Ultron: A High-Performance Sequence-Processing Model Based on the Composite Mapping Layer

Lingkai Hu[a,*], Feng Zhan[a,*], Wenkai Huang[a,**], Weiming Gan[a], Kunbo Han[b]

[a]*School of Mechanical and Electrical Engineering, Guangzhou University, Guangzhou, 510006, China*
[b]*School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, 510006, China*

Abstract

A crucial task in deep-learning research, sequence processing involves extracting semantic or structural relationships between distant elements in a sequence. However, existing sequence-processing methods have several limitations, such as high levels of complexity, difficulty in modeling long-term dependencies, and insufficient generalization ability. To address these issues, Ultron, a high-performance sequence-processing model based on the composite mapping layer (CM layer), is proposed. The CM layer is a novel neural layer that uses a composite function of multiple information-mapping functions to capture long-term information dependencies. Compared with the self-attention mechanism, the CM layer has a lower complexity of $O(LlogL)$. Ultron adopts the CM layer as the core component of the encoder and decoder structures, achieving the same parallel-computing capability as the Transformer model and improving its fitting ability and robustness. A high-quality benchmark dataset for Chinese language modeling, zhwiki520, is also constructed based on the complete text of Chinese Wikipedia as of May 20, 2023. Ultron is evaluated on long-range arena, wikitext-103, zhwiki520, and enwik8. The experimental results show that Ultron surpasses Transformer and its variants in all the tasks and meets the current criteria for state-of-the-art performance on the enwik8 dataset. This paper contributes a novel and effective method to the sequence-processing field and provides a valuable evaluation metric for Chinese language modeling tasks.

*Co-first author.
**Corresponding author.

Keywords: Composite mapping, Sequence processing, Long-term dependency, Language modeling

1. Introduction

Sequence processing is a fundamental problem in various fields, such as natural language processing (NLP) [1, 2, 3], speech recognition [4, 5, 6], machine translation [7, 8, 9, 10, 11], and bioinformatics [12, 13, 14, 15]. It involves tasks that require the analysis, understanding, generation, or transformation of sequences of different modalities (e.g., text, audio, or image) [2]. The main challenge of sequence processing is to effectively capture the long-term dependencies in sequences, which are the semantic or structural relations between distant elements [16, 17, 18].

Traditional sequence-processing models rely mainly on either recurrent neural networks (RNNs) [19, 20] or convolutional neural networks (CNNs) [21, 22]. RNNs can handle arbitrary-length sequences by recursively updating their hidden states, but they suffer from gradient vanishing or exploding, parallelization difficulty, and ineffective long-term dependency modeling [23, 24]. CNNs can extract local features through multiple convolutional operations and expand their receptive field through pooling or skip connections, thus achieving parallelization and long-term dependency modeling. However, they also have limitations, including a fixed receptive field size and sequential-information loss [25, 26].

In recent years, Transformers [27] have gained widespread attention and application. They are sequence-processing models that use self-attention mechanisms, which can model global dependencies by computing the relevance between any two positions in a sequence and have efficient parallel training and inference capabilities. Transformer-based models have improved performance significantly in various NLP tasks and have inspired many variants and extensions, such as BERT [28], GPT [29], and XLNet [30].

However, Transformer-based models face several challenges and issues[31]. First, the self-attention mechanism has a quadratic time complexity and space complexity with respect to the input-sequence length, which requires a large number of parameters and computational resources. This hinders their performance in long-sequence tasks [16]. Second, they tend to disregard the relative relationships and dependencies among different positions in the sequence, which may affect the model's sensitivity to long-term dependencies and structural information [32]. Third, they may have limited generalizability or transferability when dealing with texts from diverse languages or domains [33, 34]. Therefore, designing a more efficient, robust, and scalable sequence-processing model is a valuable research endeavor.

This study proposes a novel neural layer for sequence processing called the composite mapping layer (CM layer). By applying a composite function of multiple information-mapping functions, the CM layer can capture long-term information dependencies. Based on the CM layer, a high-performance neural network called Ultron is designed. Ultron has lower complexity than Transformer and outperforms all the Transformer variants in the experiments on fitting ability (Fig .1). Most existing research on language modeling has focused on English datasets, such as Penn Treebank [35], wikitext-103 [36], and enwik8 [37]. However, Chinese, a widely used language, lacks a general benchmark dataset for this task. To address this gap, zhwiki520[1], a high-quality benchmark dataset for Chinese language modeling, is constructed based on the full text of Chinese Wikipedia as of May 20, 2023 [38]. It is preprocessed using Su's script [39] and released for public use.

This paper makes the following contributions:

1. A novel neural layer for sequence processing called the CM layer is proposed, which models long-term information dependencies through the composition of multiple information-mapping functions. It has a lower complexity of $O(LlogL)$ than the self-attention mechanism.
2. A novel model called Ultron is proposed, which incorporates CM layers into both the encoder and decoder structures. Like the Transformer model, the Ultron model can perform parallel computation, and it has better fitting ability and robustness than the Transformer model.
3. The Ultron model is evaluated on four datasets: long-range arena (LRA), wikitext-103, zhwiki520, and enwik8. The experimental results demonstrate that the Ultron model outperforms Transformer and its variants in all the tasks and meets the current criteria for state-of-the-art performance on the enwik8 dataset.

