Long-range Sequence Modeling with Predictable Sparse Attention

Yimeng Zhuang, Jing Zhang, Mei Tu
Samsung Research China - Beijing (SRC-B)
{ym.zhuang, jing97.zhang, mei.tu}@samsung.com

Abstract

Self-attention mechanism has been shown to be an effective approach for capturing global context dependencies in sequence modeling, but it suffers from quadratic complexity in time and memory usage. Due to the sparsity of the attention matrix, much computation is redundant. Therefore, in this paper, we design an efficient Transformer architecture, named Fourier Sparse Attention for Transformer (FSAT), for fast long-range sequence modeling. We provide a brand-new perspective for constructing sparse attention matrix, i.e. making the sparse attention matrix predictable. Two core submodules are: (1) A fast Fourier transform based hidden state cross module, which captures and pools L^2 semantic combinations in $\mathcal{O}(L \log L)$ time complexity. (2) A sparse attention matrix estimation module, which predicts dominant elements of an attention matrix based on the output of the previous hidden state cross module. By reparameterization and gradient truncation, FSAT successfully learned the index of dominant elements. The overall complexity about the sequence length is reduced from $\mathcal{O}(L^2)$ to $\mathcal{O}(L \log L)$. Extensive experiments (natural language, vision, and math) show that FSAT remarkably outperforms the standard multi-head attention and its variants in various long-sequence tasks with low computational costs, and achieves new state-of-the-art results on the Long Range Arena benchmark.

1 Introduction

Models based on the Transformer architecture (Vaswani et al., 2017) have been firmly established as state of the art approaches across a range of domains like language (Brown et al., 2020; Clark et al., 2020; Devlin et al., 2018), and vision (Carion et al., 2020; Dosovitskiy et al., 2020). The Transformer architecture perceiving long-range context heavily relies on the multi-head self-attention mechanism, in which the relevance of every token pairs is computed to decide the attention scores and token's representations are the weighted average of all tokens using the attention scores.

Despite its effectiveness, self-attention mechanism's quadratic time and memory complexity about the sequence length is an obstacle to extend Transformer for very long sequences, such as document-level text tasks, high-resolution images, videos, etc. Shen et al. (2021, 2018) elaborate the issue of high computational complexity. For instance, more than 68GB GPU memory and 1.6T multiply-accumulation operations are required for a $64 \times 64 \times 32$ 3D feature volume.

Great efforts have been made to develop Transformer's variants for long-range sequence modeling tasks. Tay et al. (2020c) categorize the researches of efficient Transformers: (a) Fixed patterns or combination of patterns (Beltagy et al., 2020; Zaheer et al., 2020), in which the field to be attended is pre-defined by fixed pattern. (b) Learnable patterns (Kitaev et al., 2020; Tay et al., 2020a), in which tokens are sorted or clustered in a data-driven fashion. (c) Memory (Ma et al., 2021; Lee et al., 2019), in which spacial tokens with global view are introduced to compress the input sequence. (d) Low-rank methods (Tay et al., 2021; Wang et al., 2020), which adopt low-rank approximations of the self-attention matrix. (e) Kernels (Katharopoulos et al., 2020; Choromanski et al., 2020a,b), which view the attention mechanism through kernelization. (f) Recurrence (Rae et al., 2019), which connects multiple segments via recurrence structure. Despite their variety, approximating the quadratic-cost attention matrix by the sparsity of attention matrix is the common idea.

In this paper, we propose predictable sparse attention, and name it as Fourier Sparse Attention for Transformer (FSAT) due to fast Fourier transform is a key operation in our method. FSAT is a brand-new perspective of efficient Transformer, i.e. learning the sparse structure of an attention ma-

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 234 - 243

trix in end-to-end fashion. Specifically, we firstly compute the semantic relevance of token pairs and then use it to predict the indices of dominant (non-zero) elements of an attention matrix, and finally attention scores are filled according to the predicted sparse structure. In this process, two problems have to be solved: (1) Efficiently capturing semantic relevance of L^2 token pairs where L is the length of an input sequence. (2) Learning discrete indices with gradient descent algorithm. To this end, we propose pooled hidden state cross to efficiently calculate and compress semantic relevance in $\mathcal{O}(L \log L)$ time complexity. For end-to-end training, we get continuous and meaningful gradients for learning discrete indices by reparameterization and gradient truncation. Consequently, FSAT is out of the scope of Tay et al. (2020c)'s taxonomy. It's worth noting that predictable sparse attention is different from the methods of learnable patterns. Although these methods use learnable algorithms to sort or cluster tokens, they still exploit fixed patterns (chunked patterns). Instead, FSAT directly predicts the sparse structure of an attention matrix without any pre-defined pattern.

