer and adjust the learning rate according to the GPT-2 paper.

The structural parameters of Moss in this experiment are consistent with GPT-2's small medium and large models. The structural parameters and experiment results are presented in Table VI. Without using any additional training data, Moss-small achieves a perplexity (PPL) of 21.27 on WikiText-103, Moss-medium achieves a PPL of 19.42, and Moss-large achieves a PPL of 17.88, both of which are much better than GPT-2 small model's 37.50, GPT-2 medium model's 26.37 and GPT-2 large model's 22.05. This demonstrates that Moss has a stronger ability and generalization performance on English text-generation tasks.

*2) **Character-Level Language Modelling**:* Character-level language modeling greatly increases the length and complexity of the token sequence, which challenges the ability of language models to capture long-distance dependencies. Conventional sequence models struggle with this task. We experimented on Text8, and enwik8 data sets without using any extra training data. In the Text8 and enwik8 data sets, we set the last 10M characters as the test set. In the language modeling task of enwik8, we also attempted to add the last 900M characters of enwik9 [61] as an additional training set. In addition, we randomly cropped 1024 consecutive characters as the neural network input. Table VII compares Moss-large with GPT-2 large and SOTA (GPT-2 xl) in terms of bits per character (BPC). Moss outperforms GPT-2 in all tasks.

V. CONCLUSION AND PROSPECTS

In this paper, we introduced Moss, a novel neural network that exhibits superior fitting ability and robustness. Moss mod-

TABLE VI
STRUCTURAL PARAMETERS AND EXPERIMENTAL RESULTS OF MOSS AND GPT-2 IN WIKITEXT-103.

Model	N	d_{model}	d_{ff}	Parameters	PPL
GPT-2 small	12	768	3072	114M	37.50
GPT-2 medium	24	1024	4096	345M	26.37
GPT-2 large	36	1280	5120	752M	22.05
Moss-small	12	768	3072	117M	21.27
Moss-medium	24	1024	4096	328M	19.42
Moss-large	36	1280	5120	712M	17.88

TABLE VII
BPC AND PARAMETERS OF MOSS-LARGE AND GPT-2.

Dataset	GPT-2		Moss-large	
	BPC	Parameters	BPC	Parameters
Text8	1.02 (large) [4]	752M	0.99	712M
enwik8	0.97 (large) [4]	752M	0.915	712M
enwik8	0.93 (SOTA) [4]	1542M	0.915	712M

els long-distance dependencies through an adaptive inductive bias attention mechanism that has near-linear complexity and enables parallel computation. In both encoder and decoder experiments, the model surpasses all Transformer-based models in terms of accuracy with the same number of structural parameters. The model also excels in language modeling tasks with long sequences and achieves SOTA results, even with a small parameter size in certain tasks.

As future work, we plan to apply this model to image recognition, translation, and ultra-long character-level text generation, among other domains. We also aim to explore more applications of Moss models in multimodal learning and cross-lingual transfer learning. Due to its remarkable fitting ability and robustness, we believe this model has great potential in various research fields. However, we also acknowledge that our model has a few limitations and challenges, such as the scalability of very large data sets and vocabularies, the interpretability of the attention mechanism, and the ethical issues of generating realistic texts. Moreover, we note that our current experiments are conducted on medium-scale networks due to the limited computational resources. We intend to investigate the performance of larger-scale networks in the future. We hope our work can inspire more research on these aspects.

ACKNOWLEDGMENTS

This work was supported by Guangzhou Youth Science and Technology Education Project under Grant No. KP2023243. The name of our proposed neural network, Moss, is inspired by the AI "Moss" in the movie The Wandering Earth, a science fiction film that depicts the human struggle to save the Earth from a dying sun. We hope that our work can contribute to the advancement of AI and its applications for the benefit of humanity.

REFERENCES

[1] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[5] N. Zhang, "Learning adversarial transformer for symbolic music generation," *IEEE transactions on neural networks and learning systems*, 2020.

[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[7] Y. Luan and S. Lin, "Research on text classification based on cnn and lstm," in *2019 IEEE international conference on artificial intelligence and computer applications (ICAICA)*. IEEE, 2019, pp. 352–355.

[8] S. N. Tran, A. d. Garcez, T. Weyde, J. Yin, Q. Zhang, and M. Karunanithi, "Sequence classification restricted boltzmann machines with gated units," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4806–4815, 2020.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[10] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[11] B. Zhang, D. Xiong, J. Xie, and J. Su, "Neural machine translation with gru-gated attention model," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4688–4698, 2020.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.

[14] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[15] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu *et al.*, "Summary of chatgpt/gpt-4 research and perspective towards the future of large language models," *arXiv preprint arXiv:2304.01852*, 2023.

[16] R. M. Schmidt, "Recurrent neural networks (rnns): A gentle introduction and overview," *arXiv preprint arXiv:1912.05911*, 2019.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–28, 2022.

[19] M. Hahn, "Theoretical limitations of self-attention in neural sequence models," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 156–171, 2020.

[20] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, 2022.

[21] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 344–16 359, 2022.

[22] C. Wu, F. Wu, T. Qi, Y. Huang, and X. Xie, "Fastformer: Additive attention can be all you need," *arXiv preprint arXiv:2108.09084*, 2021.

[23] M. Arar, A. Shamir, and A. H. Bermano, "Learned queries for efficient local attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 841–10 852.

[24] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[25] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[26] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[27] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[28] 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.

