Evolving Masked Low-Rank Transformer for Long Text Understanding

Chenjing Liu[a,b], Xiangru Chen[a,b], Jie Lin[b], Peng Hu[a,b], Junfeng Wang[a,*], Xue Geng[b,*]

[a] *College of Computer Science, Sichuan University, 610065, Chengdu, China*
[b] *Institute for Infocomm Research, Agency for Science, Technology and Research, 138632, Singapore*

Abstract

Long sequence text processing is time-consuming owing to the ultra-large-scale self-attention computing. Recent advances demonstrate the attention in transformer can be accelerated by redundancy removal, and there are various sparse variants for attention in large sequences are proposed, which leads to state-of-the-art performance on language and vision task. Low-rank method achieve outstanding success in the field of efficient transformer. The dynamic token sparsification is efficiently time-saving and cost-saving, which can be easily extended to prune redundant spans and to yield semantic features. Evolutionary algorithm is attractive for selecting hyperparameter which is of significant importance in effectiveness. Motivated by these works, we propose an efficient transformers model, termed EMLT, to alleviate time and cost without sacrificing the accuracy. EMLT effectively combines strengths of Low-rank trans-

*This work is financially supported by the China Scholarship Council (202106240095, 202106240096). This work is also supported in part by the National Key Research and Development Program under Grant 2022YFB3305203; in part by the National Natural Science Foundation of China under Grant U2133208 and Grant 62101368; and in part by the Sichuan Youth Science and Technology Innovation Team under Grant 2022JDTD0014 and in part by the Major Science and Technology Special Project of Sichuan Province under Grant 2022ZDZX0008. This work is also supported in part by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funds (Project No. A1892b0026 and Project No. A19E3b0099).

*Corresponding author

Email addresses: cjliu1996@gmail.com (Chenjing Liu), cxrasdfg@gmail.com (Xiangru Chen), jie.dellinger@gmail.com (Jie Lin), penghu.ml@gmail.com (Peng Hu), wangjf@scu.edu.cn (Junfeng Wang), geng_xue@i2r.a-star.edu.sg (Xue Geng)

formers, dynamic token sparsification and evolutionary algorithm to ulteriorly cut redundant token and meanwhile maintains the original precision, which can achieve a linear memory and time complexity. We compress transformer in three stages. Firstly, sliding window is validated as local attention to capture fine-grained dependency semantics. After that, low-rank approximation of attention matrix is applied as global attention to store long-range dependency semantics, and aggregated with local attention. On this basis, we consistently prune redundant token in accordance with importance score to further sparse the attention operation. Finally, Evolutionary algorithm is utilized to optimize the hyper-parameters of every layer. The results of comprehensive experiments and analysis show that our method can rival others on accuracy, and outperforms others on efficiency by a large margin in terms of the computational complexity.
Keywords: evolutionary computing, efficient transformer, genetic algorithm, natural language processing, neural architecture search

---

1. Introduction

Transformer is a deep learning [1] model based self-attention [2]. Recent developments in Transformers were investigated quite intensively, which have been widely adopted in the field of NLP and made great progress [3].

Although the transformer has become a paradigm in processing text-based media, the huge demand of computational cost is one of the biggest challenges in the deployment of the transformer models [4, 5], especially for the very long document processing.In the realm of language processing, long text processing is specifically defined as the practice of managing documents that exceed the conventional length of 512 tokens [4, 6, 7]. The computational exigencies and memory requisites of the transformer model are conspicuously accentuated when confronted with the task of processing longer input sequences. This intricate computational load emanates from a multitude of sources, encompassing the substantial parameter count necessitating training and the heightened computational demands imposed by the pairwise attention mechanism. As a corollary,

the deployment of transformer-based models for endeavors entailing protracted textual inputs on memory-constrained hardware or time-critical platforms is frequently deemed unviable. This formidable challenge underscores the imminent imperative for the development of more memory-efficient and computationally streamlined iterations of the transformer architecture, thereby rendering such applications more amenable to real-world, practical deployment.

To tackle this computational bottleneck, the community has tried many efforts to devise the intense attention operation to lower the computational complexity raised by computing the pairwise similarities. Then the computational complexity would be reduced by a large margin. Most of works focus on changing the pattern of the attended behaviour, such as sliding window. e.g., Child *et al.* [8] proposed Sparse Transformer, where they employed the factorized attention schemes to reduce the complexity of the original attention. And they've got the a better attention pattern with lower computational complexity. Qiu *et al.* [9] presented Blockwise Transformer. In their work, they introduced a sparse block structure into the attention matrix. Therefore, the memory consumption and the training time could be reduced. BigBird proposed by [10] was another method used the sparse attention pattern that reduced this quadratic dependency to linear. In addition to global and local sparse attention, BigBird also introduced random attention pattern to mitigate the information loss.

Low-rank based methods have been proposed as an alternative solution to address the issue of declining performance due to the capacity challenges of the self-attention layer and the high inductive biases of predefined sparse pattern-based methods. These methods transform the original input sequence into a lower resolution version, significantly reducing the computational burden through the use of a shorter representative sequence. However, each specific token is only able to attend to a limited number of tokens in these previously mentioned predefined sparse pattern-based methods, which may result in a decline in performance for certain tasks that are not suitable for their high inductive biases. One such method is Linformer. In their approach, a low-rank matrix is utilized to down-sample the original Keys and Values K, V, resulting in linear attention.

However, the projection matrix employed in Linformer is global, meaning that it is shared by all input sequences, potentially overlooking the local semantic information present within these sequences. This can be a limitation of the method, as it may not always be able to capture the nuances and complexity of certain tasks. Another method, Nyströmformer, was proposed by Xiong *et al.* [11]. In that approach, the Nyström method is used to approximate standard self-attention. Despite this, the performance of those low-rank methods is not always as good as full attention or sparse attention on tasks that require fine-grained local information, as noted in [5]. This may be due to the loss of high-fidelity token-wise information, which can be critical for certain types of tasks.

A considerable number of the previously mentioned techniques for improving the efficiency of transformer models have primarily focused on revising and optimizing the attended behavior of the model, while neglecting the issue of redundancy in the raw input tokens. However, recent research has uncovered the fact that certain words in the original input data, such as function words, do not necessarily require advanced modeling in the higher layers of the model [12]. This is because these types of words contain minimal information and have already been effectively processed by pretrained transformer-based models in the lower layers. Therefore, the main objective of pruning token is to remove these words that seem to have less importance in order to decrease the complexity introduced by a large number of input sequences, as well as conserve computational resources and time, ultimately making the model more efficient and effective.

By eliminating these tokens that are considered less important, we are able to streamline the model and enhance its overall performance [4, 13]. For instance, Ye *et al.* [12] introduced TR-BERT, in which they utilized a policy network driven by reinforcement learning to select the desired tokens and skip the unnecessary ones. However, a major limitation of this approach is that the performance of the token masking significantly declines when conducting experiments without pre-trained models. The work presented in [14] introduced a

novel method for reducing the size of input sequences by pruning tokens with low attention probabilities. This technique not only enables the reduction of the number of heads, but also allows for the simultaneous implementation of quantization. As such, SpAtten represents a significant advance in the field of input sequence optimization.

In generally speaking, the complexity and performance of transformer not only depend on the length of input sequence, but also are closely related to the dimension of embedding, the number of multiple attention mechanisms and etc. [4]. Motivated by previous research, we aim to optimize the performance of the transformer model by addressing three key components: the number of input tokens, the attended patterns (e.g., the low-rank based attention), and the hyper-parameters within the transformer layer. To successfully accomplish this task, it is necessary to identify an appropriate set of hyper-parameters that effectively balance accuracy and complexity.

However, optimizing the architecture of transformer models is a complex and non-trivial task [15], as traditional techniques such as hand-crafted design or brute force are often time-consuming and may not yield optimal results in terms of both complexity and accuracy. In order to effectively address this challenge, we propose the use of the Multi-objective Evolutionary Algorithm (MoEA) [16], a powerful optimization method with the ability to handle non-convex problems and insensitivity to local minima. By utilizing MoEA, we can achieve the design of highly efficient, compact transformer models with impressive levels of accuracy, while minimizing the time and resources required for the inference in the practical scenarios.

To solve the above problems, we propose "**E**volving **M**asked **L**ow-Rank **T**ransformer for Long Text Understanding", called EMLT, which is under the framework of Multi-objective evolutionary paradigm, and jointly optimized with two novel developed modules, the low-rank attention (LoRA) module and dynamic mask (DM) module. The LoRA model has been designed to consider both local sparse patterns and a projection for lower-resolution queries and keys, in order to reduce computational resources. In comparison to previous methods,

the developed module has linear complexity due to the sliding window, while also incorporating global context information through the parameterization of the projection matrix. This allows each sequence to have a unique projection matrix that preserves representative differences among different sentences. To further increase efficiency and remove unimportant tokens from the input sequence, the DM module has been designed to prune input tokens before each transformer layer by predicting their scores. The chosen unimportant tokens are removed before being input into the LoRA module, reducing computational complexity while still maintaining a good level of accuracy as only unimportant tokens with minimal impact on final performance are removed. In order to automate the optimization process and obtain a better model architecture with the configured settings of the DM and LoRA modules, a novel encoding space has been proposed for evolution. This encoding space includes information on embedding dimensionality, the number of heads in attention, and other key factors in the attached DM and LoRA modules, such as the pruning ratio of tokens. This design allows the MoEA to output a desirable model architecture configuration that considers both accuracy and computational complexity. Additionally, the overall MoEA-based model design makes it accessible for end users without expert knowledge. We have demonstrated the effectiveness of EMLT through several experiments on the Long Range Arena benchmark database, which is a benchmark dataset for evaluating the design of efficient transformers without the need for pretraining. Long-Range Arena (LRA), contains three specific datasets: ListOps, Text and Retrieval, which have token lengths of 2K, 4K, and 4K respectively. These three sub datasets clearly fall within the category of long text in language processing. To further reinforce our findings, we have included the original IMDb dataset, which belongs to the category of long text. The proposed method achieves significantly better efficiency compared to the state-of-the-art method, and also outperforms existing efficient transformer methods with notable margins.

The contributions of this work are:

1. We propose Evolving Masked Low-Rank Transformer (EMLT), a novel Multi-objective Evolutionary Algorithm-based method by jointing optimizing the regular hyper-parameters, settings of the low-rank attention pattern and input token pruning. To the best of our knowledge, this is the first work to evolve the efficient transformer on the long text processing tasks.
2. Our proposed module, LoRA, combines a dynamic low-rank projection-based attention model with both a global attention sub-module and a local attention sub-module, enabling it to capture fine-grained correlations. Additionally, LoRA effectively resolves the scale mismatch issue between the embeddings of the global and local attention sub-modules. Notably, LoRA improves efficiency of sparse patterns, offering linear time and memory complexity.
3. Our proposed DM module is designed to reduce the number of input tokens by pruning less important ones, in which way the computation complexity is reduced considerably. We introduce a prediction module that estimates the importance score of each token based on current features. To optimize the prediction module in an end-to-end manner, DM leverages an attention masking strategy that differentially prunes tokens by blocking their interactions with other tokens.
4. In our experimental evaluation on the long text benchmarks, we demonstrate significant improvements over other state-of-the-art methods in terms of computational complexity, while maintaining high accuracy levels with minimal or no loss. Furthermore, when compared to the original attention mechanism, our proposed approach achieves remarkable acceleration, up to 120 times faster.

2. Background Related Work

2.1. Transformer

Transformer revolutions have recently hit the field of artificial intelligence [2] and can achieve better performance with large-scale pre-training. Transformer was originally proposed as a sequence-to-sequence [17] model for machine translation. Later works show that Transformer-based pre-trained models can achieve state-of the-art performances on various tasks. As a consequence, Transformer has become the go-to architecture in NLP tasks with a set of multiple sequential attention layers [6]. It surpassed the earlier approaches by such a wide margin that all the recent cutting edge models seem to rely on these Transformer-based architectures [18]. In the last five years, hundreds of papers and language models inspired by Transformers were published, the bestknown being the auto-encoder architecture like BERT [19], and post-BERT models, such as RoBERTa [20], AlBERT [21], DistilBERT [22], and ERNIE [23], as well as the auto-regressive models, such as GPT, GPT-2, GPT-3 and XLNET [24]. The models relying on this Transformer architecture have fully outperformed the previous state-of-the-art networks.

