Sparse Transformer for Long-term Forecasting via Genetic Algorithm

Jingeun Kim
Dept. IT Conv., Gachon University
Gyeonggi-do, Republic of Korea
wlsrms27@gachon.ac.kr

Yourim Yoon
Dept. Comp. Eng., Gachon University
Gyeonggi-do, Republic of Korea
yryoon@gachon.ac.kr

ABSTRACT

In recent years, transformer models have achieved remarkable success across various fields. However, reducing the computational complexity while minimizing performance degradation has become a critical challenge. To address this challenge, we propose a pruning method for transformers, using a genetic algorithm, aimed at forecasting long-term time-series data. We used binary encoding to remove unnecessary information from the multi-head attention layers in the transformer encoder. It also leverages Fisher information-based fitness approximation to reduce the evaluating time for each individual. Additionally, we modified the knapsack problem formulation to penalize individuals exceeding the predefined threshold, ensuring a balance between maintaining a compression rate and selecting the best-performing individuals. Using eight time-series datasets, the sparse transformer consistently achieved superior performance compared to the vanilla transformer, with significantly fewer computational complexity. These results demonstrate that the proposed method effectively eliminates redundant attention heads.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; *Bio-inspired approaches.*

KEYWORDS

genetic algorithm, pruning, time series, forecasting, deep learning

ACM Reference Format:
Jingeun Kim and Yourim Yoon. 2025. Sparse Transformer for Long-term Forecasting via Genetic Algorithm. In *Genetic and Evolutionary Computation Conference (GECCO '25 Companion), July 14–18, 2025, Malaga, Spain.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3712255.3726750

1 INTRODUCTION

In recent years, transformer architecture has been a game-changer across various fields, including natural language processing, computer vision, time series forecasting, and more [3, 7].

Despite its outstanding performance, the transformer architecture faces several challenges. Due to its sophisticated computational demands, it requires a vast number of parameters, and it lead to significant increase the model size.

Previous studies have successfully pruned the transformer architecture in various fields, such as computer vision (CV) and natural language processing (NLP), effectively reducing computational costs [5, 9]. TPrune [5] is a pruning framework designed to optimize transformer architectures for resource-constrained platforms. TPrune employs a structured pruning approach to reduce computational costs and inference time, achieving a higher compression rate with less performance degradation in NLP tasks. Lu Yu and Wei Xiang [9] developed an explainable pruning framework (X-Pruner) to address the high computational costs and memory requirements of vision transformer models, which hinder their deployment on edge devices. X-Pruner employs adaptive layer-wise pruning and introduces an explainability-aware mask. It demonstrated superior performance compared to other pruning methods, significantly reducing computational costs with minimal performance degradation.

However, optimizing transformer architectures using genetic algorithms specifically for the time-series data remains unexplored. While transformer architectures for long-term time-series forecasting may be smaller than those in other fields, it is critical to reduce computational costs, such as floating-point operations per second (FLOPs), while minimizing performance degradation.

In this paper, we propose a structured pruning method to optimize the multi-head attention layers of the encoder block of the vanilla transformer architecture [8] for long-term time-series forecasting using a genetic algorithm. We incorporate a Fisher information-based fitness approximation to accelerate fitness evaluation during the genetic algorithm. Furthermore, we formulate the pruning of the transformer encoder structure as an optimization problem to identify individuals with exceptional objective values while reducing FLOPs. We conducted experiments on eight time-series datasets and compared the performance of the vanilla transformer with the sparse transformer using a genetic algorithm, evaluating them with mean squared error (MSE), mean absolute error (MAE), and FLOPs as metrics.