The paper is structured as follows. Section 2 introduces the research background of the sequence-processing model. Section 3 examines the principle and implementation of the CM layer as well as the structure and details of the

[1]Our dataset zhwiki520 is published on: https://kaggle.com/datasets/lingkaihu/zhwiki520

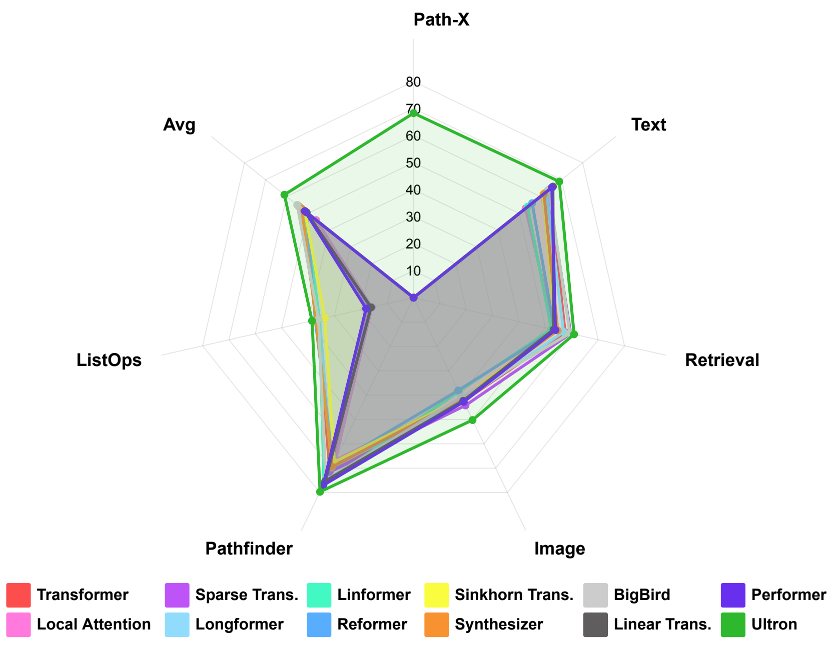

Fig. 1. Accuracy comparison of Ultron and Transformer variants in the LRA [33] dataset.

Ultron model. Section 4 outlines the experimental settings and results. Section 5 concludes the paper by summarizing its contributions and suggesting directions for future work.

2. Background

Sequence processing is a crucial machine learning problem with various applications in NLP, speech recognition, machine translation, bioinformatics, and other domains. However, sequence processing also poses some challenges, the most important of which concerns how to model the long-term dependencies in the sequence effectively.

The current mainstream sequence-processing models fall into three categories: RNN-based models, CNN-based models, and self-attention-based models [40, 41]. Each of these types of models has a unique set of strengths and weaknesses. This section will briefly introduce their principles and performance and discuss some of their problems and limitations in long-sequence tasks.

2.1. Models based on RNNs

RNNs have a neural-network structure that can handle sequences of arbitrary length by recurrently updating the hidden state, which captures the sequential and contextual information in the sequence [19]. RNNs have achieved good results in many sequence-processing tasks, but they also have some drawbacks, such as those identified by [23, 24]:

1. RNNs suffer from the problem of gradient vanishing or exploding, which means that the gradient decays or grows exponentially as the time step increases in the backpropagation process, leading to difficult or unstable training.
2. RNNs are difficult to parallelize, because each time step depends on the calculation result of the previous time step, which limits the training speed and scale of RNN.
3. RNNs cannot effectively model long-term dependencies, because the hidden state gradually loses or confuses past information as the time step increases.

Some improved and extended models, such as long short-term memory [42] and gated recurrent units [43], have been proposed to overcome the shortcomings of RNNs. These models introduce gate structures to control the information flow and forgetting degree, thereby alleviating the gradient-vanishing and long-term-dependency problems. However, when the sequence length is extremely long, these improvements are still weak. Further, some studies have introduced cross-time skip connections to alleviate memory decay [44, 45], but this approach breaks the continuity of the sequence, reduces the recognition accuracy of the neural network, and increases the complexity of the neural network. Other studies have incorporated attention into RNN computation to establish long-term dependencies [46, 47], but the resulting models have the complexity and robustness problems that characterize other attention-based techniques. In general, these improved models are still limited by the RNN structure itself, and they still have significant drawbacks when the input sequence is excessively long. Moreover, they require more parameters and computational resources [48].

2.2. Models based on CNNs

CNNs are neural networks that can extract local features and model spatial- or temporal-translation invariance by performing multiple convolutional operations. They also expand the receptive field by using pooling or skip connections [21]. CNNs have been widely applied in image processing, computer vision, and other fields. They can also process the sequential data used for tasks such as text classification and semantic analysis. In contrast to RNNs, CNNs can achieve parallelization, because each convolution kernel depends only on local input rather than on the entire sequence. This improves the neural network's training speed. Moreover, CNNs can capture features of different scales by using convolution kernels of different sizes, thus achieving multiscale modeling [22]. However, CNNs have some drawbacks, the most important of which are as follows [25, 26]:

1. The receptive-field size of CNNs is fixed, meaning that each output depends only on a fixed length of input. This may cause CNNs to fail to capture long-term dependencies that lie beyond the range of the receptive field.
2. CNNs cannot capture the sequential information in a sequence, meaning that the correspondence between each output and input is fixed. This may cause CNNs to fail to handle position changes or structure changes in the sequence.
3. CNNs requires a large number of convolution kernels to cover features of different positions and different scales. This increases the parameter and computational complexity of CNNs.

To overcome the shortcomings of CNNs, some improved and extended models have been proposed, such as dynamic convolutional neural networks [49], dilated convolution [26], and deformable convolutional networks [50]. Yang et al. proposed the Shuffle network, which uses a depthwise separable CNN to form a new group convolutional fusion unit. It also uses shuffling and an intraskip-connection mechanism to expand the receptive field of CNNs [51]. Xue et al. proposed a novel fault-diagnosis method based on a local binary temporal CNN, which uses a new temporal module with dilated causal convolution to establish long-term dependencies. It also adopts a local binary convolution layer to reduce computational parameters and improve robustness [52]. These models introduce new types of convolution kernels to adapt to features of different positions and different scales, thereby enhancing the flexibility and adaptability of CNNs. However, these models cannot solve the fundamental problem of CNNs, that is, their inability to obtain a global receptive field [53].