[29] Y. Tay, D. Bahri, D. Metzler, D. Juan, Z. Zhao, and C. Zheng, "Synthesizer: Rethinking self-attention in transformer models. arxiv 2020," *arXiv preprint arXiv:2005.00743*, vol. 2, 2020.

[30] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in neural information processing systems*, vol. 33, pp. 17 283–17 297, 2020.

[31] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.

[32] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, D. Belanger, L. Colwell *et al.*, "Masked language modeling for proteins via linearly scalable long-context transformers," *arXiv preprint arXiv:2006.03555*, 2020.

[33] S. Hao, D.-H. Lee, and D. Zhao, "Sequence to sequence learning with attention mechanism for short-term passenger flow prediction in large-scale metro system," *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 287–300, 2019.

[34] D. Ju, S. Roller, S. Sukhbaatar, and J. E. Weston, "Staircase attention for recurrent processing of sequences," *Advances in Neural Information Processing Systems*, vol. 35, pp. 13 203–13 213, 2022.

[35] A. Hernández and J. M. Amigó, "Attention mechanisms and their applications to complex systems," *Entropy*, vol. 23, no. 3, p. 283, 2021.

[36] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler, "Long range arena: A benchmark for efficient transformers," *arXiv preprint arXiv:2011.04006*, 2020.

[37] B. Li, Y. Hu, X. Nie, C. Han, X. Jiang, T. Guo, and L. Liu, "Dropkey," *arXiv preprint arXiv:2208.02646*, 2022.

[38] M. Mahoney, "Large text compression benchmark," 2011.

[39] "enwik8 ·datasets at hugging face," https://huggingface.co/datasets/enwik8, accessed on May 26, 2023.

[40] Y. Kim, C. Denton, L. Hoang, and A. M. Rush, "Structured attention networks," *arXiv preprint arXiv:1702.00887*, 2017.

[41] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 452, pp. 48–62, 2021.

[42] G. W. Lindsay, "Attention in psychology, neuroscience, and machine learning," *Frontiers in computational neuroscience*, vol. 14, p. 29, 2020.

[43] Q. Lai, S. Khan, Y. Nie, H. Sun, J. Shen, and L. Shao, "Understanding more about human and machine attention in deep neural networks," *IEEE Transactions on Multimedia*, vol. 23, pp. 2086–2099, 2020.

[44] S. Ghaffarian, J. Valente, M. Van Der Voort, and B. Tekinerdogan, "Effect of attention mechanism in deep learning-based remote sensing image processing: A systematic literature review," *Remote Sensing*, vol. 13, no. 15, p. 2965, 2021.

[45] A. Galassi, M. Lippi, and P. Torroni, "Attention in natural language processing," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 10, pp. 4291–4308, 2020.

[46] 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.

[47] X.-Y. Zhang, C. Li, H. Shi, X. Zhu, P. Li, and J. Dong, "Adapnet: Adaptability decomposing encoder-decoder network for weakly supervised action recognition and localization," *IEEE transactions on neural networks and learning systems*, 2020.

[48] Z. Ji, Y. Zhao, Y. Pang, X. Li, and J. Han, "Deep attentive video summarization with distribution consistency learning," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 4, pp. 1765–1775, 2020.

[49] S. Takase and S. Kiyono, "Lessons on parameter sharing across layers in transformers," *arXiv preprint arXiv:2104.06022*, 2021.

[50] L. Wu, J. Li, Y. Wang, Q. Meng, T. Qin, W. Chen, M. Zhang, T.-Y. Liu *et al.*, "R-drop: Regularized dropout for neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10 890–10 905, 2021.

[51] Z. Zhang, J. Yang, and H. Zhao, "Retrospective reader for machine reading comprehension," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 16, 2021, pp. 14 506–14 514.

[52] M. Lippi, M. A. Montemurro, M. Degli Esposti, and G. Cristadoro, "Natural language statistical features of lstm-generated texts," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3326–3337, 2019.

[53] J. Baxter, "A model of inductive bias learning," *Journal of artificial intelligence research*, vol. 12, pp. 149–198, 2000.

[54] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[55] A. Goyal and Y. Bengio, "Inductive biases for deep learning of higher-level cognition," *Proceedings of the Royal Society A*, vol. 478, no. 2266, p. 20210068, 2022.

[56] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5301–5310.

[57] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[58] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[60] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[61] Marcus Hutter, "Hutter prize for lossless compression of human knowledge," 2006, http://prize.hutter1.net/, Last accessed on 2023-05-26.

Lingkai Hu received a bachelor's degree from the School of Mechanical and Electrical Engineering, Guangzhou University, China, in 2021. He is currently studying for a master's degree in mechanical engineering at Guangzhou University, Guangzhou, China. His current research interests include deep learning (DL), sequence analysis, object detection, attention mechanism, and multilevel feature fusion.

Feng zhan is currently pursuing an undergraduate degree in Electrical Engineering at Guangzhou University. His research focuses on fault diagnosis using deep learning techniques, with a particular emphasis on aspects such as few-shot learning, diffusion models, and self-attention.

Wenkai Huang received his B.S. degree and M.S. from Guangdong University of Technology. He received his Ph.D. from Guangzhou University. In 2007, he joined the School of Mechanical and Electrical Engineering at Guangzhou University, where he is currently an Associate Professor. His research interests are AI, robot vision, medical image processing, and soft robotics.

Weiming Gan is currently studying for a master's degree in machinery engineering at Guangzhou University, Guangzhou, China. His current research interests include industrial data mining recognition and analysis, deep learning (DL), and image processing technology.