2 THE PROPOSED METHOD

2.1 Vanilla transformer architecture

Figure 1 illustrates the vanilla transformer architecture. Generally, the vanilla transformer consists of an encoder and a decoder. An encoder block primarily comprises a multi-head self-attention module and a position-wise feed-forward network (FFN), and a decoder block consists of two multi-head self-attention modules and a position-wise FFN. We configured a vanilla transformer architecture for long-term forecasting, consisting of two encoder blocks

ACM ISBN 979-8-4007-1464-1/2025/07
https://doi.org/10.1145/3712255.3726750

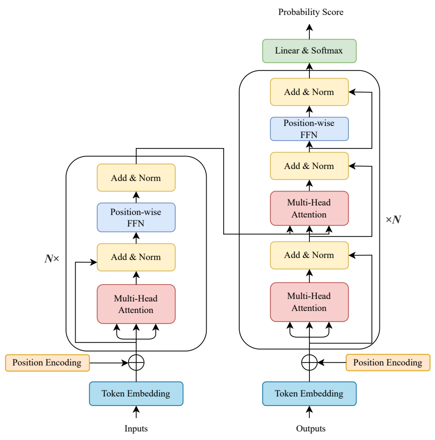

Figure 1: Transformer architecture

and one decoder block. The multi-head attention (MHA) layer in both the encoder and decoder blocks is composed of N attention heads, each with its own set of parameters. Here, N is set to 8 for the vanilla transformer.

In the encoder block, MHA is computed as follows:

$$\begin{aligned} \text{MHA}(Q,K,V) &= \sum_{i=1}^{N} \text{Attn}_i(QW_i^Q, KW_i^K, VW_i^V) \\ \textit{where} \quad \text{Attn}(Q,K,V) &= \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \end{aligned} \tag{1}$$

where N is the number of attention heads, and W_i^Q, W_i^K, W_i^V are the learned projection matrices for the query, key, and value for the N-th head, d_k indicates the dimension of the models, which is set to 512.

For long-term time series forecasting, the vanilla transformer employs two convolution layers for the point-wise FFN following the MHA layer. These convolution layers expand and then reduce the dimensionality of the outputs. The normalization layer is applied after the MHA and point-wise FFN layers.

2.2 Encoding

Before being input into the encoding layer, the time-series data was transformed into high-dimensional matrix that the model can process through embedding layers. For the input $\mathbf{X}^{\text{in}} \in \mathbb{R}^{B\times L\times C}$, where B represents the batch size, L denotes the sequence length, and C is the number of features, the sum of token embedding, temporal embedding, and positional embedding is applied to obtain the output $\mathbf{X}^{\text{out}} \in \mathbb{R}^{B\times L\times d_k}$.

As mentioned above, the MHA layer consists of N attention heads. To reduce the computation complexity of the vanilla transformer, we pruned attention heads that contain less information and are deemed unnecessary. The output of the MHA with N attention heads is the sum of output $Attn(X) \in \mathbb{R}^{d_k\times d_k/N}$ of each attention head. We employed binary encoding with an array length equal to the number of attention heads for each encoder. The encoding is represented as a 2×8 array, where each element corresponds to a binary value based on discrete uniform distribution. If the ith element is set to zero, the weights of the ith attention head are removed. In contrast, if the value is set to one, the weights for that attention head are preserved.

2.3 Fitness approximation

In a previous study [1], the constraints of the Knapsack problem were simplified and then formulated with the objective of reducing the complexity of the deep learning model.

$$\begin{aligned} &\text{maximize} && \sum_{i=1}^{n} v_i x_i \\ &\text{subject to} && \sum_{i=1}^{n} w_i x_i \le W \quad \text{and} \quad x_i \in \{0,1\} \end{aligned} \tag{2}$$

where x_i is a binary decision variable, taking the value 1 if item i is selected and 0 otherwise, v_i denotes the value of item i, w_i represents the weight of item i, and W is the total capacity of the knapsack. In [1], the value v and capacity W of the knapsack problem were defined using accuracy and FLOPs to prune the convolution neural network.