2.2. Efficient Transformers

In recent year ultra-large-scale computation cost of attention quickly becomes a bottleneck of long sequences processing and the efficient transformer has developed in a variety of directions,including low rank, kernels, recurrence, hierarchical mechanism, sparse patterns and approximating attention [4].There is a wide variety of recently proposed method to make Transformers efficient for large sequences. The purpose is to improve the general efficiency of Transformers, including but not limited to tackling the quadratic complexity issue of the self-attention mechanism or reducing the computation costs by means such as pooling and/or sparsity. We also briefly discuss general improvements and other efficiency improvements such as parameter sharing [4]. A variety of patterns has been explored in the past work, the key idea of sparse patterns

is to make the attention operation sparse which is divided into content-based sparsify, including Reformer [25], Sinkhorn [26], and Routing Transformer [27], and pre-specified patterns, including Sparse Transformer [8], ETC [28], GMAT [29], Longformer [30] and BigBird [10].

The early research in this field primarily focused on block or local patterns, such as those used in the Image Transformer [31], Compressed Attention [32], Blockwise Transformer [9], and Sparse Transformer [8]. The idea of dividing various fixed patterns was first introduced in the work of Child et al. [8] and CCNet [33]. At around the same time, we see the emergence of model-memory-based approaches, including the inducing point method in the Set Transformer [34] and global nodes [35] in the Star Transformer [35]. More recent models have employed learnable sparsity patterns, such as the Reformer [25] and Routing Transformer [27]. These models both aim to cluster or bucket tokens before performing attention, but the Reformer uses a hashing function while the Routing Transformer uses online k-means for cluster assignment.

In addition, the Sinkhorn Transformer [26] employs the idea of sorting, but at the block level. These three models follow a similar approach of rearranging sequences for efficient computation of attention scores. Later, we see several extensions based on the Sparse Transformer paradigm, including the ETC [28] and Longformer models [30]. These models incorporate the concept of a global model memory, similar to the inducing point method in the Set Transformer or the global model memory of the Star Transformer. The Longformer model also introduced the use of dilated windows as a modification to strides.

The most recent trend in Transformer models is the use of low-rank approximation or kernel methods, such as the Low-Rank Transformer [36], Linformer [37], Performer [38], and Linear Transformers [39]. On the other hand, models like the Poolingformer explicitly construct a two-level attention mechanism using techniques reminiscent of memory-based approaches and local attention. Scatter Brain [40] is a recent work that attempts to unite sparse (fixed pattern) attention with low-rank attention. Luna [41] also proposes a two-stage attention mechanism. However, it is currently unclear whether these low-rank or kernel-

based models are superior to the learnable pattern (LP) or model memory-based efficient Transformer models, due to the state of evaluation and the high number of competing research efforts. It is worth noting that the recurrent-based models (Transformer-XL and Compressive Transformers) seem to operate in a different manner and are not as directly comparable to the other models. There has also been a recent increase in the popularity of sparse models [42, 43], which have been successfully applied to attention modules [44] and have demonstrated strong performance in recent months. There have been several efforts to improve the inference speed of Transformer models. For example, TinyBERT [45] proposes a distillation method to accelerate the inference of Transformer models. The Star-Transformer [35] reduces the quadratic space and time complexity to linear by replacing the fully connected structure with a star-shaped topology. It would be more informative if these studies had considered dynamic masking.

The demand for efficient transformer models is pressing, particularly when it comes to deploying vision transformers on devices with limited resources. Various approaches have been proposed to reduce complexity and enhance efficiency in vision transformers.

One such approach is Swin-Transformer [46], which achieves linear complexity by confining self-attention computation to local windows within the vision transformer. HaloNet [47], on the other hand, applies local attention to blocked images, resulting in quadratic complexity relative to the block size. Perceiver [48] addresses the quadratic complexity bottleneck by replacing self-attention on data with cross-attention between data and latent arrays. ViL (Vision Longformer) [49], a concurrent work, adapts Longformer [30] for vision tasks and achieves linear complexity. It enhances local window attention with task-specific global tokens, although these tokens are not suitable for image synthesis. The long-short transformer [5] combines local window attention and global dynamic projection attention, reducing the quadratic cost to linear cost, applicable to both encoding and decoding tasks. Liu et al. [50] employ a transformer encoder structure for efficient context feature extraction. To address attention head redundancy, importance scores are defined in [51] to estimate the influence of each

head on the final output, allowing for the removal of unimportant heads. Su et al. [52] conduct a search for patch size, linear projection dimensions, and attention module head numbers to obtain an efficient vision transformer.

2.3. Token Pruning in Transformer Models

One common approach to improving the inference speed of pre-trained language models (PLMs) is to reduce the number of layers, such as through knowledge distillation models like DistillBERT [22] and Patient Knowledge Distillation [53], and adaptive inference models like DeeBERT [54] and FastBERT [55]. However, this layer-wise pruning can significantly reduce the model's computational complexity but also decrease its ability to perform complex reasoning tasks. Recently, there has been an increasing focus on pruning input sentences to Transformers, rather than model parameters. This is referred to as token pruning, where less important tokens are progressively removed during inference. PoWER-BERT [56] was one of the first works to propose directly learning token pruning configurations. LAT [57] extended this idea by introducing LengthDrop, a method that trains a model with different token pruning configurations and then uses evolutionary search to find the best configuration. This has the advantage of not needing to retrain the model for different pruning ratios. While these methods have demonstrated significant computational reduction on various NLP downstream tasks, they fix a single token pruning configuration for the entire dataset. This means that all input sequences are pruned to the same length, regardless of their original length. However, input sequence lengths can vary greatly within a task, and using a fixed pruning configuration can either under-prune shorter sequences in order to retain enough tokens for processing longer sequences, or over-prune longer sequences to remove enough tokens to efficiently process shorter sequences. Additionally, a fixed pruning configuration lacks robustness against input sequence length variations, such as when the input sentences at inference time are longer than those in the training dataset [58].

In contrast, SpAtten [14] tries to address this issue by assigning a pruning

configuration proportional to the input sequence length. While this allows for more tokens to be pruned from longer sequences and fewer tokens from shorter ones, it is not adaptive to individual input sequences as it assigns the same configuration to all sequences with the same length regardless of their contents. In addition, the pruning configurations are determined heuristically, which may not result in the optimal solution. Recently, TR-BERT [12] proposed using a policy network to learn adaptive pruning configurations for each input sequence. However, the authors note that the problem has a large search space which can be difficult for Reinforcement Learning to solve. To mitigate this, the method uses heuristics involving imitation learning and sampling of action sequences, which significantly increases the cost of training.

2.4. Evolutionary Neural Architecture Search

Evolutionary algorithms are a class of optimization algorithms that are inspired by the evolutionary processes of living organisms. They typically involve operations such as gene encoding, population initialization, crossover mutation, and retention mechanisms. Compared to traditional optimization methods like calculus-based and exhaustive methods, evolutionary computation is a robust and widely applicable method. It has the advantages of self-organization, self-adaptation, and self-learning, and can effectively solve complex problems that are difficult to solve with traditional optimization algorithms, such as NP-hard optimization problems, without being limited by the nature of the problem [59, 60]. GeNet [61] is a neural architecture search method that uses an evolutionary computation approach and divides the architecture being searched into multiple stages. AmoebaNet [62] is another NAS method that aims to reduce the search space by using a predefined search space [63] and achieved a new state-of-the-art performance on the ImageNet [64] dataset, but with high computational complexity. EffPNet is a NAS method based on particle swarm optimization that searches for DenseNet-style transferable blocks and uses an SVM-based binary classifier as a surrogate model to reduce search cost [65]. These NAS methods often have fixed macro-architectures, leading to the de-

velopment of a flexible two-level PSO algorithm that can simultaneously evolve CNN architectures with both macro-level and micro-level components efficiently.

3. The EMLT Algorithm

3.1. Overall Framework

The proposed EMLT is a multi-objective architecture search method based on the genetic algorithm [15], and it is designed for the optimization of the compact architectures of the dynamic efficient transformers in natural language processing. After the searching processing, the developed EMLT algorithm will obtain a thinner dynamic and low-rank attention model while maintaining the accuracy as much as possible. This is achieved by approximating the computational complexity versus the final accuracy of the Pareto optimal on the specific downstream tasks.

The developed EMLT consists of four major processes, namely Population Initialization, Evaluation, Mating Pool Selection, Genetic Operators and Environmental Selection. To provide a clearer overview of the developed EMLT, we present the overall framework in Figure 1, in which the connections between the major processes are indicated by black arrows:

1) *Initialization*: the proposed EMLT initializes a population of candidate architectures from the proposed encoding space. This is an important step because the quality and diversity of the initial population can greatly impact the overall performance of the search for the final efficient transformer models. The details of the designing of the encoding space will be elaborated in the Section 3.4.

2) *Evaluation*: In the next step, the algorithm evaluates each candidate architecture, which encompasses various parameter configurations, by considering two objectives: model complexity and validation accuracy. These objective functions are specifically designed to assess the performance of the architecture across different tasks and domains. It is important to note that the evaluated

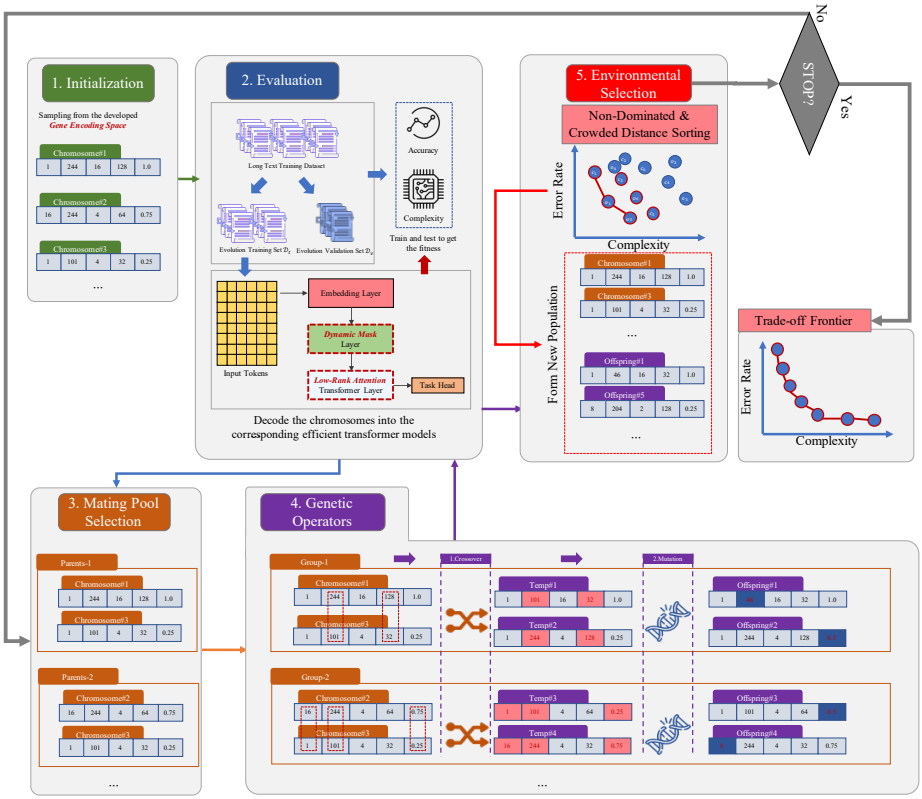

Figure 1: The overall flowchart of the EMLT. For ease of understanding, we draw several chromosomes and visualize the whole procedure in the first round. 1) Initialization: The algorithm will first sample individuals from the encoding space. 2) Evaluation: The initialized population will then be evaluated one by one, trained on the target long text dataset. The accuracy on the validation set and the FLOPs will be obtained to form the fitness of the initial population. 3) Mating Pool Selection: This process elects the parents of the offspring. The binary tournament selection mechanism is used to generate the parent pairs. 4) Genetic Operators: This process performs crossover and mutation on the chosen parents and generates new individuals. After the offspring are obtained, they will be evaluated to compute their fitness. 5) Environmental selection: By combining the old population and the new individuals, the non-dominated and crowded distance sorting will be used to compute the ranks of each individual in the new population set, and the individuals for the next round will be chosen based on their ranks or crowded distance. Once the algorithm satisfies the stop criterion or matches the maximal iterations, the individuals representing the transformer models in the final Pareto front will be returned to the end users for the further analyses or applications.