2.3. Models based on self-attention

The self-attention mechanism is a technique that computes and aggregates the relevance between any two positions in a sequence using matrix-multiplication and -normalization operations [27]. It has been widely applied in NLP, machine translation, and other fields and has been used to construct sequence-processing models based purely on self-attention mechanisms, such as Transformer. The self-attention mechanism has the following advantages over RNNs and CNNs [54, 55]:

1. The self-attention mechanism can model global dependency relationships, meaning that each output can depend on the entire sequence of inputs rather than on local or fixed-length inputs. This improves the self-attention mechanism's sensitivity and expressiveness for long-term dependency relationships.
2. The self-attention mechanism can achieve parallelization, because each output depends only on the input matrix and weight matrix rather than on the previous output or previous time step. This improves the self-attention mechanism's training speed and scale.
3. The self-attention mechanism can capture position information and structure information by using position encoding or relative-position encoding to assign different weights or biases to different positions. This enhances the understanding of the self-attention mechanism for sequential-order information and structure information.

However, the self-attention mechanism has some drawbacks, the most important of which are as follows [16, 32, 33, 34]:

1. Both the time complexity and space complexity of the self-attention mechanism are quadratic functions of the input-sequence length. Consequently, a large number of parameters and extensive computational resources are required for the self-attention mechanism to achieve efficient sequence modeling.
2. The self-attention mechanism tends to ignore the relative relationship and dependency between different positions in the sequence, meaning that the relevance between each output and input depends only on their content rather than on their position. As a result, the self-attention mechanism may be insensitive to long-term dependency and structure information in the sequence.

3. The self-attention mechanism may have insufficient generalizability or transferability when dealing with different languages or domains of text, meaning that the relevance between each output and input depends only on their semantics rather than on their syntax or pragmatics. As a result, the self-attention mechanism may not fully capture the language features and domain knowledge in the sequence.

To overcome the self-attention mechanism's drawbacks, some studies have proposed improved and extended models, such as Linformer, Performer, and Longformer [56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]. Essentially, these models compress the information in the sequence into lower-dimensional data and either perform information mapping based on the compressed data [56, 67, 68] or sparsify or group queries and keys [69, 70, 71], thereby reducing the complexity. However, this compression process may lead to information loss or limit the attention mechanism to local information, which prevents the neural network from obtaining complete, global information. Therefore, in practical performance comparisons, these models often have a weaker fitting ability than the original Transformer [33].

3. Methodology

3.1. CM layer

The fitting process of artificial neural networks involves adjusting the network parameters to approximate or simulate an unknown-function relationship. For example, in a unidirectional-sequence-prediction model, the hidden layer function relationship is defined as $Y = F(X)$, where X is an input sequence of length L. The output of Y at the nth time step can be expressed as

$$Y_n = F_n\left(X_1, \cdots, X_n\right). \tag{1}$$

This equation can be rewritten as

$$H_n = \sum_{1}^{n} F_{(i,n)}\left(X_i\right) \tag{2}$$

$$Y_n = G_n\left(H_n\right), \tag{3}$$

where (2) represents the fusion of the context features of the entire sequence, $F_{(i,n)}$ is the information-mapping function from the ith time step to the nth time step, and (3) is the feature-mapping function of the fused information. The complexity of this calculation is

$$O\left(F\left(X\right)\right)=O\left(\sum_{1}^{L}i\right)=O\left(L^{2}/2\right). \tag{4}$$

To model the information relationship between any two time points, both the number and complexity of parameters in this calculation are quadratic functions of the sequence length, which consumes a large amount of computation. One method of simplifying the computation is to assume that the information-mapping relationship between any two time points with the same distance is the same, such as:

$$F_{(i,n)}=F_{(i+T,n+T)}, \tag{5}$$

and assume that the long-term-mapping relationship can be expressed as the iteration of the unit-mapping function of the distance number, such as:

$$F_{(i,n)}=F_{1}^{n-i}. \tag{6}$$

In (6), we use F_1 to represent any unit-mapping function, such as $F_{(i,i+1)}$. By iterating multiple unit-information-mapping functions F_1, we simulate the long-term information-mapping function, such as:

$$Y_{n+1}=F_{1}\left(Y_{n},X_{n+1}\right). \tag{7}$$

Eq. (5) is a typical Markov process, in which the output of each step is a function of the output and input of the previous step. RNNs can be considered a special case of it. In this computation method, due to parameter sharing, the number of parameters is greatly reduced; due to the presence of only L iterations, the complexity is reduced to $O\left(L\right)$. This method improves the model's computational efficiency to some extent, but the iterative structure prevents the model operation from being parallelized. When the sequence length is long, the information tends to decay in the iteration, making it difficult to establish long-term dependency relationships.

To address the limitations of existing sequence-processing models, this study proposes a novel algorithm improvement called the CM layer. The CM layer consists of several information-mapping functions with distances of 2 to the power of integers, such as: $F_1, F_2, F_4, F_8, \cdots$.

These functions can be composed to form any information-mapping function with an arbitrary distance, such as

$$F_{(3,10)} = F_7 \;\; = F_1 \circ F_2 \circ F_4 \tag{8}$$

$$F_{(1,15)} = F_{14} = F_2 \circ F_4 \circ F_8 \tag{9}$$

$$F_{(7,18)} = F_{11} = F_1 \circ F_2 \circ F_8 \tag{10}$$

$$F_{(6,21)} = F_{15} = F_1 \circ F_2 \circ F_4 \circ F_8. \tag{11}$$

This technique enables long-term information mapping through shorter computation paths. It can be efficiently implemented with parallel computing, and its specific implementation technique is expressed as Algorithm 1: In this paper, the mapping function is defined as a linear layer with parameters $W_{2^i} \in \mathbb{R}^{d\times d}$ for each information-mapping function and $W^Y \in \mathbb{R}^{d\times d}$ for the feature-mapping function in (3). Thus, all the information-mapping functions can be simplified as a single linear transformation by multiplying several weight matrices. Therefore, the length of the information-propagation path in this structure is 1.

Algorithm 1: Information Mapping in Decoder

```
Input: $X \in \mathbb{R}^{L\times d}$
Output: $Y \in \mathbb{R}^{L\times d}$
1 for $i \leftarrow 0$ to $\lceil \log_2 L \rceil$ do
2     $X[2^i:] += X[:-2^i]W_{2^i}$;
3 end
4 $Y = YW^Y$;
```

To enable neurons to access complete contextual information in the classification model, a bidirectional CM layer is designed, as shown in Algorithm 2.