Inspired by this approach, we propose a similar method. However, to reduce evaluation time, we used fisher information instead of accuracy to approximate the amount of preserved information [4, 6]. We set the value v as the Fisher information of the sparse transformer, the weight w as the FLOPs of the sparse transformer (T_{pruned}), and the capacity W as $\frac{2}{3}$ of the FLOPs of the vanilla transformer (T). The problem we formulated is as follows:

$$\begin{aligned} &\text{maximize} && \text{Fisher Information}(T_{\text{pruned}}) \\ &\text{subject to} && \text{FLOPs}(T_{\text{pruned}}) \le \text{FLOPs}(T)\cdot\frac{2}{3} \end{aligned} \tag{3}$$

The fitness of an individual is determined by the fisher information as follows:

$$\begin{aligned} \mathcal{I} \approx \frac{1}{H\cdot L}\sum_{l=1}^{L}\sum_{h=1}^{H}\Bigg(&\left(\frac{\partial\mathcal{L}(D_s)}{\partial W_{l,h}^Q}\right)^2 + \left(\frac{\partial\mathcal{L}(D_s)}{\partial W_{l,h}^K}\right)^2 \\ &+ \left(\frac{\partial\mathcal{L}(D_s)}{\partial W_{l,h}^V}\right)^2 + \left(\frac{\partial\mathcal{L}(D_s)}{\partial W_{l,h}^O}\right)^2\Bigg) \end{aligned} \tag{4}$$

where L indicates the number of encoders, and D_s represents a sampled tiny subset. The sampled tiny subset D_s from the first batch of the training set is input into the encoders, where it passes through N attention heads. The Fisher information is calculated using the first-order method based on the gradients of the weights of the query, key, value, and output, and summed. This sum is then divided by the number of attention heads H and encoders L, and used as the fitness value of the individual. This allows us to evaluate how much each head contributes to preserving information within the MHA. As shown in Equation 3, if the FLOPs of the pruned model exceed $\frac{2}{3}$ of the FLOPs of the vanilla transformer, a penalty is applied by using half of the Fisher information value as the fitness value. Conversely, if the FLOPs of the pruned model are less than or equal to $\frac{2}{3}$ of the FLOPs of the vanilla transformer, the Fisher information value is used as the fitness value without any penalty. We assume that a higher fisher information indicates that the model's parameters capture more information from the data. Therefore, an individual with a higher fitness value is considered superior.

2.4 Genetic operator

Let P denote population size. From the P, $P/2$ pairs are randomly selected, and crossover and mutation are applied to each pair, generating O offspring of the same size as the parents. Parents and newly generated offspring are ranked, with the top 20% of the parents and the top 80% of the offspring being transferred to the next generation. In our experiments, we set the population size to 20 and the number of generations to 50. We used uniform crossover and bit-flip mutation.

- Uniform Crossover: Each bit of the offspring $o = \{o_1, o_2, \ldots, o_p\}$ is independently generated from the corresponding bits of the parents $x = \{x_1, x_2, \ldots, x_p\}$ and $y = \{y_1, y_2, \ldots, y_p\}$ with equal probability of 0.5, a value that is widely used [2]. This method is possible for the offspring to inherit more genetic information from one parent than the other due to the stochastic process.
- Bit-flip mutation: Mutation is applied with mutation rate of 0.5. A ith gene is randomly selected for each row, and the corresponding bit is flipped, changing 0 to 1 and 1 to 0.
- Repair: We added a repair step to prevent all elements in rows from being filled with zero by genetic operators. If elements in each row do not contain 1, elements will randomly filled with one based on discrete uniform distribution.

3 EXPERIMENTS

3.1 Experimental setup

Table 1: Overview of datasets

Dataset	Dimension	Length	Domain
ETTh1	7	17,420	Electricity (1h)
ETTh2	7	17,420	Electricity (1h)
ETTm1	7	69,680	Electricity (15 min)
ETTm2	7	69,680	Electricity (15 min)
Electricity	321	26,304	Electricity (1h)
Weather	21	52,696	Environment (10 min)
Traffic	862	17,544	Transportation (1 h)
Exchange	8	7,588	Economic (1 d)

We used eight long-term time series dataset including ETT datasets, and their basic attribution are shown in the Table 1. the look-back window is uniformly set to 96, and the prediction lengths are 96, 192, 336, and 720, respectively.