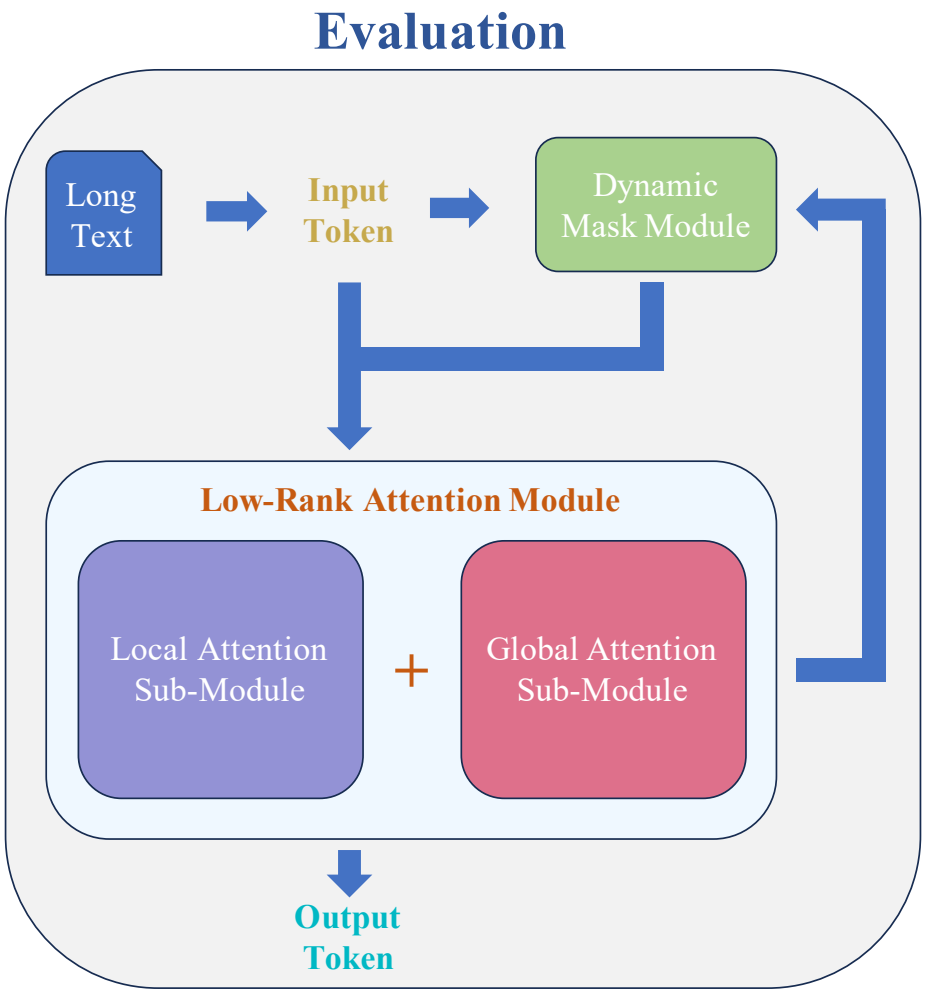

Figure 2: In the evaluation phase of the evolutionary algorithm, the long text undergoes word embedding to convert it into input tokens. These tokens are then fed into the dynamic mask module to remove insignificant ones. Following that, they are inputted into the low-rank attention module for additional processing. This entails extracting features from the long text using both the local attention sub-module and the global sub-module, resulting in the production of output tokens.

model is a dynamic masking transformer (Section 3.3) with a developed low-rank attention module (Section 3.2). This crucial step enables the algorithm to identify the most promising individuals in the population and prioritize them for further exploration. The evaluation process will be discussed in Section 3.5. To enhance the depiction of the association between EA and individual modules, we present the model's data flow during the evaluation in Figure 2. Arrows are utilized to indicate the connections between each sub-module.

3) *Mating Pool Selection & Genetic Operators*: After evaluation, the algorithm selects a subset of the most promising individuals from the population to form a mating pool. These individuals are used as parents to generate new offspring through genetic operators such as mutation and crossover. The genetic operators introduce diversity and novelty into the population and are crucial for exploring the search space effectively. In this work, we adopt a specific set of evolutionary operators, including the widely-used binary tournament selection, point crossover, and polynomial mutation [15]. These operators have been demonstrated to be effective in exploring the search space and generating diverse and high-performing candidate solutions.

4) *Environmental Selection*: The offspring generated from the mating pool are then evaluated, and the fittest individuals are added to the population. However, to maintain diversity and prevent premature convergence, the algorithm also removes less fit individuals through environmental selection. This step ensures that the search continues to explore the search space effectively and avoids getting stuck in local optima. To effectively sort candidate solutions based on their dual objectives, this work employs the Non-dominated and crowded distance sorting method [66]. This method enables the selection of individuals based on multiple objectives, ensuring that only the most promising solutions are chosen for further optimization.

The algorithm continues iterating through these steps, generating and evaluating new offspring, selecting the fittest individuals for the next generation, and removing less fit individuals until a satisfactory set of architectures are evolved. These architectures can then be developed into efficient transformer models that

exhibit strong performance across a range of tasks and domains. Through this evolutionary procedure, the algorithm is able to search for optimal architectures of the dynamic masking transformers for the efficient long text processing, balances multiple objectives and exhibits strong performance.

In the following subsections, we will introduce the low-rank attention module, dynamic masking module, and the gene encoding space representing the importance factors of these modules. Additionally, we will discuss the evaluation process, which involves assessing each individual in the population.

3.2. Low-rank Attention

Existing transformers in processing long documents have been witnessed a large computational burden. The NLP community considers this performance gap is due to the attention mechanism [4], where the complexity of the vanilla attention module in the wide-used methods is proportional to the square of the number of input tokens. Specifically, the attention mechanism, as a semantic modeling method, needs to compute the pairwise similarities among the tokens. i.e., for the given queries $Q \in \mathbb{R}^{n \times d_m}$ and the key-value pairs $K, V \in \mathbb{R}^{n \times d_m}$, multi-head self-attention (MHA) is commonly employed to obtain the attended values [2]. The MHA function is expressed by:

$$\mathrm{MHA}(Q, K, V) = \mathrm{Cat}(H_1, H_2, ..., H_h)W^O \in \mathbb{R}^{n \times d_m}, \tag{1}$$

where each head H_i will project the original tokens to different sub-spaces, and conduct the scale dot-production attention in the corresponding sub-space:

$$\begin{aligned} H_i &= \mathrm{Attn}(QW_i^Q, KW_i^K, VW_i^V) \\ &= \mathrm{Softmax}(\frac{QW_i^Q \left(KW_i^K\right)^\mathsf{T}}{\sqrt{d_k}})VW_i^V \\ &= A_i VW_i^V \in \mathbb{R}^{n \times d_k}. \end{aligned} \tag{2}$$

As can be seen from the above equation, the attention matrix A_i is an $n \times n$ matrix. Therefore, this processing will consume much more memory space to continue the computational flow when inputting a very long article (e.g., the

length of the sentence larger than 2048). To tackle this problem, we proposed a novel low-rank attention mechanism, which reduces the complexity through a local sparse attention pattern, and preserves the global correlations by a global attention sub-module at the same time. A simple way is to use a sparse pattern of attention, such as local sliding window when computing the attention map, which has been adopted in [30, 10] due to its simplicity and effectiveness. It assumes most tokens are only correlated with its own neighbored tokens, which usually makes sense in most scenarios. But when this local window pattern is employed, the global indecencies of the input sequences are neglected, which would lead to a loss of global semantic information, especially for the long-range dependencies. To compensate that, the global semantic information among the original sentences should be considered. Therefore, a local attention sub-module and a global attention sub-module is developed for the proposed EMLT, in order to capture the rich semantic information among the context while making the complexity of the model be linear with the length of the input sequence. In the following paragraphs, we will describe the Local Attention Sub-module and the Global Attention Sub-module in more details.

3.2.1. The Local Attention Sub-module

We use a sliding window strategy to compute the local attention maps. For the queries Q and keys K, their pairwise similarity is an $n \times n$ matrix. The sliding window is performed on the matrix and only considers w neighbor tokens when a specific token is selected. i.e., the window size w represents the fixed range of span when calculating the similarity of q and k. Hence, each token in Q or K will be attended to $2w$ tokens. This attention process can be regarded as n new sets of key-values pairs K^L, V^L to be attended with each q in the original queries Q, respectively. For a query q_t at the t-th position, each row of

the selected key-value pairs $K_t^L \in R^{2w \times d_m}, V_t^L \in R^{2w \times d_m}$ can be obtained by:

$$k_{t,i}^L = \begin{cases} k_{idx_{t,i}}, 1 \leq idx_{t,i} \leq n \\ \mathbf{0} \quad\quad , \text{else} \end{cases}, \quad \forall i \in [1, \cdots, 2w], \tag{3}$$
$$v_{t,i}^L = \begin{cases} v_{idx_{t,i}}, 1 \leq idx_{t,i} \leq n \\ \mathbf{0} \quad\quad , \text{else} \end{cases},$$

where $k_{t,i}^L, v_{t,i}^L \in \mathbb{R}^{d_m}$, and the zero vector $\mathbf{0} \in \mathbb{R}^{d_m}$ means zero padding when accessing is out of the boundary of the matrix. The corresponding index variable idx is defined by:

$$idx_{t,i} = i + w\lfloor \frac{t-1}{w} \rfloor - \lceil \frac{w}{2} \rceil \tag{4}$$

Next, the revised local attention at q_t can be rewritten as:

$$\begin{aligned} \text{LocAttn}(Q, K, V, t) &= \text{Softmax}(\frac{q_t W_i^Q (K_t^L W_i^K)^\intercal}{\sqrt{d_k}}) V_t^L W_i^V \\ &= A_t^L (V_t^L W_i^V) \in \mathbb{R}^{d_k} \end{aligned}. \tag{5}$$

The complexity of this sparse local attention for the whole input sequence is $O(nw)$ $(n \gg w)$, which is linear with the length of the input tokens. In Fig. 3, we show how to compute this attention process, and we also clearly exhibit the difference between the original attention and the developed local attention module, which requires less computational resources.

3.2.2. The Global Attention Sub-module

The original attention can be approximately decomposed by the low-rank matrices [5], thus the complexity of computing the attention with those low-rank matrices can be reduced while accessing all the tokens in the sequence. To achieve the low-rank decomposition goal, researchers tried to learn a function $f^P : \mathbb{R}^n \to \mathbb{R}^r$ to reduce the length of the input tokens by introducing a new projection matrix $P^r \in \mathbb{R}^{n \times r}$ with a lower rank r $(n \gg r)$ [37, 5]. Inspired by the work presented in [5], we parameterize the projection matrix P since that of the matrix with the same entries for all the input sequences may result in the positional variations of semantics.

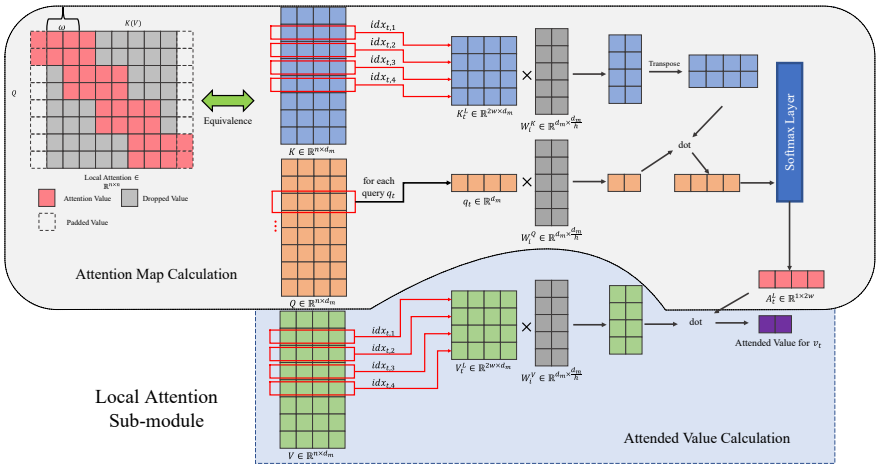

Figure 3: Local Attention Sub-module. This module is a type of sparse attention pattern, which employs the sliding windows paradigm used in many recent works. By rearranging the input K, V, we will obtain a fast method for the local attention mechanism.