This algorithm allows neurons to obtain comprehensive contextual information.

3.2. Feedforward networks

To introduce nonlinearity, we define a nonlinear feedforward network between each CM layer. This network consists of two linear layers with a Leaky ReLU [72] activation function. Leaky ReLU is defined as

$$Leaky\ ReLU\,(X) = Max\,(0.01X, X)\,. \tag{12}$$

Algorithm 2: Information Mapping in Encoder

```
    Input: $X \in \mathbb{R}^{L \times d}$
    Output: $Y \in \mathbb{R}^{L \times d}$
1   $\overrightarrow{X} = X[:, : d|2]$;
2   $\overleftarrow{X} = X[:, d|2 :]$;
3   for $i \leftarrow 0$ to $\lceil \log_2 L \rceil$ do
4       $\overrightarrow{X}\ [2^i :] += \overrightarrow{X}\ [: -2^i]\ \overrightarrow{W}_{2^i}$;
5       $\overleftarrow{X}\ [: -2^i] += \overleftarrow{X}\ [2^i :]\ \overleftarrow{W}_{2^i}$;
6   end
7   $X = concat(\overrightarrow{X}, \overleftarrow{X})$;
8   $Y = YW^Y$;
```

The feedforward network is defined as

$$FFN\,(X) = Leaky\ ReLU\,(XW_1 + b_1)\,W_2 + b_2. \tag{13}$$

We choose Leaky ReLU as the activation function because it has a nonzero gradient in the negative region, which can prevent the neuron from becoming inactive. The input- and output-embedding dimensions of the feedforward network are d, and the intermediate-layer-embedding dimension is d_{ff}.

3.3. Model architecture

Conventional sequence-processing models typically adopt an encoder–decoder architecture for various tasks. Ultron follows this design and adds learnable position encoding as an input layer, enabling the model to capture relative-position relationships. Fig. 2 shows its encoder and decoder structures, where Fig. 2A is the encoder and Fig. 2B is the decoder:

Encoder: The encoder consists of N layers, each containing two sublayers. The first sublayer is a bidirectional CM layer that models both forward and backward contexts. The second sublayer is a feedforward network that applies a nonlinear transformation to the output of the CM layer. Each sublayer has layer normalization and residual connection for gradient stability.

Decoder: The decoder also consists of N layers, each containing two sublayers. In the odd-numbered layers, the first sublayer is a unidirectional CM layer that models only the forward context. The second

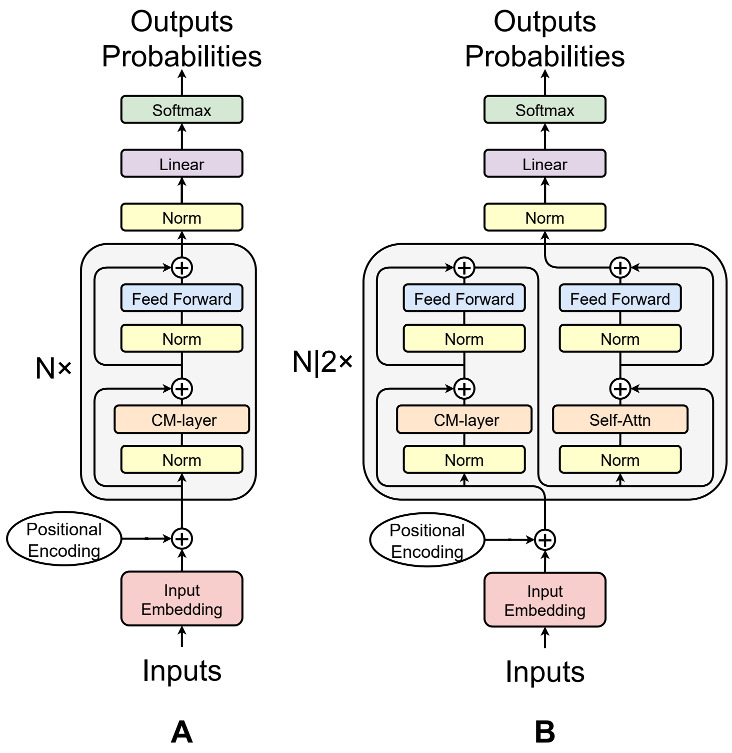

Fig. 2. The architecture of the Ultron.

The decoder interleaves the CM layer and the self-attention layer because better accuracy is achieved by this strategy than using only the CM layer or only the self-attention layer in our experiments. It is conjectured that the CM layer 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.

3.4. Complexity analysis

The complexity of the CM layer arises from its iterative computation, which requires approximately $log_2 L$ iterations. The complexity can be ex-

Table 1
Comparison of complexity, maximum path length, and perceptual field of various neural network layers [16]

Layer Type	Complexity per Layer	Maximum Path Length	Receptive Field
Self-Attention	$O(L^2D)$	$O(1)$	$O(L)$
CM Layer	$O(LD^2Log_2L)$	$O(1)$	$O(L)$
Fully Connected	$O(L^2D^2)$	$O(1)$	$O(L)$
Convolutional	$O(KLD^2)$	$O(log_K L)$	$O(K)$
Recurrent	$O(LD^2)$	$O(1)$	$O(L)$

"L" is the sequence length, "D" is the number of hidden state dimensions, "K" is the convolution kernel size, and "T" is the sampling period of the cuneate layer.

pressed as

$$O\left(\int_0^{\lceil \log_2 L \rceil} d^2 \left(L - 2^x\right) dx\right) = O\left(\frac{d^2}{\ln 2}\left(L \ln L + L + 1\right)\right) \tag{14}$$

$$\approx O\left(Ld^2 \log_2 L\right). \tag{15}$$

Hence, the CM layer exhibits a complexity of $L \log_2 L$, which is lower than that of the self-attention mechanism. The CM layer demonstrates a more pronounced advantage when L is large. Table 1 compares the performance metrics of common neural-network layers and highlights the CM layer's relatively high scores in these metrics.