We trained the vanilla transformer using the training dataset, and applied proposed pruning approach. We then obtained the performance of the sparse transformer without additional training or fine-tuning. Each model is trained for 10 epochs, with patience set to 3, and the loss function is the MSE, to keep consistent with the mainstream model training methods.

Additionally, we used MSE and MAE to evaluate the performance of the transformer, while FLOPs and Saving FLOPs were used as measures of compression. All experiments were implemented using PyTorch and experimented on an Intel Core i7-7700k CPU (4.20GHz), NVIDIA GeForce RTX 4090 GPU, and 64GB RAM.

3.2 Result

Table 2 shows the detailed performance comparison of the vanilla transformer and the sparse transformer on eight datasets. Sparse transformer outperformed 24 out of 32 evaluation metrics. This demonstrates that our pruning method effectively compressed the MHA and achieved better performance with fewer FLOPs. Notably, for the ETTh1, ETThm2, Weather, and Exchange datasets, the performance of the sparse transformer was better than that of the vanilla transformer. For ETTh2, ETThm1, and ECL datasets, the proposed method achieved superior performance in all but one of the four evaluation metrics. In the Traffic dataset, the performance was not as good, which indicates that as the number of dimensions increases, more computation demands are required to achieve better performance. Additionally, it was observed that the sparse transformer had higher FLOPs for the Traffic dataset than other datasets. This reflects the increased computational demand as the dimensionality increases. However, by applying a penalty to high-FLOPs individuals according to Equation 3, the proposed pruning method can minimize FLOPs while achieving minimal performance degradation. The proposed pruning method achieved a compression rate ranging from a maximum of 2.68× to a minimum of 1.63×, with an average compression of 2.05×.

4 CONCLUSION AND FUTURE WORKS

In this study, we proposed a method for pruning the attention heads of the multi-head attention layer in the encoder to reduce computational demands while minimizing performance degradation for long-term time-series forecasting. To evaluate individuals in the genetic algorithm, Fisher information was used to calculate how much each head preserves information. Based on the knapsack problem formulation, the model size was adjusted to ensure that the defined FLOPs were not exceeded. The proposed pruning method demonstrated outstanding performance across eight datasets compared to the vanilla transformer. We observed that the sparse transformer achieved similar or even better performance than the vanilla transformer with significantly fewer FLOPs.

However, this study has several limitations. First, the architectures of current state-of-the-art models are more complex than the vanilla transformer. Therefore, further research is needed to develop model-agnostic compression methods. Second, more extensive ablation studies are required. It is necessary to compare the proposed method with other meta-heuristic and mathematical programming approaches, as well as to investigate the impact of parameter tuning in the genetic algorithm.

ACKNOWLEDGMENTS

This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry & Energy, Republic of Korea (RS-2024-00442817). This work was also supported by the Ministry of Education of the Research

Table 2: Comparison performances between transformer and sparse transformer via genetic algorithm across datasets. Boldface indicates the best performance. The FLOPs of the transformer was 6,881,280.