The low-rank matrix is determined by the input set of keys K, hence this dimensionality reduction process is dynamic and self-adaptive for various input sequences. To be more specific, the estimated projection matrix P_i^A of the i-th head is determined by:

$$P_i^r = \text{Softmax}(KW_i^P), \tag{6}$$

where the matrix $W_i^P \in \mathbb{R}^{d_m \times r}$ is a differentiable learnable variable. Sequentially, the projected key-value pairs K_i^P, V_i^P in the i-th head can be computed by directly performing the linear transformations:

$$K_i^P = (P_i^r)^\mathsf{T} K W_i^K, V_i^P = (P_i^r)^\mathsf{T} V W_i^V. \tag{7}$$

Thus, the Eqn. (2) can be replaced by the revised global attention $H_i^G \in \mathbb{R}^{n \times d_k}$:

$$H_i^G = \text{Softmax}(\frac{QW_i^Q (K_i^P)^\mathsf{T}}{\sqrt{d_k}}) V_i^P = A_i^G V_i^P, \tag{8}$$

where the new attention matrix A_i^G is an $n \times r$ matrix, and the transformed set of values V_i^P is a $r \times d_k$ matrix. Therefore, the computational complexity of calculating Eqn. (8) is $O(n \times r \times d_k)$, which is also linear with the length of input tokens. Contrasted with the local attention matrix A_t^L in Eqn. (5), the

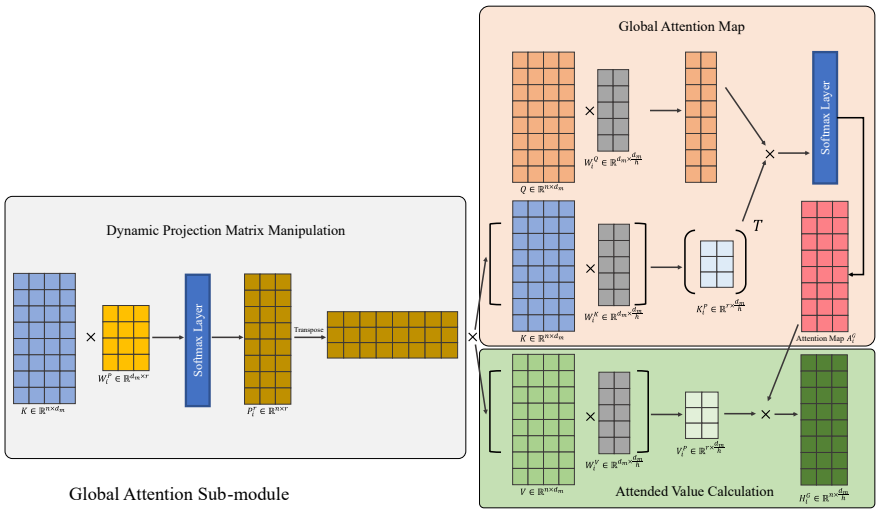

Figure 4: Global Attention Sub-module. This module parameterizes the projection matrix P in such a way that each input sequence has a unique matrix for subsequent manipulation. By controlling the rank of the projection matrix, we can significantly reduce the computational burden instead of computing the pairwise similarities of all the n tokens. At the same time, we are still able to access global semantic information because no tokens are excluded.

former one fuses all the information in the sequence, so the global correlations would be preserved. To better show this global attention calculation, we show it in the Fig. 4.

The last step is to combine the local attention with global attention to obtain the final output of the developed low-rank attention mechanism. During the aggregation procedure, we should consider both of the semantic information at different scales. A straightforward way of aggregation is to add the local attention result and that of the global attention. Although this is the simplest way to realize our goal, but this strategy will lead to an unstable training of the transformer models due to the different scales. To complete the final computation, let the new key-value pairs K_t^F, V_t^F be the fused variables:

$$\begin{aligned} K_{i,t}^F &= \mathrm{Cat}(\mathrm{Norm}(K_t^L W_i^K), \mathrm{Norm}(K_i^P)) \quad \in \mathbb{R}^{(2w+r)\times d_k} \\ V_{i,t}^F &= \mathrm{Cat}(\mathrm{Norm}(V_t^L W_i^V), \mathrm{Norm}(V_i^P)) \quad \in \mathbb{R}^{(2w+r)\times d_k} \end{aligned}, \tag{9}$$

where the "Cat" operator will concatenate the dual input variables along the

first axis, and the "Norm" symbol denotes a normalization layer. This added layer can alleviate the problem of the gradient explosion and make the distribution of the output values more stable during training.

In the end, the output attention at the i-th head and the q_t can be derived by:

$$H_{i,t}^F = \text{Softmax}(\frac{q_t W_i^Q (K_{i,t}^F)^\intercal}{\sqrt{d_k}}) V_{i,t}^F \in \mathbb{R}^{d_k}. \tag{10}$$

At last, we can collect the final attended features by:

$$H_i^F = \{H_{i,t}^F\}_{t=1}^n \in \mathbb{R}^{n \times d_k}, \tag{11}$$

which contains the local and global semantic information among the input tokens. As can be seen from the above equation, the computational complexity is $O((2w + r) \times d_k \times n)$. Please note that w, r, d_k are always smaller than the length of sequence, and can be regarded as the small constant. Thus, the final computational complexity of the developed low-rank attention is proportional to n, and is linear with the number of input tokens. For the sake of understanding, we draw a flowchart to better describe how to compute the developed low-rank attention, see Fig. 5 for the details.

3.3. Dynamic Masking

The last low-rank attention module aims to address the computational overhead caused by the original attention mechanism when the token number is fixed. On the other hand, the dynamic masking module operates directly on the number of tokens to tackle the resource-intensive requirements from a different perspective. To better reduce the complexity induced by the larger number of the input tokens n, we developed the dynamic mask module collaborated with the developed low-rank attention mechanism to strongly alleviate the performance bottleneck.

The core functionality of the dynamic mask module is a mapping $f^d : \mathbb{R}^{n \times d_m} \to \mathbb{R}^{n \times 2}$. However, this mapping function is not easy to learn especially at the early stage, since the output of different layers may have diverse scales and distributions. To mitigate such problem, we firstly apply a layer normalization

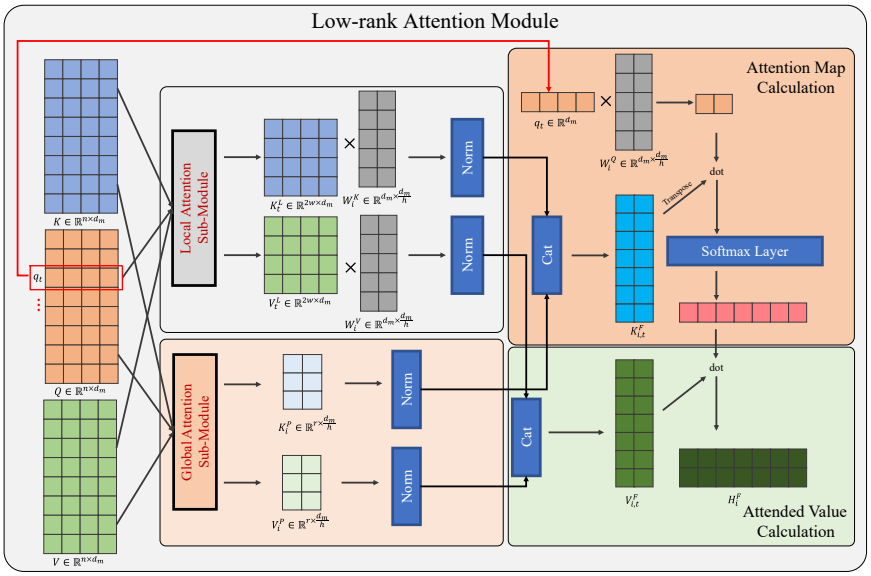

Figure 5: The overall flowchart of the low-rank attention module. The results of the dual revised attention modules will be fused to geneate the final attended values for the following computation.

on the input sequence to match the scales. And a linear transformer layer with the GELU activation function is followed after the normalization layer. This step can be viewed as a initialized pre-processing for the next feature manipulations. Specifically, for a given sequence $X \in \mathbb{R}^{n \times d_m}$, the normalized features X^I can be obtained by:

$$X^I = \text{GELU}(\text{Norm}(X)W^I + b^I), \tag{12}$$

where $W^I \in \mathbb{R}^{d_m \times d_m}$ and $b^I \in \mathbb{R}^{d_m}$ are the weights and bias, respectively. The linear transformation wrapper is an adaptive layer for the following token decision.

The importance of a token can be judged locally and globally [13]. From the point of local view, some tokens are naturally more frequent and should be allocated with a higher score for judgement, such as the words "the", "be" and etc. On the other hand, computing the importance of the word should also considers the context semantic information, which is essential for understanding

the true representation of a sentence. Therefore, we will compute the representative features locally and globally. For the local semantic features, we obtain it via collecting the features of the first half along the channel axis:

$$X^L = X^I_{:,1:\lfloor \frac{d_m}{2} \rfloor} \in \mathbb{R}^{n \times \lfloor \frac{d_m}{2} \rfloor}, \tag{13}$$

where the footnote "{1:n,:}" denotes the slicing operator for a tensor. For the global information, the features of the rest channels are used:

$$X^B = X^I_{1:n,\lfloor \frac{d_m}{2} \rfloor + 1:d_m} \in \mathbb{R}^{n \times \lceil \frac{d_m}{2} \rceil}, \tag{14}$$

and the global semantic features can be integrated by combining with a sum-pooling operation:

$$X^G = \text{SumPool}(X^B) \in \mathbb{R}^{\lceil \frac{d_m}{2} \rceil}, \tag{15}$$

where the "SumPool" operator will summation the matrix along the first dimension. Then, the features with local and global semantic information can be obtained by:

$$X^{LG} = \text{Cat}(X^L, \text{Expand}(X^G)) \in \mathbb{R}^{n \times d_m}. \tag{16}$$

Please note that there is merely no extra computing or storing cost when splitting the input features X^I, and this makes the whole processing efficient. Next, the features X^{LG} will be used to get the prediction scores for the final token decision. To complete the token classification, we add a three layers of linear layers on the top, which can be expressed by:

$$\begin{aligned} &X^S_1 = \text{GELU}(X^{LG} W^S_1 + b^S_1), X^S_2 = \text{GELU}(X^S_1 W^S_2 + b^S_2), \\ &S = \text{Softmax}(X^S_2 W^S_3 + b^S_3), W^S_1 \in \mathbb{R}^{d_m \times \lfloor \frac{d_m}{2} \rfloor}, b^S_1 \in \mathbb{R}^{\lfloor \frac{d_m}{2} \rfloor}, \\ &W^S_2 \in \mathbb{R}^{\lfloor \frac{d_m}{2} \rfloor \times \lfloor \frac{d_m}{4} \rfloor}, b^S_2 \in \mathbb{R}^{\lfloor \frac{d_m}{4} \rfloor}, W^S_3 \in \mathbb{R}^{\lfloor \frac{d_m}{4} \rfloor \times 2}, b^S_3 \in \mathbb{R}^2, \end{aligned} \tag{17}$$

This procedure will gradually reduce the dimensionality of the features into a small space, which helps the final classification layer for better decision. Compared with a single-layer shallow network, such a design makes the network have more model capacity, and it is easier to select representative features for predicting the importance of tokens. On the other hand, compared with networks consisting by dozens layers, it has advantage in terms of efficiency. Finally, we can get the final prediction scores $S \in \mathbb{R}^{n \times 2}$.