4. Experiments

In the previous section, a detailed account of Ultron's structure, which is based on the CM layer, was provided, and the feasibility and effectiveness of the CM layer were theoretically analyzed. In this section, the performance of our model is evaluated on several benchmark experiments, including LRA, wikitext-103, enwik8, and zhwiki520, a benchmark for Chinese language modeling that was constructed. The neural-network models were implemented using PyTorch, and Adam [73] was employed as the optimizer. The training and testing of the models were conducted on a single NVIDIA GeForce RTX 4090 GPU.

4.1. Long-range modeling on LRA

This study evaluated the encoder of our model on the LRA dataset, a systematic method for assessing sequence-processing models. The LRA dataset consists of six tasks and datasets that test various aspects of sequence-processing models, such as generalization ability, computational efficiency, memory usage, and reasoning skills, on long sequences. The LRA dataset covers diverse data types and modalities, including text, natural images, synthetic images, and mathematical expressions. The sequence lengths range from 1 to 16 K, and the data require similarity, structural, or visual-spatial reasoning.

Table 2
Hyperparameters of neural networks 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

Table 3
Experimental results for the LRA 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 [57]	15.82	52.98	53.39	41.46	66.63	FAIL	46.06
Sparse Trans [58]	17.07	63.58	59.59	44.24	71.71	FAIL	51.24
Longformer [59]	35.63	62.85	56.89	42.22	69.71	FAIL	53.46
Linformer [60]	35.70	53.94	52.27	38.56	76.34	FAIL	51.36
Reformer [61]	37.27	56.10	53.40	38.07	68.50	FAIL	50.67
Sinkhorn Trans [62]	33.67	61.20	53.83	41.23	67.45	FAIL	51.39
Synthesizer [63]	36.99	61.68	54.67	41.61	69.45	FAIL	52.88
BigBird [64]	36.05	64.02	59.29	40.83	74.87	FAIL	55.01
Linear Trans [65]	16.13	65.90	53.09	42.34	75.30	FAIL	50.55
Performer [66]	18.01	65.40	53.82	42.77	77.05	FAIL	51.41
Our Work	38.49	68.91	60.88	50.24	79.71	68.27	61.08

In this experiment, the encoder of Ultron was used as the neural network for classification and the temporal average of its output as its output layer. Ultron's accuracy, generalization ability, computational efficiency, and memory usage were compared with those of some Transformer-based models. To ensure a fair comparison of the various models, the same model configuration as [33] was followed, including structure and number of iterations. Table 2 summarizes the neural-network hyperparameters used in each task. Table 3 shows the accuracy of each model in each subtask of LRA. Even though Ultron was limited by the low number of iterations, which prevented it from converging well in these tasks, its accuracy still surpassed that of all the Transformer-based models in all the tasks.

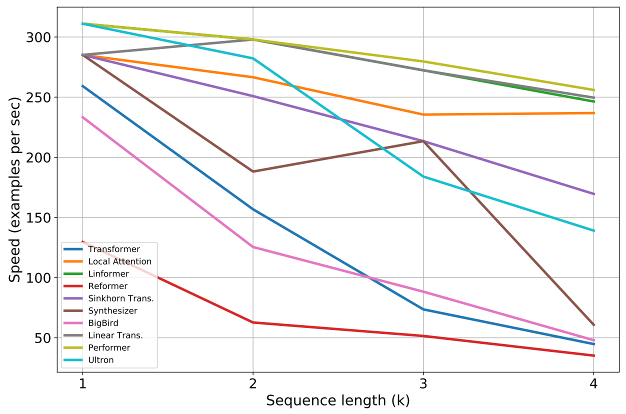

Fig. 3. Comparison of speed among various models with different input lengths.

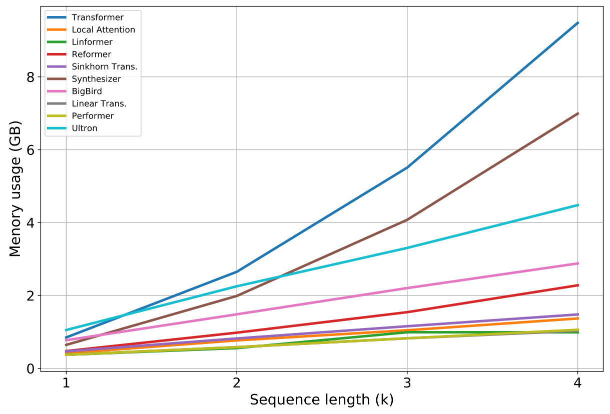

Fig. 4. Comparison of memory usage among various models with different input lengths.

Following [33]'s experimental setup, we evaluated all the models on byte-level-classification tasks with the same batch size. Figs. 3 and 4 present a comparison of speed and memory usage among various models with different input lengths (1 K, 2 K, 3 K, and 4 K). Ultron outperformed some of the Transformer-based sequence-processing models in terms of computational speed and memory usage.

Fig. 5 shows the trade-off between memory usage, computational speed, and accuracy among various models. Most models that reduce the complexity of Transformer-based models sacrifice accuracy. However, our model achieves both higher accuracy and lower time and space complexity than Transformer. A noticeable drawback of Transformer and its variants is their low level of robustness. Unlike attention-based models, the Ultron model uses position-related features on sequences, which have a higher inductive bias for information that is close in spatial distance. This makes it robust in long-sequence tasks. Table 4 shows the training and testing accuracy of various models on the image-classification benchmark. The neural networks in the comparison use strong regularization algorithms, but they suffer from severe overfitting, with a large gap between their training and testing accuracy. In contrast, Ultron does not use any regularization algorithms in this experiment, but it has the highest testing accuracy, and its training and testing accuracy are very close. This shows that Ultron has better robustness than the other models.