		Vanilla transformer		Sparse transformer via genetic algorithm			
Dataset	pred length	MSE	MAE	MSE	MAE	FLOPs	Saving FLOPs
ETTh1	96	0.5331	0.4929	**0.4871**	**0.4557**	3,231,744	2.13
	192	0.6850	0.5562	**0.4897**	**0.4775**	3,231,744	2.13
	336	0.6945	0.5587	**0.5276**	**0.5019**	2,899,968	2.37
	720	0.7224	0.6139	**0.6667**	**0.5902**	3,231,744	2.13
ETTh2	96	0.3901	0.4194	**0.3769**	**0.4105**	3,895,296	1.77
	192	0.5599	0.4852	**0.5459**	**0.4775**	2,899,968	2.37
	336	**0.5770**	**0.5109**	0.5837	0.5119	3,231,744	2.13
	720	0.5637	0.5152	**0.5011**	**0.4874**	2,899,968	2.37
ETTm1	96	0.4004	0.4117	**0.3863**	**0.4012**	3,895,296	1.77
	192	**0.4808**	0.4462	0.4976	**0.4399**	4,227,072	1.63
	336	0.5510	0.4830	**0.5277**	**0.4679**	3,895,296	1.77
	720	0.6656	0.5368	**0.6464**	**0.5186**	3,231,744	2.13
ETTm2	96	0.2185	0.2918	**0.2137**	**0.2867**	2,568,192	2.68
	192	0.4268	0.3919	**0.3272**	**0.3529**	2,899,968	2.37
	336	0.5655	0.4650	**0.4891**	**0.4338**	3,563,520	1.93
	720	0.5880	0.4899	**0.5309**	**0.4647**	3,563,520	1.93
ECL	96	**0.2932**	**0.3946**	0.3122	0.4141	3,895,296	1.77
	192	0.4014	0.4648	**0.3975**	**0.4606**	3,563,520	1.93
	336	0.4909	0.5226	**0.4152**	**0.4744**	3,231,744	2.13
	720	0.4884	0.5254	**0.4722**	**0.5194**	3,231,744	2.13
Weather	96	0.4759	0.4773	**0.4243**	**0.4461**	3,895,296	1.77
	192	0.6309	0.5652	**0.4930**	**0.4819**	2,899,968	2.37
	336	0.6955	0.6080	**0.5048**	**0.4983**	3,563,520	1.93
	720	1.1796	0.8003	**0.7708**	**0.6303**	3,563,520	1.93
Traffic	96	**0.6793**	**0.3772**	0.7566	0.4070	3,563,520	1.93
	192	**0.6922**	**0.3772**	0.7068	0.3877	3,895,296	1.77
	336	0.7063	**0.3849**	**0.7002**	0.3858	4,227,072	1.63
	720	**0.6840**	**0.3723**	0.6898	0.3802	4,227,072	1.63
Exchange	96	0.7272	0.6644	**0.6430**	**0.6505**	3,231,744	2.13
	192	1.1835	0.8254	**0.9650**	**0.7673**	3,231,744	2.13
	336	1.7286	0.996	**1.6633**	**0.9839**	2,568,192	2.68
	720	2.3274	1.2566	**2.0445**	**1.1574**	3,231,744	2.13
Average	-	0.6877	0.5400	0.6174	0.4944	3,418,368	2.05

Foundation of Korea (NRF-2022S1A5C2A07090938) and supported the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1F1A1066017).

REFERENCES

[1] Yonathan Aflalo, Asaf Noy, Ming Lin, Itamar Friedman, and Lihi Zelnik. 2020. Knapsack pruning with inner distillation. *arXiv preprint arXiv:2002.08258* (2020).

[2] Benjamin Doerr and Zhongdi Qu. 2023. Runtime analysis for the NSGA-II: Provable speed-ups from crossover. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 12399–12407.

[3] Saidul Islam, Hanae Elmekki, Ahmed Elsebai, Jamal Bentahar, Nagat Drawel, Gaith Rjoub, and Witold Pedrycz. 2024. A comprehensive survey on applications of transformers for deep learning tasks. *Expert Systems with Applications* 241 (2024), 122666.

[4] Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems* 35 (2022), 24101–24116.

[5] Jiachen Mao, Huanrui Yang, Ang Li, Hai Li, and Yiran Chen. 2021. Tprune: Efficient transformer pruning for mobile devices. *ACM Transactions on Cyber-Physical Systems* 5, 3 (2021), 1–22.

[6] Sidak Pal Singh and Dan Alistarh. 2020. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems* 33 (2020), 18098–18109.

[7] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[8] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. 2024. Deep Time Series Models: A Comprehensive Survey and Benchmark. (2024).

[9] Lu Yu and Wei Xiang. 2023. X-pruner: explainable pruning for vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 24355–24363.