3.3.1. Training Phase

Once the importance score of each token is obtained, we can leverage this score to decide which token should be removed or remained. However, this "selection" operation is non-differentiable. This indicates that the gradient information cannot be back-forwarded to the developed dynamic mask module, if we force to use the "selection" operation. To address this issue, we adopt the Gumbel softmax to sample the mask vector $m \in \mathbb{R}^n$ of the input token, which is fully differentiable and can be used to determine the kept tokens:

$$m^t = \text{GumbelSoftmax}(S) \in \{0, 1\}^n, \tag{18}$$

where sampling "0" means the corresponding token should be removed, and "1" corresponds to the remaining operation. Once we obtain the decision mask m^t for the input token of a specific transformer layer, we can use this mask vector to filter the original tokens that should be discarded and feed the remaining ones to the developed low-rank attention module.

Unfortunately, an practical problem will be raised for the batch-based training, if we remove the less importance tokens during training. This problem is attributed to introducing Gumbel softmax that each input sequence often has different number of tokens to be pruned after sampling, which breaks the intact structure of the original input tensor. To tackle this issue and maximize the parallel computing power of the hardware, we do not really prune or cut the desired tokens, but remove the influence of the irreverent tokens by revising the softmax function, when calculating the low-rank attention in Eqn. (10). The dynamic attended values can be computed by:

$$H_i^{FP} = \frac{\text{Exp}(\frac{q_t W_i^Q (K_{i,t}^F)^\intercal}{\sqrt{d_k}}) \odot m}{\text{Sum}(\text{Exp}(\frac{q_t W_i^Q (K_{i,t}^F)^\intercal}{\sqrt{d_k}}) \odot m)} V_{i,t}^F \in \mathbb{R}^{d_k}, \tag{19}$$

where the output features affected by the dynamic masking have the same dimensional shape with the regular one during the training phase. Thus, in the training process, this mechanism does not actually play a role in further reducing storing consumption.

3.3.2. Testing Phase

During the testing phase, we can directly employ the proposed dynamic mask strategy by removing the less-important tokens.

Assume the pruning ratio for the j-th reduction prediction branch is ρ_j. We firstly sort the prediction scores along with the last dimension, and get the the indices of values from high to low, and select first $(1-\rho_j)n$ indices from the set. This post-procession step in the reduction module can be rewritten as:

$$m^v = \text{Argsort}(\text{Argsort}(S)) > \rho_j n, \tag{20}$$

where m^v is a binary mask for the current input tokens X, and the set of pruned tokens X' can be obtained by:

$$X'_i = \begin{cases} X_i, m_i = 1 \\ \\ \emptyset, m_i = 0 \end{cases}, \quad \forall i \in [1..n]. \tag{21}$$

After eliminating the less important tokens, the new set of tokens X' will be transmitted to the next layer. As can been seen, the number of the tokens have been greatly reduced when the following transformer layer process the pruned tokens compared with the original input. This will save a lot memory consumption and training time. Specifically, for the j-th transformer layer, it will reduce 50% computational complexity if setting ρ_j to be 0.5. With this dynamic strategy, we can double the computing performance on top of the previously proposed low-rank attention mechanism.

For the sake of description, an example of the dynamic mask module with two adjacent transformer layers is drawn in the Fig. 6.

3.4. The Gene Encoding Space

Gene encoding space or search space is essential for the evolution of the architecture search, and a suitable coding space would help to fast find the optimal solution [16]. Therefore, the gene encoding space for the transformer models is one of the essential components in the MOEA-based architecture search methods, and it should be carefully treated [67]. As for the dynamic

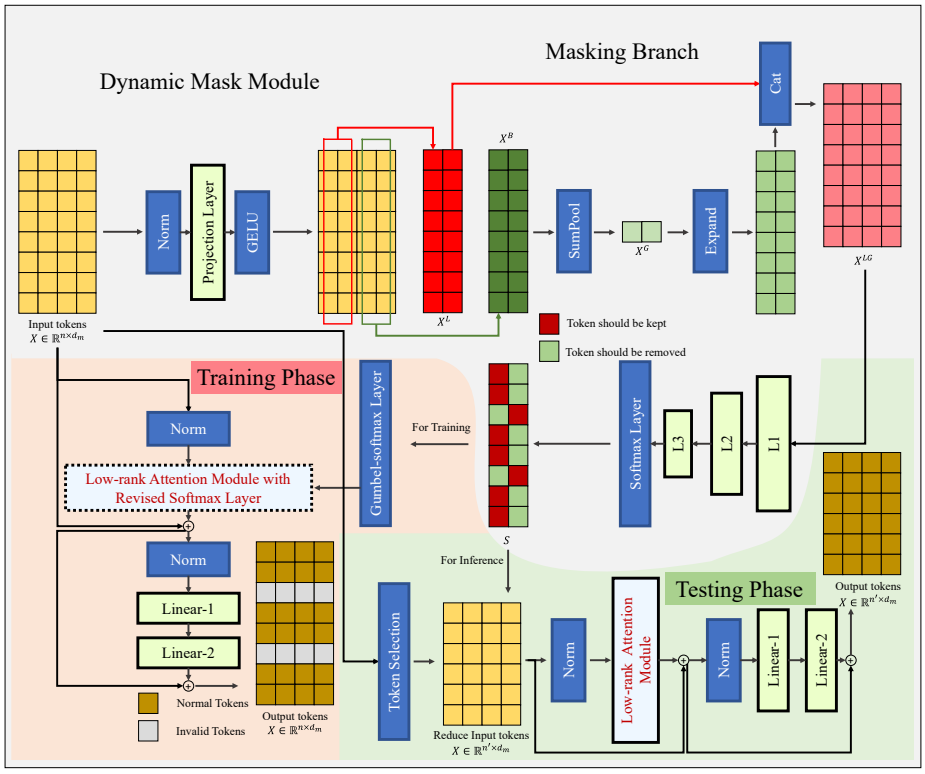

Figure 6: The dynamic mask module with the developed low-rank attention module. This figure shows how the dynamic mask module works in different phases. For the models during training, the tokens after the DM layer will keep the same length to implement the batch-based training. To alleviate the semantics shift between the remaining and dropping, a revised softmax layer will be created to replace the old one. For the testing phase, each input sequence through the DM module will be downsized.

masking transformers, one key parameter for the developed dynamic masking mechanism is the predefined token pruning ratio (or the remaining ratio) ρ. The token pruning ratio controls the complexity of input sequences, and larger pruning ratio will greatly reduce the computational burden of the sequential network forwarding. The other key factors which have great impact on the final performance are the hyper-parameters in the developed low-rank attention mechanism, i.e., the embedding size (or embedding dim) d_m [68], the window size w and the rank r. Theoretically, increasing the embedding size would promote the model capability to fit the original dataset, but more computational resources could be consumed. Besides, the number of the multi-head attention h will also contributes to the model complexity and final accuracy. Larger number of the heads will project the input token to more sub-spaces, and the whole computational workload increases.

To better investigate the potentially promising hyper-parameters of the compact transformer-based models with considerable accuracy, we proposed a novel gene encoding space for the developed EMLT algorithm. This encoding space contains the information of the dynamic masking in each layer and the hyper-parameters of the transformer models with the low-rank attention mechanism, which will be jointly optimized for better candidates of the fast and effective transformer models in processing the downstream tasks of long text.

The details of the encoded information in this work are summarized in Table 1. Taking a given i-th transformer layer as example, if the number of the input tokens $n_i = 2048$ and the encoded pruning ratio $\rho_i = 0.25$, then the number of the output tokens in this layer should be $2048 \times (1-0.75) = 1536$, which indicates that the model will gain at least 43.75% computational cost reduction under the original attention mechanism settings. However, the network might be unstable during training under such high pruning ratio. This is because some important tokens are wrongly judged in the early training stage, then those tokens will be mistakenly discarded and lead to an obvious precision drop. Thus, there must be a trade-off between the token pruning ratio and the inference speed. The embedding size d_m represents the embedding dimensionality of the embedding

Table 1: The encoding information in the proposed EMLT

Unit Type	Encoded Information	Parameter Choices
Token Pruning Ratio	Remaining ratio for the dynamic masking, large number represents this layer will keep more tokens, and vice verse	[0.25,0.5,0.75,1]
Window Size	The local window size in the proposed low-rank attention module	[1,2,4,8,16,32]
Rank of Attention	the rank of the low-rank attention	[1,2,3,$\cdots$,256]
Head Number	The number of heads for the Multi-Head Attention (MHA)	[2,4,8,16]
Embedding Dim	Embedding dimensionality in the transformer layer	[32,64,128]

layer at the bottom of the transformer models. For the sake of simplicity, we follow the traditions in the most transformer-based models that the number of neurons in the hidden layer is as twice as the d_m [2, 5].

To better show how the encoding space works, several examples of chromosomes sampling from the proposed encoding space are depicted in Fig. 7. From this figure, we can see the encoding strategy of the chromosomes is fixed-length, and the length of the chromosome is dependent on the number of the transformer layers. But, searching with brute force is often infeasible due to the limit of the computational resources and longer training time. e.g., more than one billion models would be evaluated for an eight-layer transformer model. Hence developing a heuristic method to optimize the architectures of the dynamic efficient transformer models is necessary under such a search space.

Each dynamic masking transformer-based model can be represented as a unique chromosome, and the reverse is also possible. To evaluate the chromosome and determine its performance, the information contained within it must first be decoded to construct a real deep learning model. Fig. 8 demonstrates this process using "Chromosome#2" from Fig. 7 as an example. The corresponding two-layer model is illustrated, clearly depicting how the encoded gene information is derived to construct the model. Decoding the chromosome is a crucial step in the proposed EMLT, allowing to evaluate the performance of the dynamic transformer model. It enables the EMLT to transform the genetic information into a real model that can be trained and evaluated. This process is necessary to determine the effectiveness of the model in achieving the desired task, and to make any necessary modifications to improve its performance. Fur-

Figure 7: Visualization of the example chromosomes sampling from the gene encoding space, where the target model is a two-layer transformer. Each unit is drawn by different colors, and plays different role in the transformer model.

thermore, it provides a means to analyze and compare different models and their corresponding chromosomes, aiding in the selection of the most promising candidate solutions.

3.5. Evaluation

The evaluation process is designed to obtain the fitness of individuals in the given population P_t, i.e., the precision and FLOPs of the corresponding individual. Each individual to be evaluated should be decoded from the chromosome into a real transformer-based model, as depicted in Fig. 8. Since all models are randomly-initialized, they should be first trained on the target training dataset $\mathcal{D}_t = \{X_i, Y_i\}_{i=1}^{n_t}$, and then tested on the validation set $\mathcal{D}_v = \{X_i, Y_i\}_{i=1}^{n_v}$, which is disjointed from the $\mathcal{D}_t$. Then, the accuracy on the validation set and the computational complexity are reported to the next evolution step, enabling subsequent steps to compare the pros and cons of any two individuals. Please note that the testing set is independent from the $\mathcal{D}_t$ and $\mathcal{D}_v$. This is to prevent from training on testing dataset.