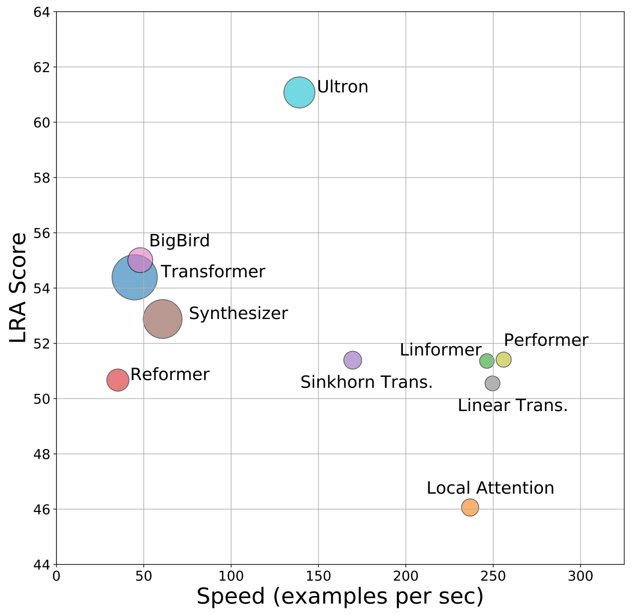

Fig. 5. Performance (y-axis), speed (x-axis), and memory footprint (size of the circles) of various models.

Table 4
Testing and training accuracy of various models on the image-classification 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	50.24	52.60

4.2. Natural language modeling

The language modeling performance of the proposed model's decoder was compared with that of the GPT model on various natural language datasets, including wikitext-103, enwik8, and zhwiki520, which is a Chinese language dataset created for this study. Three metrics were used to evaluate the models: loss, computational speed, and memory usage. The model sizes were varied and the metrics were measured on each dataset. The results show that the proposed model outperforms the GPT model in all metrics with the same structural parameters.

4.2.1. Token-level language modeling on wikitext-103

In this section, three sets of comparative experiments are presented, each based on one of the three structures of GPT-2. The GPT-2 model is a pretrained Transformer-based model that, like other GPT models, consists of self-attention mechanisms and feedforward neural networks. The performance of these models is evaluated using the wikitext-103 dataset, which is a large English corpus dataset containing 1.03 million tokens from English Wikipedia articles. 1024 consecutive tokens are randomly selected from the documents as the input sequence of the neural network.

Table 5 shows the three model-size groups that are compared in this experiment. No additional training datasets are used for Ultron. The results indicate that Ultron outperforms GPT-2 in all the experiments, achieving

Table 5
Structure parameters and experimental results for Ultron and GPT in wikitext-103.

Model	N	d_{model}	d_{ff}	PPL
GPT-small	12	768	3072	37.50
GPT-medium	24	1024	4096	26.37
GPT-large	32	1280	5120	22.05
Ultron-small	12	768	3072	23.69
Ultron-medium	24	1024	4096	21.22
Ultron-large	32	1280	5120	19.54

lower perplexity (PPL) and demonstrating superior performance in English token-level text-generation tasks.

4.2.2. Byte-level language modeling

This study investigated the challenges of language modeling for Chinese, a language that lacks explicit word boundaries. Approaches that use words as the smallest input units depend on the performance and efficiency of segmentation tools, which may affect the model's performance. Chinese vocabulary also exhibits many variations, such as synonyms, near-synonyms, homophones, and variant characters, which may lead to semantic loss or ambiguity due to tokenization [74]. Therefore, the gb18030 encoding method was adopted to encode Chinese text into byte-level-sequence data. Byte-level units are character-based, so they do not require segmentation tools. They can theoretically handle any language, and they avoid out-of-vocabulary tokens. They also have a high degree of flexibility. Moreover, because the vocabulary size is 256, they can significantly reduce the parameter size and complexity of the word-embedding module.

The text from the main body of the entries in Chinese Wikipedia on May 20, 2023 was collected, and text information was extracted based on Su's script [39]. A total of 1 355 711 entries were obtained, of which 13 557 entries were split as a test set, and the rest were used as a train set. The character \x00 was used as a separator between each entry in the dataset. After preprocessing, the training set contained 1 868 574 744 characters and the test set contained 18 817 080 characters. For performance testing, three scales of neural networks with the same parameter settings presented in Table 5 were used. The neural network input was 1024 consecutive characters randomly cropped from the document. Fig. 6 shows the experimental results.

We also evaluated this model's byte-level English language modeling performance. The enwik9 dataset [75] was used to train the model, and the enwik8 dataset's last 10 M characters were employed for testing. The content of the test set was intentionally excluded from the training set. Fig. 7 shows the experimental results. As Figs. 6 and 7 show, Ultron outperformed the GPT model in both Chinese and English byte-level language modeling tasks. In particular, its bit per character (BPB) achieved state-of-the-art performance (0.93) [76] in the enwik8 test. This demonstrates Ultron's superior performance in byte-level language modeling.

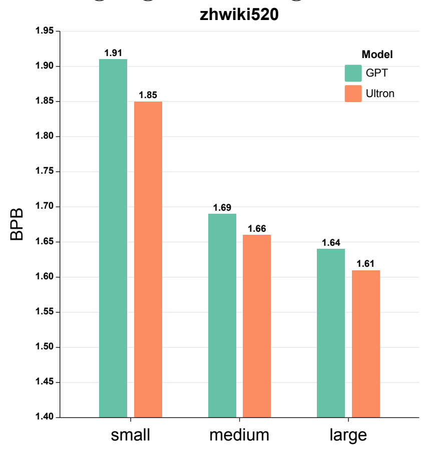

Fig. 6. Experimental results for Ultron and GPT in zhwiki520.

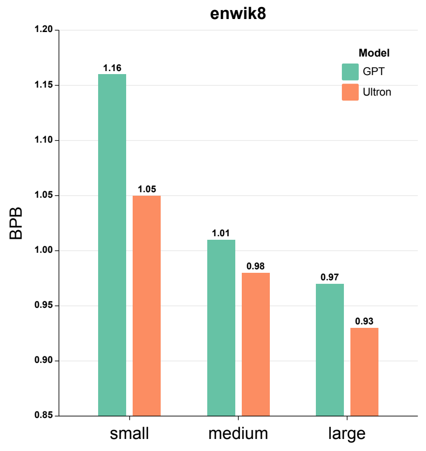

Fig. 7. Experimental results for Ultron and GPT in enwik8.