In order to fit the predicted sparse attention matrix, the key and value vectors in self-attention mechanism are projected from pooled hidden state cross vectors, which can be viewed as 2-order features of the tokens. As an extra benefit, model's expressiveness may increase. Therefore, unlike some efficient Transformer variants which approximate the quadratic-cost attention matrix at the expense of accuracy, FSAT not only reduces computational complexity but also improves model accuracy in some tasks. On Long Range Arena benchmark, FSAT outperforms the Transformer and several recent efficient self-attention methods by a large margin.

To summarize, our contributions are as follows:

- We propose Fourier Sparse Attention for Transformer (FSAT) to extend Transformer for long sequences. The overall complexity about the sequence length is reduced from $\mathcal{O}(L^2)$ to $\mathcal{O}(L \log L)$.
- We introduce the pooled hidden state cross to implement FSAT.
- Empirically, extensive experiments (natural language, vision, and math) demonstrate the advantages of our proposed methods, and new state-of-the-art results are achieved on the Long Range Arena benchmark.

2 Related Works

2.1 Efficient Transformer

Tay et al. (2020c) have provided a comprehensive overview of existing efficient Transformers. Some promising models are compared with our method in the experiments. Big bird (Zaheer et al., 2020) uses random, sliding window and global attention to build hybrid attention pattern. Performer (Choromanski et al., 2020a,b) utilizes orthogonal random features to approximate softmax-attention kernels with linear complexity. Linformer (Wang et al., 2020) achieves linear complexity by adopting random projections based on the JL lemma to compress the attention length to a fix length. Longformer (Beltagy et al., 2020) combines local windowed attention with task-motivated global attention for long documents. Reformer proposed in Kitaev et al. (2020) clusters similar tokens by locality-sensitive-hashing, and dot-product attention is performed inside clusters.

2.2 Feature Crosses

Feature crosses, which synthesize crossing combinations of features, is a widely used technique for extending features' predictive ability in machine learning. For example, Takahashi et al. (2018) demonstrate their gender identification system leveraging synergy of both texts and images by feature cross technique. Yu et al. (2018) and Seo et al. (2016) utilize crossed feature to design a trilinear attention function. Chen et al. (2021) explore how to search the best feature crosses by submodular optimization. More researches involving feature crosses focus on feature selection (Zadeh et al., 2017; Wei et al., 2015; Hoque et al., 2014; Nie et al., 2010; Guyon and Elisseeff, 2003; Kwak and Choi, 2002; Rogati and Yang, 2002; Weston et al., 2000).

2.3 Fourier Transform in Transformer

Recently, Fourier transform in Transformer has garnered interest. Choromanski et al. (2020a,b) propose Performer by approximating softmax attention-kernels via orthogonal random Fourier features. Tamkin et al. (2020) propose BERT + Prism model using spectral filters in the activations of neurons for producing multi-scale representations, and got positive experimental results at

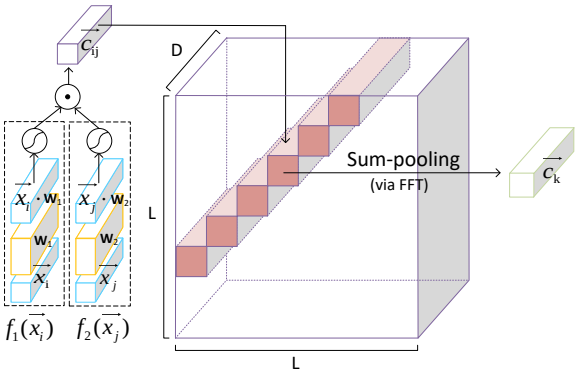

Figure 1: The process of feature mapping, hidden state cross, and sum-pooling along antidiagonals, corresponding to Formula 2.

utterance- and document-level tasks. More radically, Lee-Thorp et al. (2021) reform Transformer by replacing the entire self-attention sub-layer with discrete Fourier transforms along sequence dimension and hidden dimension respectively.