To train a single dynamic masked transformer models, a regular term related

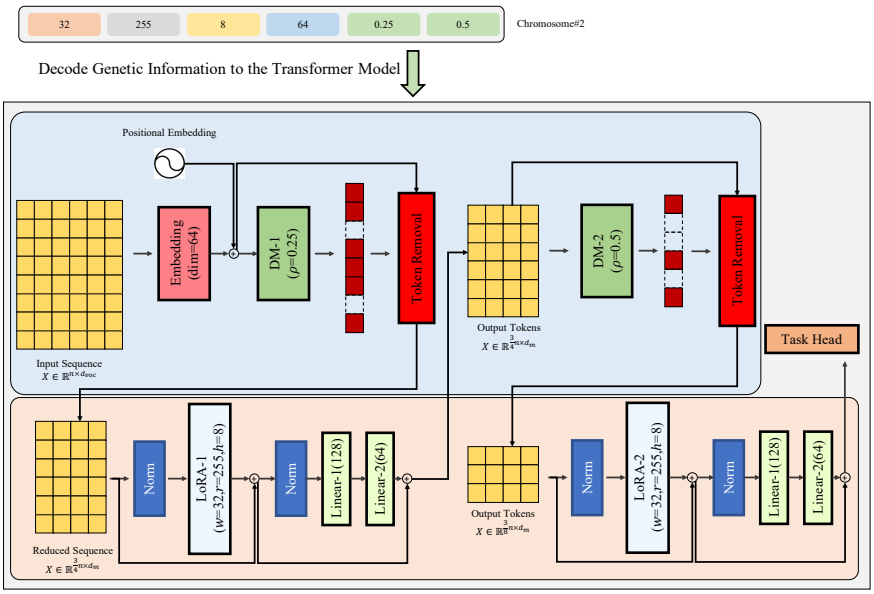

Figure 8: Example of decoding a particular chromosome into the corresponding dynamic masking transformer model (take the "Chromosome#2" for example). The light blue part (upper area) shows the function of dynamic mask module, which significant reduces the computational complexity especially when the number of input tokens is large. The light pink part (lower area) shows the corresponding relationship between the parameters in the two transformer layers and the genetic information in the chromosome. Please note that the dark red square denotes the tokens to be remained, while the empty ones with dashed lines represents the removed tokens.

to the dynamic mask branch will be added to the original loss. For the i-th data point and the j-th dynamic branch, their regular term is expressed by:

$$\mathcal{L}_{i,j}^{p} = (\frac{\sum_{k=1}^{n} m_{i,k}^{t}}{n} - \rho_j)^2, \tag{22}$$

which will encourage the pruning ratio to approximate the predefined ρ_j. We combine it with the classic cross-entropy loss via a coefficient α, and the final objective function can be derived by:

$$\mathcal{L} = -\frac{1}{n_t}\sum_{i=1}^{n_t}\sum_{k=1}^{c} y_{j,k}\log(y'_{j,k}) + \frac{\alpha}{n_t}\sum_{i=1}^{n_t}\sum_{j=1}^{L}\mathcal{L}_{i,j}^{p}, \tag{23}$$

3.6. Overall Algorithm

The overall algorithm of the proposed EMLT is depicted in Algo. 1. Firstly, the population P_0 at the beginning will be initialized by sampling from the proposed gene encoding space (line 1), and each individual in P_0 represents a set of the particular hyper-parameters for the dynamic masking transformer models. Immediately after, the initialized population P_0 will be trained and evaluated to obtain the fitness F_0 contains the information of the computational complexity and accuracy of the corresponding models. Next, the algorithm will choose the parent individuals C_t via the binary tournament to form the mating pool (line 4). C_t will be employed to generate the offspring through the genetic operators (line 5), and the new individuals contain the new configurations about the hyper-parameters of the low-rank attention and the token pruning ratios. Similar to the initial population P_0, the new generated individuals O_t should also be evaluated to gain the corresponding fitness. Sequentially, the environmental selection will be performed on the combined set of the evaluated offspring and the population of the last generation to form the current population P_t (line 7) according to their fitness. i.e., the non-dominated sorting and crowded distance sorting will be employed. The whole process will be terminated when reaching the stop condition, then output the finally evolved population P_T. After obtaining P_T, the configuration with the most desired computational demands can be selected.

Algorithm 1: Algorithm of EMLT.

Input: The number of generations T

Output: The candidate model configurations P_T

$P_0 \leftarrow$ Initialize the population from the proposed gene encoding space;
$F_0 \leftarrow$ Evaluate the population P_0;
for $t = 1$ **to** T **do**
    $C_t \leftarrow$ Select the parent solutions via the binary tournament selection;
    $O_t \leftarrow$ Generate the offspring from C_t with the genetic operators;
    $F_t \leftarrow$ Evaluate the new set of individuals O_t;
    $P_t \leftarrow$ Environmental selection from $P_{t-1} \cup O_t$;
end
return P_T

4. Experiments

4.1. Baseline Models

To demonstrate the effectiveness of the proposed EMLT method, we choose the recent SOTA efficient transformer models as the peer competitors. They are Sinkhorn Transformer [26], Reformer [25], Linformer [37], Performer[69] Nyströmformer [11] and Transformer-LS [5]. The transformer model with vanilla attention mechanism is also included. Then, we report the accuracy and FLOPs of each method on the target dataset. Lastly, the accuracy and the computational complexity are reported.

4.2. Datasets

Long Range Arena [7] dataset is a systematic and unified benchmark, which is released to specifically focus on evaluating the model performance under the long-text scenarios. We uses three sub-dataset in Long Range Arena and IMDb dataset to demonstrate the generality and effectiveness of our approach:

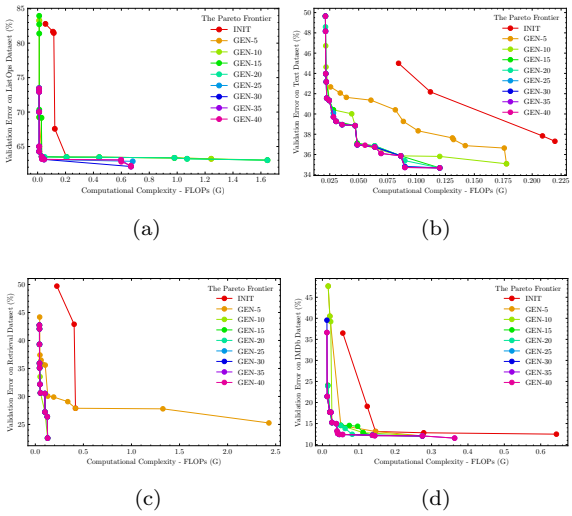

Figure 9: The trajectory of the evolutionary dynamic masking transformers with low-rank attention.

Table 2: Overall results of the proposed EMLT on the ListOps, Text and Retrieval dataset, respectively.

Methods	ListOps		Text		Retrieval	
	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)
Vanilla Attention	37.1	1.21	65.4	4.57	82.3	9.14
Reformer	36.4	0.27	64.9	0.58	78.6	1.15
Linformer	37.4	0.41	56.1	0.81	79.4	1.62
Performer	32.8	0.41	65.2	0.82	81.7	1.63
Nyströmformer	37.3	0.61	65.8	1.02	81.3	2.03
Transformer-LS	37.5	0.20	66.0	0.40	81.8	0.80
EMLT-ours *	37.3	0.01	65.2	0.09	79.5	0.13

Table 3: Overall results of the proposed EMLT on IMDb dataset.

Methods	IMDb	
	Accuracy	FLOPs (G)
Vanilla Attention	87.32	1.21
Reformer	87.83	0.41
Linformer	86.44	0.41
Performer	87.02	0.27
Nyströmformer	87.23	0.18
Transformer-LS	86.63	0.20
EMLT -ours *	**87.9**	**0.14**

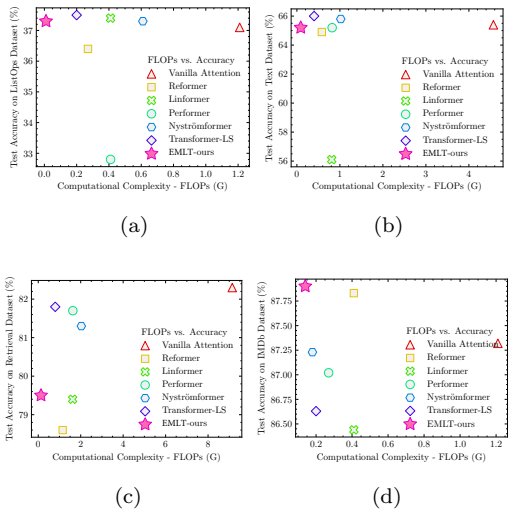

Figure 10: The FLOPs and accuracy of each method on the ListOps, Retrieval, Text and IMDb dataset, separately. The closer to the upper left corner, the better from the perspective of complexity and accuracy.

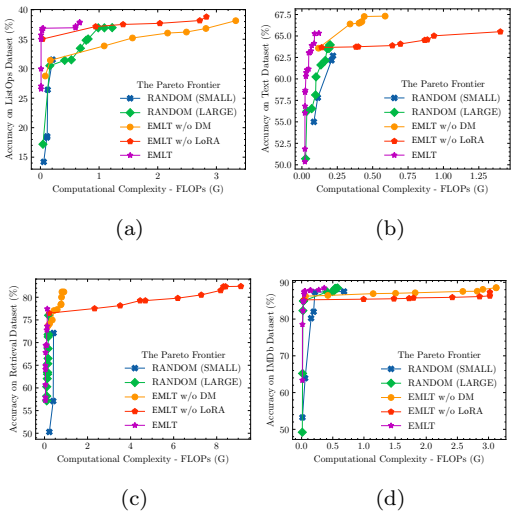

Figure 11: Pareto front of the ablation study for each dataset, where "RANDOM (SMALL)" denotes the random search method with 40 sampled data points, "RANDOM (LARGE)" is the random search method with 1600 sampled data points, "EMLT w/o DM" represents the EMLT method without the dynamic masking module, and the "EMLT w/o LoRA" means the EMLT method without the low-rank attention module.

ListOps: This dataset assesses a model's ability to parse hierarchical data by testing its capacity to process instances with fewer than 2k tokens.

Text: This dataset involves classifying movie reviews from IMDB as positive or negative based on sentiment. To make accurate predictions, the model must be able to understand long, unbroken sequences of characters at the component level and handle sequences up to 4k characters in length.

Retrieval: In this dataset, the model must determine whether two papers have a shared citation, which requires the ability to encode long sequences for similarity-based matching. Each byte-level document has a maximum sequence length of 4k and the model processes two documents at a time.

IMDb: In contrast to the character-level encoding used for Text in the Long Range Arena, we employ word-level encoding for the original IMDb data. This is due to the longer text length, which exceeds the typical sequence length of 512, reaching up to 2k. Thus, it falls under the category of long text processing.

Table 4: Quantitative results of the ablation experiments on the ListOps dataset.

Methods	Smallest Models		Middle Models		Largest Models	
	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)
RANDOM (SMALL)	14.2	0.06	18.5	0.12	31.5	0.207
RANDOM (LARGE)	17.1	0.04	34.8	0.78	37.0	1.22
EMLT w/o DM	28.7	0.08	35.2	1.57	38.1	3.33
EMLT w/o LoRA	35.0	0.04	37.5	1.41	38.8	2.83
EMLT	26.6	0.01	36.8	0.03	37.9	0.69

Table 5: Quantitative results of the ablation experiments on the Text dataset.

Methods	Smallest Models		Middle Models		Largest Models	
	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)
RANDOM (SMALL)	55.0	0.08	57.8	0.11	62.7	0.22
RANDOM (LARGE)	50.7	0.03	61.6	0.13	64.0	0.20
EMLT w/o DM	63.6	0.12	66.6	0.41	67.3	0.59
EMLT w/o LoRA	63.7	0.14	64.1	0.70	65.0	1.40
EMLT	50.4	0.02	61.1	0.04	65.3	0.12

4.3. Experimental Settings

In this study, we conducted an experimental evaluation of the performance of our model on four different datasets: ListOps, Text, and Retrieval in Long Range Arena and IMDb. These datasets represent different types of tasks and data distributions, and we wanted to see how our model performs in each of these settings.

To evaluate the model's performance, we used the following experimental settings. We ran the evolutionary optimization process for a total of 40 generations. This allowed us to explore a wide range of model architectures and evaluate their performance on the datasets. During each generation, we evaluated the performance of a population of 40 individuals, each with a different model architecture. This allowed us to identify the best-performing architectures and use them as the basis for the next generation. For each generation, we generated a total of 40 offsprings by applying evolutionary operators (such as crossover and mutation) to the best-performing individuals from the previous generation. This allowed us to explore a diverse set of model architectures and find the ones that perform well on the datasets. In addition, we used a learning rate of 0.0001 to train the models on the datasets. This value was chosen based

on our previous experience and the performance of the models on a validation set. During evolution the number of the transformer layers is set to be two. This allowed us to capture long-range dependencies in the data and learn more complex representations. Also, we used a batch size of 32 for the Listops, Text and IMDb datasets and 16 for the Retrieval dataset. This allowed us to trade off between computation time and the ability to capture statistical patterns in the data. All the experiments are conducted with the NVIDIA DGX-A100 workstation with eight A100 GPUs.

Overall, these experimental settings were chosen to evaluate the robustness and generalization ability of our model under different conditions and to identify the architectures that perform well on the datasets.

Table 6: Quantitative results of the ablation experiments on the Retrieval dataset.