5. Conclusion and prospects

This paper presents Ultron, a novel sequence-processing model that surpasses Transformer in terms of complexity, robustness, and fitting ability. Ultron's CM layer captures long-term information dependencies by combining mapping functions, which reduces the complexity to $O(LlogL)$ and enables parallel computing. This advantage becomes more evident as the sequence length increases. In the encoder experiments, Ultron outperforms Transformer and its variants in all the tasks. In the decoder experiments, Ultron has a lower degree of loss than the GPT model of the same scale on English and Chinese language modeling tasks. On the enwik8 dataset, Ultron meets the current criteria for state-of-the-art performance. Our study demonstrates the effectiveness and potential of Ultron, which have important implications for NLP.

In future research, the aim is to apply Ultron to a broader array of domains, including image classification, signal detection, and translation. Given its exceptional fitting capability, Ultron demonstrates versatility across a wide spectrum of applications. At the algorithmic level, efforts will be directed toward further optimization of Ultron's memory utilization and computing speed, as well as the development of a more tailored regularization algorithm. It is acknowledged that the experiments conducted are constrained by the available computing resources, and plans are in place to assess Ultron's performance on larger parameter sizes and longer sequences in forthcoming research. The aspiration is that this paper will serve as a source of inspiration for subsequent studies in this field.

Acknowledgments

This work was supported by Guangzhou Youth Science and Technology Education Project under Grant No. KP2023243 & KP2023245.

References

[1] P. Kumar, B. Raman, A bert based dual-channel explainable text emotion recognition system, Neural Networks 150 (2022) 392–407.

[2] D. W. Otter, J. R. Medina, J. K. Kalita, A survey of the usages of deep learning for natural language processing, IEEE transactions on neural networks and learning systems 32 (2020) 604–624.

[3] J. C.-W. Lin, Y. Shao, Y. Djenouri, U. Yun, Asrnn: A recurrent neural network with an attention model for sequence labeling, Knowledge-Based Systems 212 (2021) 106548.

[4] D. Ren, G. Srivastava, A novel natural language processing model in mobile communication networks, Mobile Networks and Applications 27 (2022) 2575–2584.

[5] J. Lei, X. Zhu, Y. Wang, Bat: Block and token self-attention for speech emotion recognition, Neural Networks 156 (2022) 67–80.

[6] A. Mukhamadiyev, M. Mukhiddinov, I. Khujayarov, M. Ochilov, J. Cho, Development of language models for continuous uzbek speech recognition system, Sensors 23 (2023) 1145.

[7] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, Advances in neural information processing systems 27 (2014).

[8] B. Zhang, D. Xiong, J. Xie, J. Su, Neural machine translation with gru-gated attention model, IEEE transactions on neural networks and learning systems 31 (2020) 4688–4698.

[9] A. Fan, S. Bhosale, H. Schwenk, Z. Ma, A. El-Kishky, S. Goyal, M. Baines, O. Celebi, G. Wenzek, V. Chaudhary, et al., Beyond english-centric multilingual machine translation, The Journal of Machine Learning Research 22 (2021) 4839–4886.

[10] B. Saunders, N. C. Camgoz, R. Bowden, Continuous 3d multi-channel sign language production via progressive transformers and mixture density networks, International journal of computer vision 129 (2021) 2113–2135.

[11] X. Yin, D. Gromann, S. Rudolph, Neural machine translating from natural language to sparql, Future Generation Computer Systems 117 (2021) 510–519.

[12] D. Grechishnikova, Transformer neural network for protein-specific de novo drug generation as a machine translation problem, Scientific reports 11 (2021) 321.

[13] S. Mahony, P. V. Benos, T. J. Smith, A. Golden, Self-organizing neural networks to support the discovery of dna-binding motifs, Neural Networks 19 (2006) 950–962.

[14] J. Li, B. Wu, X. Sun, Y. Wang, Causal hidden markov model for time series disease forecasting, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 12105–12114.

[15] J. Ren, K. Song, C. Deng, N. A. Ahlgren, J. A. Fuhrman, Y. Li, X. Xie, R. Poplin, F. Sun, Identifying viruses from metagenomic data using deep learning, Quantitative Biology 8 (2020) 64–77.

[16] T. Lin, Y. Wang, X. Liu, X. Qiu, A survey of transformers, AI Open (2022).

[17] A. Mahalunkar, J. D. Kelleher, Understanding recurrent neural architectures by analyzing and synthesizing long distance dependencies in benchmark sequential datasets, arXiv preprint arXiv:1810.02966 (2018).

[18] W. Sun, X. Peng, X. Wan, Capturing long-distance dependencies in sequence models: A case study of chinese part-of-speech tagging, in: Proceedings of the Sixth International Joint Conference on Natural Language Processing, 2013, pp. 180–188.

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

[20] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: Lstm cells and network architectures, Neural computation 31 (2019) 1235–1270.

[21] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, G. D. Hager, Temporal convolutional networks for action segmentation and detection, in: proceedings

of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 156–165.

[22] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, arXiv preprint arXiv:1609.03499 (2016).

[23] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Gated feedback recurrent neural networks, in: International conference on machine learning, PMLR, 2015, pp. 2067–2075.

[24] M. O. Turkoglu, S. D'Aronco, J. D. Wegner, K. Schindler, Gating revisited: Deep multi-layer rnns that can be trained, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2021) 4081–4092.

[25] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, D. Yu, Convolutional neural networks for speech recognition, IEEE/ACM Transactions on audio, speech, and language processing 22 (2014) 1533–1545.

[26] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, arXiv preprint arXiv:1511.07122 (2015).

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

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

[29] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., Improving language understanding by generative pre-training (2018).

[30] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, Q. V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, Advances in neural information processing systems 32 (2019).

[31] S. Chaudhari, V. Mithal, G. Polatkan, R. Ramanath, An attentive survey of attention models, ACM Transactions on Intelligent Systems and Technology (TIST) 12 (2021) 1–32.