3 Self-attention with Pooled Hidden State Cross

In this section, we start by explaining the motivation of introducing pooled hidden state cross, then introduce how to compute pooled hidden state cross, and finally discuss the way to equip self-attention with pooled hidden state cross.

3.1 Why Pooled Hidden State Cross

Three desiderata motivate our use of pooled hidden state cross: (1) Long-range semantic dependency and relevance can be captured by hidden state crosses, since the combinations of every token pair have been included. Capturing semantic relevance is also the basis of the predictable sparse attention proposed in the next section. (2) Hidden state cross is a way to extract 2-order token features, intuitively it may generate more expressive feature representations. (3) Crossing and pooling hidden states are conducted depth-wisely, so that they can be efficiently implemented via fast Fourier transform.

3.2 Pooled Hidden State Cross

Inspired by the feature cross technique, we propose the concept of hidden state cross. Briefly speaking, feature cross technique (a.k.a feature combination) synthesizes new feature xy by multiplying feature x and feature y. We extend it to the level of hidden state of deep learning models. Specifically, given the hidden states of a token sequence $\vec{x}_0, \cdots, \vec{x}_{L-1}$, we define

$$\vec{c}_{ij} = f_1(\vec{x}_i) \odot f_2(\vec{x}_j) \tag{1}$$

as the crossed hidden state vector of the i-th and the j-th token, where $f(\cdot)$ is a parameterized non-linear feature mapping function, subscripts indicate containing different parameters, and $\odot$ is the Hadamard product. We expect that the semantic combination of two tokens can be learned and encoded in the crossed hidden state vector.

Problems arise when computing the hidden state crosses of L^2 token pairs. Firstly, the computational complexity is $\mathcal{O}(L^2)$ about the sequence length L, which is computationally prohibitive for long sequences. Secondly, the output should be L^2 vectors which is too large to be attended in the Transformer model. To alleviate the problems, crossed hidden states are sum-pooled in this paper.

$$\vec{c}_k = \sum_{i+j=k} \vec{c}_{ij} = \sum_{i+j=k} f_1(\vec{x}_i) \odot f_2(\vec{x}_j) \tag{2}$$

Figure 1 illustrates the computation. The pooled hidden state cross $\vec{c}_k$ represents the sum of crossed hidden states along the k-th antidiagonal. Therefore, the output vectors are compressed from L^2 vectors, i.e. $\{\vec{c}_{ij}\}_{i,j\in[0,L-1]}$, to $2L-1$ vectors, i.e. $\{\vec{c}_k\}_{k\in[0,2L-2]}$.

3.2.1 Implementation via Fast Fourier Transform

Formula 2 can be efficiently implemented by Fast Fourier Transform (FFT). Specifically, hidden states are firstly non-linearly converted by the feature mapping functions, and then 1D discrete Fourier transform is applied along the sequence dimension to transform the mapped hidden states into frequency domain, crossing and pooling are then conducted via multiplication in frequency domain, finally by applying inverse 1D discrete Fourier transform the pooled hidden state crosses are transformed back from frequency domain. By the Hermitian property, the imaginary part of the output is zero. Thus, we can safely only keep the real part of the output and avoid involving complex numbers into the model. Formally,

$$\mathbf{C}_0 = \Re(\mathcal{F}^{-1}(\mathcal{F}(f_1(\mathbf{X})) \odot \mathcal{F}(f_2(\mathbf{X})))) \tag{3}$$

in which, $\mathbf{X} \in \mathbb{R}^{L\times D}$ denotes the matrix consisting of the L D-dimensional hidden states. $\mathcal{F}$ and $\mathcal{F}^{-1}$

are 1D Fourier transform and inverse 1D Fourier transform respectively. $\Re$ means keeping the real part of complex numbers. $\mathbf{C}_0 \in \mathbb{R}^{(2L-1)\times D}$ is the output matrix of pooled hidden state crosses. The computational complexity about the sequence length is reduced from $\mathcal{O}(L^2)$ to $\mathcal{O}(L \log L)$ by FFT.