Methods	Smallest Models		Middle Models		Largest Models	
	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)
RANDOM (SMALL)	50.3	0.22	57.1	0.40	72.1	0.42
RANDOM (LARGE)	57.2	0.10	65.3	0.15	76.0	0.19
EMLT w/o DM	73.7	0.16	77.3	0.59	81.2	0.90
EMLT w/o LoRA	76.5	0.22	79.8	6.19	82.4	9.12
EMLT	57.3	0.04	67.8	0.05	77.4	0.13

4.4. Overall Results

To demonstrate how the EMLT algorithm improves the dilemma objective and determines the optimal architecture configurations, we have plotted the evolution trajectories on ListOps, Text, and Retrieval in Fig. 11. To enhance the robustness of our experimental results, we have incorporated the IMDb

Table 7: Quantitative results of the ablation experiments on the IMDb dataset.

Methods	Smallest Models		Middle Models		Largest Models	
	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)
RANDOM (SMALL)	53.2	0.01	80.2	0.15	87.5	0.68
RANDOM (LARGE)	49.2	0.01	87.5	0.49	88.4	0.58
EMLT w/o DM	86.2	0.06	87.1	1.8	88.5	3.1
EMLT w/o LoRA	85.2	0.04	85.6	1.8	87.3	3.0
EMLT	63.4	0.01	86.8	0.04	88.5	0.36

dataset. Also, to better illustrate the optimization process, we have computed and displayed the current Pareto frontier only every five generations due to the large number of evaluated data points. This figure could reflect the incremental process of the evolution, and also indicates the convergence of method. In technical terms, a good multi-objective algorithm should aim to approximate the Pareto front, meaning that one of the objectives can be improved under the same constraints. We can obtain thoughtful information from each subplots. On the ListOps dataset, the EMLT algorithm began to converge around the 15-th generation, and the Pareto front became stable in the 25-th generation. On the Text dataset, the algorithm converged starting from the 10-th generation. On the Retrieval dataset, the algorithm converged starting from the 15-th generation, similar to the results observed on the ListOps dataset. On the IMDb dataset, the Pareto front became stable in the 20-th generation. Please note some points on the figure may be not very clear due to the point overlapping, but we can see the convergence tread from the newly generated dominated point. When the algorithm starts to converge, it does not mean that there will be no new domainted points. Therefore, in order to better find the set of Pareto optimals, we set all iterations to 40, which satisfies the algorithm to search for a best frontier.

Next, we can choose the required points which satisfy the minimal computational complexity, and evaluate it on the test dataset to get the final accuracy. For our experiment, we selected configurations with low computational burden and competitive accuracy. Specifically, we chose the configuration with 0.01G FLOPs on the ListOps dataset, the configuration with 0.09G FLOPs on the Text dataset, and the configuration with 0.13G FLOPs on the retrieval dataset. These points were chosen because they have a very low computational burden and still achieve competitive accuracy compared to other competitors. Table 2 provides the overall results of the final optimized architecture and that of other SOTA methods.

On the ListOps dataset, our proposed method EMLT achieved competitive performance with a 37.3% accuracy, which is comparable to other state-of-the-

art methods such as Vanilla Attention, Reformer, Linformer, and Nyströmformer. However, EMLT stands out in terms of computational efficiency with a FLOPs of only 0.01G, which is more than 20 times smaller than the next most efficient model (Transformer-LS with 0.2G FLOPs). This significant reduction in computational cost demonstrates the potential of our method for real-world applications with limited computational resources.

On the Text dataset, EMLT achieved an accuracy of 65.2%, which is again competitive with the state-of-the-art models such as Vanilla Attention, Reformer, and Nyströmformer. In terms of computational efficiency, EMLT's FLOPs of 0.09G is the smallest among all the models, which is more than 4 times smaller than the next most efficient model (Transformer-LS with 0.4G FLOPs). Additionally, EMLT achieved this efficiency without sacrificing accuracy, demonstrating its effectiveness for natural language processing tasks.

On the Retrieval dataset, EMLT achieves a respectable accuracy score of 79.5, which is comparable to other methods such as Reformer and Transformer-LS. However, EMLT still demonstrates impressive computational efficiency with only 0.13G FLOPs, which is more than 6 times faster than the next most efficient method, Nyströmformer, with 1.02G FLOPs.

The EMLT method attains the highest accuracy score of 87.9 on the IMDb dataset, surpassing all other methods. Notably, EMLT boasts the lowest FLOPs, standing at 0.14G. EMLT achieves nearly 9 times faster performance than the original Transformer(Vanilla Attention). Overall, EMLT outperforms all state-of-the-art methods, showcasing its excellence.

When comparing EMLT to other methods across all datasets, it is evident that EMLT offers the best trade-off between accuracy and computational efficiency. For instance, while Reformer performs well on the Retrieval dataset with an accuracy score of 78.6, it requires 1.15G FLOPs which is almost 9 times slower than the proposed EMLT. Similarly, while Performer achieves a high accuracy score of 65.2 on the Text dataset, it requires 0.82G FLOPs which is more than 9 times slower than our proposed EMLT.

In order to show the performance of each method more intuitively, we have

drawn the performance of all methods about FLOPs and accuracy on each test dataset in Fig. 10. In that figure, the data point closer to the upper left corner mean higher accuracy and lower FLOPs. Among all four data concentrations, our method is located on the far right of each figure, and the accuracy is equivalent to the existing state-of-the-art methods, while reducing the computational burden by a large margin.

Overall, these results demonstrate the strong performance of our proposed method in terms of accuracy and computational efficiency. Our proposed method is among the top performers in terms of accuracy and has the lowest computational burden among the seven methods compared. Further research could focus on exploring the mechanisms behind the improved performance of our method and on evaluating its performance on a wider range of tasks and datasets.

Based on these results, we can conclude that the proposed EMLT is more effective at reducing the model complexity than other efficient transformer-based methods. This finding is significant because it has the potential to improve the effectiveness of language modeling and lead to more compact models.

Table 8: The sensitive analyses of the proposed low-rank attention module (on the ListOps dataset).

r \ w	1	2	4	8	16	32
1	36.5	36.9	36.9	37.2	37.2	36.7
2	37.1	36.8	37.1	35.6	37.0	37.0
4	37.1	37.8	35.5	37.0	36.2	36.8
8	26.8	37.0	35.6	37.1	37.1	37.0
16	36.7	38.2	36.6	36.6	36.8	36.1
32	34.1	36.1	36.5	36.9	36.9	33.5

4.5. Ablation Studies

In this part, we will conduct ablation experiments on each module proposed in the experiment, and analyze the performance of each part to prove the effectiveness of our method.

Table 9: The sensitive analyses of the dynamic mask modules, under the original full attention mechanism.

Dynamic Token Settings	ListOps		Text		Retrieval		Average
	Accuracy (%)	FLOPs(G)	Accuracy (%)	FLOPs(G)	Accuracy (%)	FLOPs(G)	
OA w/o DM	37.1	1.21	65.4	4.57	82.3	9.14	61.6
DM+OA+#1(1.0)	38.1	1.23	64.9	4.60	78.2	9.20	60.4
DM+OA+#1(0.75)	39.1	0.72	64.6	2.65	77.3	5.30	60.3
DM+OA+#1(0.8),#2(0.8)	38.2	0.69	53.7	2.50	77.5	5.01	56.5
DM+OA+#1(0.7),#2(0.7)	37.7	0.50	52.2	1.78	78.3	3.56	56.0
DM+OA+#1(0.6),#2(0.6)	37.1	0.35	61.4	1.23	77.1	2.46	58.5
DM+OA+#1(0.5),#2(0.5)	37.7	0.24	64.0	0.82	77.7	1.63	59.8
DM+OA+#1(0.4),#2(0.4)	39.3	0.16	64.3	0.52	77.5	1.03	60.4
DM+OA+#1(0.3),#2(0.3)	37.1	0.10	64.6	0.30	73.1	0.60	58.3
DM+OA+#1(0.2),#2(0.2)	36.5	0.06	62.5	0.16	75.5	0.31	58.2
DM+OA+#1(0.1),#2(0.1)	36.9	0.03	61.4	0.07	76.5	0.14	58.3

Table 10: The sensitive analyses of the dynamic mask modules, under the low-rank attention settings. The "N/A" denotes the pruning ratio is invalid for the corresponding models.

Dynamic Token Settings	ListOps		Text		Retrieval		Average
	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)	Accuracy (%)	FLOPs (G)	
DM+LoRA+#2(∼0.90)	35.7	0.20	64.6	0.41	60.6	0.82	53.6
DM+LoRA+#2(0.75)	32.6	0.19	65.3	0.38	65.3	0.75	50.4
DM+LoRA+#2(0.5)	37.0	0.16	64.6	0.33	60.0	0.65	53.8
DM+LoRA+#2(0.4375)	N/A	N/A	65.2	0.31	59.8	0.63	N/A
DM+LoRA+#2(0.375)	37.0	0.15	64.1	0.30	61.2	0.60	54.10
DM+LoRA+#2(0.3125)	N/A	N/A	65.2	0.29	58.9	0.58	N/A
DM+LoRA+#2(0.25)	36.8	0.14	63.8	0.28	59.3	0.55	53.3

To better inspect the effect of evolutionary algorithm, we add experiments with random search of small and large sampled data points from the gene encoding space. The "RANDOM (SMALL)" will sample 40 data points from the proposed gene encoding space, which is the same as the population size defined in the EMLT. The "RANDOM (LARGE)" will sample and evaluate 1600 data points from the encoding space. Therefore, the total computational burden of the "RANDOM (LARGE)" is the same as that of the evolutionary process of the EMLT. Afterwards, the Pareto frontier of the all the sampled architectures will be computed and compared with the proposed EMLT. Under the same amount of computational complexity, we can see the influence of the heuristic evolutionary search algorithm on the final optimization result, which can be seen from the Pareto front of all the models finally obtained in Fig. 11. Com-

pared with the random search algorithm, the Pareto front obtained by EMLT shows obvious advantages on the four datasets, which means that the method is effective in searching and optimizing the architectures of the dynamic masking transformers.

In this work, both LoRA and DM can accelerate the transformer model alone. In order to explore the contribution of the two modules to the final performance, we also conducted ablation experiments on the two modules of LoRA and DM. By removing the corresponding modules separately, we can see the impact of the two modules on the final results in Fig. 11. We use "EMLT w/o DM" to indicate the DMET method without the DM module, and "EMLT w/o LoRA" to indicate the EMLT without the LoRA module. In the results of the ListOps dataset, the LoRA module can achieve higher accuracy than the DM module with similar computational complexity. In the Text dataset, The accuracy of the LoRA module is better than that of using the DM module sole in the case of similar computational burden. In addition, on the Retrieval dataset, it can be clearly seen that there is an intersection between the methods using the DM module and the algorithm using the LoRA module on the Pareto front. This shows that at a certain complexity, there is a performance turning point between the method based solely on the DM module and the method based solely on the LoRA module. In addition, we can also see that the DM algorithm has more models with a larger amount of compleixty than the LoRA module under the original attention mechannism. Finally, from Fig. 11, for all the datasets, the final combination of DM and LoRA to form the EMLT method can well promote the optimization of the final architectures of the transformer models.

Furthermore, to demonstrate the ablation experiments quantitatively, we sampled the smallest model, medium model, and largest model from the Pareto frontier of each method, and report all selected models on the ListOps, Text, and Retrieval in Long Range Arena and IMDb datasets in Tables 4, 5, 6 and 7, respectively. It is worth noting that in all the tables, the significant advantage of the EMLT method in terms of computational efficiency is demonstrated.

e.g., among all the smallest, medium, and largest models, the computational consumption of the model obtained by EMLT is almost the smallest, and the maximum relative efficiency can be improved by nearly 70 times. However, not all the improvements are obvious. For example, in Table 5, the search results of the "RANDOM (LARGE)" are closer to our method on the smallest model, which shows that under that search conditions of the same computational scale , the evolutionary search paradigm may not be able to further compress the model size of the transformers. That is, the complexity of the model has reached a relatively extreme value on the current Text dataset, and it is difficult to further compress the size of the transformer model through a heuristic search method. Recall that the quality of the model depends on the two indicators of complexity and accuracy. If a model is better in one metric but lower in another, it doesn't fully judge the pros and cons of the methods.