[32] M. Hahn, Theoretical limitations of self-attention in neural sequence models, Transactions of the Association for Computational Linguistics 8 (2020) 156–171.

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

[34] D. Variš, O. Bojar, Sequence length is a domain: Length-based overfitting in transformer models, arXiv preprint arXiv:2109.07276 (2021).

[35] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, S. Khudanpur, Recurrent neural network based language model., in: Interspeech, volume 2, Makuhari, 2010, pp. 1045–1048.

[36] S. Merity, C. Xiong, J. Bradbury, R. Socher, Pointer sentinel mixture models, arXiv preprint arXiv:1609.07843 (2016).

[37] M. Mahoney, Large text compression benchmark, 2011.

[38] W. Foundation, Index of zhwiki, `https://dumps.wikimedia.org/zhwiki/`, 2023. Accessed on Sep 20, 2023.

[39] J. Su, Obtain and process chinese wikipedia corpus, 2017. URL: `https://spaces.ac.cn/archives/4176`.

[40] Z. Wang, Q. Yu, J. Wang, Z. Hu, A. Wang, Grammar correction for multiple errors in chinese based on prompt templates, Applied Sciences 13 (2023) 8858.

[41] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al., A survey of large language models, arXiv preprint arXiv:2303.18223 (2023).

[42] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[43] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

[44] J. Kim, M. El-Khamy, J. Lee, Residual lstm: Design of a deep recurrent architecture for distant speech recognition, arXiv preprint arXiv:1701.03360 (2017).

[45] B. Yue, J. Fu, J. Liang, Residual recurrent neural networks for learning sequential representations, Information 9 (2018) 56.

[46] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, H. Cao, X. Cheng, M. Chung, M. Grella, K. K. GV, et al., Rwkv: Reinventing rnns for the transformer era, arXiv preprint arXiv:2305.13048 (2023).

[47] Z. Zhao, N. Lv, R. Xiao, S. Chen, A novel penetration state recognition method based on lstm with auditory attention during pulsed gtaw, IEEE Transactions on Industrial Informatics (2022).

[48] W. Fang, Y. Chen, Q. Xue, Survey on research of rnn-based spatio-temporal sequence prediction algorithms, Journal on Big Data 3 (2021) 97.

[49] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, arXiv preprint arXiv:1404.2188 (2014).

[50] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 764–773.

[51] Y. Xian, Y. Sun, W. Wang, S. M. Naqvi, Convolutional fusion network for monaural speech enhancement, Neural Networks 143 (2021) 97–107.

[52] Y. Xue, R. Yang, X. Chen, Z. Tian, Z. Wang, A novel local binary temporal convolutional neural network for bearing fault diagnosis, IEEE Transactions on Instrumentation and Measurement (2023).

[53] I. Khalfaoui-Hassani, T. Pellegrini, T. Masquelier, Dilated convolution with learnable spacings, arXiv preprint arXiv:2112.03740 (2021).

[54] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, C. Zhang, Disan: Directional self-attention network for rnn/cnn-free language understanding, in: Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.

[55] Z. Niu, G. Zhong, H. Yu, A review on the attention mechanism of deep learning, Neurocomputing 452 (2021) 48–62.

[56] C. Wu, F. Wu, T. Qi, Y. Huang, X. Xie, Fastformer: Additive attention can be all you need, arXiv preprint arXiv:2108.09084 (2021).

[57] M. Arar, A. Shamir, A. H. Bermano, Learned queries for efficient local attention, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10841–10852.

[58] R. Child, S. Gray, A. Radford, I. Sutskever, Generating long sequences with sparse transformers, arXiv preprint arXiv:1904.10509 (2019).

[59] I. Beltagy, M. E. Peters, A. Cohan, Longformer: The long-document transformer, arXiv preprint arXiv:2004.05150 (2020).

[60] S. Wang, B. Z. Li, M. Khabsa, H. Fang, H. Ma, Linformer: Self-attention with linear complexity, arXiv preprint arXiv:2006.04768 (2020).

[61] N. Kitaev, Ł. Kaiser, A. Levskaya, Reformer: The efficient transformer, arXiv preprint arXiv:2001.04451 (2020).

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

[63] Y. Tay, D. Bahri, D. Metzler, D. Juan, Z. Zhao, C. Zheng, Synthesizer: Rethinking self-attention in transformer models. arxiv 2020, arXiv preprint arXiv:2005.00743 2 (2020).

[64] 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 33 (2020) 17283–17297.

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

[66] 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).

[67] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, Y. W. Teh, Set transformer: A framework for attention-based permutation-invariant neural networks, in: International conference on machine learning, PMLR, 2019, pp. 3744–3753.

[68] X. Ma, X. Kong, S. Wang, C. Zhou, J. May, H. Ma, L. Zettlemoyer, Luna: Linear unified nested attention, Advances in Neural Information Processing Systems 34 (2021) 2441–2453.

[69] J. Ding, S. Ma, L. Dong, X. Zhang, S. Huang, W. Wang, F. Wei, Longnet: Scaling transformers to 1,000,000,000 tokens, arXiv preprint arXiv:2307.02486 (2023).

[70] Y. Sun, L. Dong, S. Huang, S. Ma, Y. Xia, J. Xue, J. Wang, F. Wei, Retentive network: A successor to transformer for large language models, arXiv preprint arXiv:2307.08621 (2023).

[71] A. Roy, M. Saffar, A. Vaswani, D. Grangier, Efficient content-based sparse attention with routing transformers, Transactions of the Association for Computational Linguistics 9 (2021) 53–68.

[72] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al., Rectifier nonlinearities improve neural network acoustic models, in: Proc. icml, volume 30, Atlanta, GA, 2013, p. 3.

[73] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[74] C.-R. Huang, P. Šimon, S.-K. Hsieh, L. Prévot, Rethinking chinese word segmentation: tokenization, character classification, or wordbreak identification, in: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, 2007, pp. 69–72.

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

[76] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (2019) 9.