4.6. Further Investigation

In this section, we conduct further analysis of the proposed modules. We first conduct a sensitivity analysis on the key parameters r and w in the low-rank attention to find out the model with the highest accuracy. We adopted the grid search method to search for the most important parameters of r and w. Specifically, in the experiment, we used a two-layer transformer model employing the low-rank attention layer for training, and reported the results of the corresponding parameter pair in Table 8. In the results of this grid search, we found that even with only two parameters r, w, it has a significant impact on the final experimental results. For example, the best result in the table reaches 38%, but can be as low as 26% for the worst result. It can also be seen from the results in this table that even without considering the computational complexity, the influence of the only two parameters on the accuracy rate is not linear. In addition, it is worth noting that although we only analyzed these two parameters on the ListOps Dataset, it is enough to prove the necessity of our parameter search and optimization for the architectures of the dynamic masking transformer since the final search space covers more parameters belonging

to the non-linear optimization problems.

Next, we tried to test the effect of the pruning rate ρ on the performance of the transformer model applying the DM module, and tested the original attention mechanism and the proposed low-rank attention mechanism respectively. The transformer used in these experiments is still a two-layer model. We list the corresponding results in the Tables 9 and 10, where the string "DM+OA+#1(0.8), #2(0.8)" indicates that the corresponding experimental results use the DM module and original attention together, and the first layer and the second layer have remained 80% of the input tokens.

In Table 9, for a more convenient and intuitive exhibition of the impact of adding the DM module on the original attention. We added an experiment of adding a DM layer with $\rho = 1$ before the first layer, namely "DM+OA+#1(1.0)". It can be seen from the results that the impact of adding the DM module on performance cannot be ignored, even if it is set to retain 100% of the target tokens. Compared with the original Attention, the addition of DM slightly increases the complexity of the model. But in terms of accuracy, the introduction of DM has achieved higher accuracy in ListOps dataset. On the contrary, the results on the other two datasets are worse. In addition, as the pruning ratio increases, we find a more counter-intuitive phenomena. In theory, as more and more tokens are removed, the accuracy should be getting lower and lower. Although the transformer models employing both of the DM module and original attention mechanism has shown this tread in the four datasets as a whole, there are exceptions. e.g., on the ListOps dataset, the result of "DM+OA+#1(0.4)#2(0.4)" obtains higher accuracy than all the results of the its above methods. In the Table 10 where the original attention mechanism is replaced by the LoRA module, this phenomenon is more obvious. Such results show that for the introduction of the DM module, the parameter ρ does not affect the final result in an intuitive sense, and there may be more complicated mechanisms, especially mixing other hyper-parameters, just like r, w for the LoRA module. They have similar impacts to some extent.

5. Conclusion

The complexity of sequence attention computing poses a huge challenge to natural language processing. We propose a novel efficient Transformer method based on evolutionary algorithms, named EMLT, that can automatically optimize the model hyper-parameters within the proposed low-rank attention and dynamic masking, to accelerate the model while preserving the accuracy. To the best of our knowledge, this is the first work that introduces the evolutionary algorithm to the efficient transformers. Additionally, we conduct experiments on the benchmark datasets and obtain competitive results with high acceleration rate. The FLOPs of the model is still at least four times lower than that of the most advanced efficient transformer. This discovery may help to obtain semantic representations in the field of tackling with the long text training problem.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436.

[2] 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).

[3] F. A. Acheampong, H. Nunoo-Mensah, W. Chen, Transformer models for text-based emotion detection: a review of bert-based approaches, Artificial Intelligence Review 54 (8) (2021) 5789–5829.

[4] Y. Tay, M. Dehghani, D. Bahri, D. Metzler, Efficient transformers: A survey, ACM Computing Surveys (CSUR) (2020).

[5] C. Zhu, W. Ping, C. Xiao, M. Shoeybi, T. Goldstein, A. Anandkumar, B. Catanzaro, Long-short transformer: Efficient transformers for language and vision, Advances in Neural Information Processing Systems 34 (2021) 17723–17736.

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

[7] 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, in: International Conference on Learning Representations, 2021.
URL https://openreview.net/forum?id=qVyeW-grC2k

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

[9] J. Qiu, H. Ma, O. Levy, W.-t. Yih, S. Wang, J. Tang, Blockwise self-attention for long document understanding, in: Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 2555–2565.

[10] 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.

[11] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, V. Singh, Nyströmformer: A nyström-based algorithm for approximating self-attention, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 14138–14148.

[12] D. Ye, Y. Lin, Y. Huang, M. Sun, Tr-bert: Dynamic token reduction for accelerating bert inference, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 5798–5809.

[13] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, C.-J. Hsieh, Dynamicvit: Efficient vision transformers with dynamic token sparsification, Advances in neural information processing systems 34 (2021) 13937–13949.

[14] H. Wang, Z. Zhang, S. Han, Spatten: Efficient sparse attention architecture with cascade token and head pruning, in: 2021 IEEE International

Symposium on High-Performance Computer Architecture (HPCA), IEEE, 2021, pp. 97–110.

[15] Y. Sun, B. Xue, M. Zhang, G. G. Yen, Evolving deep convolutional neural networks for image classification, IEEE Transactions on Evolutionary Computation 24 (2) (2019) 394–407.

[16] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, K. C. Tan, A survey on evolutionary neural architecture search, IEEE transactions on neural networks and learning systems (2021).

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

[18] C. Liu, J. Wang, X. Chen, Threat intelligence att&ck extraction based on the attention transformer hierarchical recurrent neural network, Applied Soft Computing 122 (2022) 108826.

[19] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.

[20] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, in: Proceedings of the International Conference on Learning Representations, 2019.

[21] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, in: International Conference on Learning Representations, 2020.
URL https://openreview.net/forum?id=H1eA7AEtvS

[22] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, CoRR abs/1910.01108 (2019).

[23] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, H. Wang, Ernie 2.0: A continual pre-training framework for language understanding, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 8968–8975.

[24] 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).

[25] N. Kitaev, L. Kaiser, A. Levskaya, Reformer: The efficient transformer, in: International Conference on Learning Representations, 2020.
URL https://openreview.net/forum?id=rkgNKkHtvB

[26] 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.

[27] 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.

[28] J. Ainslie, S. Ontañón, C. Alberti, P. Pham, A. Ravula, S. Sanghai, Etc: Encoding long and structured data in transformers., CoRR abs/2004.08483 (2020).

[29] A. Gupta, J. Berant, Gmat: Global memory augmentation for transformers, arXiv preprint arXiv:2006.03274 (2020).

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

[31] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, D. Tran, Image transformer, in: International conference on machine learning, PMLR, 2018, pp. 4055–4064.

[32] P. J. Liu*, M. Saleh*, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, N. Shazeer, Generating wikipedia by summarizing long sequences, in: International Conference on Learning Representations, 2018.
URL https://openreview.net/forum?id=Hyg0vbWC-

[33] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, W. Liu, Ccnet: Criss-cross attention for semantic segmentation, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 603–612.

[34] 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.

[35] Q. Guo, X. Qiu, P. Liu, Y. Shao, X. Xue, Z. Zhang, Star-transformer, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 1315–1325.

[36] G. I. Winata, S. Cahyawijaya, Z. Lin, Z. Liu, P. Fung, Lightweight and efficient end-to-end speech recognition using low-rank transformer, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 6144–6148.

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

[38] 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).

[39] 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.

[40] B. Chen, T. Dao, E. Winsor, Z. Song, A. Rudra, C. Ré, Scatterbrain: Unifying sparse and low-rank attention, Advances in Neural Information Processing Systems 34 (2021) 17413–17426.

[41] 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.

[42] P. Stock, A. Fan, B. Graham, E. Grave, R. Gribonval, H. Jegou, A. Joulin, Training with quantization noise for extreme model compression, in: International Conference on Learning Representations, 2021.
URL https://openreview.net/forum?id=dV19Yyi1fS3

[43] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, Z. Chen, {GS}hard: Scaling giant models with conditional computation and automatic sharding, in: International Conference on Learning Representations, 2021.
URL https://openreview.net/forum?id=qrwe7XHTmYb

[44] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, et al., Glam: Efficient scaling of language models with mixture-of-experts, in: International Conference on Machine Learning, PMLR, 2022, pp. 5547–5569.

[45] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, Q. Liu, Tinybert: Distilling bert for natural language understanding, in: Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 4163–4174.

[46] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, in: Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 10012–10022.

[47] A. Vaswani, P. Ramachandran, A. Srinivas, N. Parmar, B. Hechtman, J. Shlens, Scaling local self-attention for parameter efficient visual backbones, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 12894–12904.

[48] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, J. Carreira, Perceiver: General perception with iterative attention, in: International conference on machine learning, PMLR, 2021, pp. 4651–4664.

[49] P. Zhang, X. Dai, J. Yang, B. Xiao, L. Yuan, L. Zhang, J. Gao, Multi-scale vision longformer: A new vision transformer for high-resolution image encoding, in: Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 2998–3008.

[50] L. Liu, X. Chen, S. Zhu, P. Tan, Condlanenet: a top-to-down lane detection framework based on conditional convolution, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 3773–3782.

[51] P. Michel, O. Levy, G. Neubig, Are sixteen heads really better than one?, Advances in neural information processing systems 32 (2019).

[52] X. Su, S. You, J. Xie, M. Zheng, F. Wang, C. Qian, C. Zhang, X. Wang, C. Xu, Vitas: Vision transformer architecture search, in: European Conference on Computer Vision, Springer, 2022, pp. 139–157.

[53] S. Sun, Y. Cheng, Z. Gan, J. Liu, Patient knowledge distillation for BERT model compression, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 4323–4332. doi:10.18653/v1/D19-1441.
URL https://aclanthology.org/D19-1441

[54] J. Xin, R. Tang, J. Lee, Y. Yu, J. Lin, Deebert: Dynamic early exiting for accelerating bert inference, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 2246–2251.

[55] W. Liu, P. Zhou, Z. Wang, Z. Zhao, H. Deng, Q. Ju, Fastbert: a self-distilling bert with adaptive inference time, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 6035–6044.

[56] S. Goyal, A. R. Choudhury, S. Raje, V. Chakaravarthy, Y. Sabharwal, A. Verma, Power-bert: Accelerating bert inference via progressive word-vector elimination, in: International Conference on Machine Learning, PMLR, 2020, pp. 3690–3699.

[57] G. Kim, K. Cho, Length-adaptive transformer: Train once with length drop, use anytime with search, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online, 2021, pp. 6501–6511. doi:10.18653/v1/2021.acl-long.508.
URL https://aclanthology.org/2021.acl-long.508

[58] O. Press, N. Smith, M. Lewis, Train short, test long: Attention with linear biases enables input length extrapolation, in: International Conference on Learning Representations, 2022.
URL https://openreview.net/forum?id=R8sQPpGCv0

[59] H. Mo, L. L. Custode, G. Iacca, Evolutionary neural architecture search for remaining useful life prediction, Applied Soft Computing 108 (2021) 107474.

[60] D. Li, Y. Bai, Z. Bai, Y. Li, C. Shang, Q. Shen, Decomposed neural architecture search for image denoising, Applied Soft Computing 124 (2022) 108914.

[61] L. Xie, A. Yuille, Genetic cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 1379–1388.

[62] S. A. R. Shah, W. Wu, Q. Lu, L. Zhang, S. Sasidharan, P. DeMar, C. Guok, J. Macauley, E. Pouyoul, J. Kim, et al., Amoebanet: An sdn-enabled network service for big data science, Journal of Network and Computer Applications 119 (2018) 70–82.

[63] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697–8710.

[64] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International journal of computer vision 115 (3) (2015) 211–252.

[65] B. Wang, B. Xue, M. Zhang, Surrogate-assisted particle swarm optimization for evolving variable-length transferable blocks for image classification, IEEE transactions on neural networks and learning systems (2021).

[66] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE transactions on evolutionary computation 6 (2) (2002) 182–197.

[67] B. Feng, D. Liu, Y. Sun, Evolving transformer architecture for neural machine translation, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2021, pp. 273–274.

[68] H. Wang, Z. Wu, Z. Liu, H. Cai, L. Zhu, C. Gan, S. Han, HAT: Hardware-aware transformers for efficient natural language processing, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 7675–7688. doi:10.18653/v1/2020.acl-main.686.
URL https://aclanthology.org/2020.acl-main.686

[69] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, A. Weller, Rethinking attention with performers, in: International Conference on Learning Representations, 2021.
URL https://openreview.net/forum?id=Ua6zuk0WRH