3.2.2 Central Token Symmetry

Formula 3 has the output matrix of shape $(2L-1) \times D$, which doubles the sequence length. To keep the computational complexity of attention not increasing, the length is needed to be reduced. As is shown in Figure 1, sum-pooling along antidiagonal produces a symmetric token combination (cross) about a central token. Specifically, in even-numbered antidiagonals, the central token is the $\frac{k}{2}$-th token. For instance, the token combinations of the 10th antidiagonal include token 5-5, 4-6, 3-7, and so on. In odd-numbered antidiagonals, the symmetric center is between $\lfloor \frac{k}{2} \rfloor$-th token and $\lceil \frac{k}{2} \rceil$-th token. For instance, the 11th antidiagonal includes the token combinations of token 5-6, 4-7, 3-8, etc. Therefore, we can reduce the length by merging the consecutive even-numbered and odd-numbered antidiagonals so that token combinations of near symmetric centers are together. In the previous example, token combinations of the 10th antidiagonal and the 11th antidiagonal are summed up. Besides, token combinations of two same tokens are subtracted, e.g. token combination 5-5. Formally,

$$\mathbf{C} = LN(\mathbf{C}_1 + \mathbf{C}_2 - f_1(\mathbf{X}) \odot f_2(\mathbf{X})) \quad (4)$$

where $\mathbf{C}_1 \in \mathbb{R}^{L\times D}$ (padding a row of zero to align its the length to $\mathbf{C}_2$) and $\mathbf{C}_2 \in \mathbb{R}^{L\times D}$ are the odd-numbered and even-numbered rows of matrix $\mathbf{C}_0$ respectively, LN denotes layer normalization which ensures stable training, $\mathbf{C} \in \mathbb{R}^{L\times D}$ is the output.

3.3 Revise Self-attention

In this section, we revise the multi-head self-attention to utilize our pooled hidden state cross. The output of an attention layer is calculated as follows.

$$\mathcal{A}(\mathbf{X}, \mathbf{C}) = (\prod_{h=1}^{H} \Psi(\frac{\mathbf{Q}^h \mathbf{K}^{h^T}}{\sqrt{d}})\mathbf{V}^h)\mathbf{W}_o \quad (5)$$

where d is the dimension of a single head, superscript h denotes the h-th head, $\prod$ denotes the concatenation operation of H heads along the last dimension, Ψ is a row-wise scoring function (e.g. softmax), $\mathbf{W}_o \in \mathbb{R}^{Hd\times D}$ is the output projection matrix. It's worth noting that the key, and value in attention mechanism are revised.

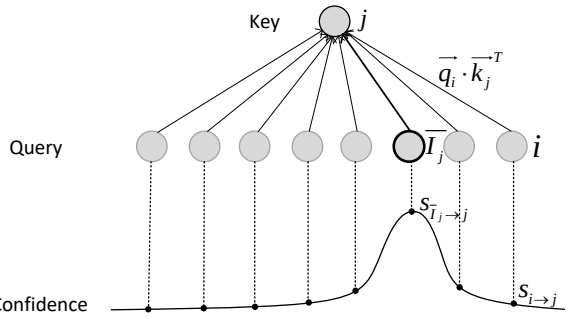

Figure 2: An illustration of the weighted directed graph with a single key vertex, and its in-neighbors. The edge pointing from the bold dominant query vertex to the key vertex corresponds to a dominant attention element in the attention matrix. Decreasing confidences are assigned to the edges away from the dominant query vertex.

$$\mathbf{Q}^h = \mathbf{X}\mathbf{W}_Q^h, \quad \mathbf{K}^h = \mathbf{C}\mathbf{W}_K^h, \quad \mathbf{V}^h = \mathbf{C}\mathbf{W}_V^h \quad (6)$$

where $\mathbf{W}_Q^h, \mathbf{W}_K^h, \mathbf{W}_V^h \in \mathbb{R}^{D\times d}$ are the learnable projection matrices.

4 Predictable Sparse Attention

Due to the sparsity of attention matrix, most elements of the attention matrix are close to zero, for the sake of simplicity, we call those elements, which are much greater than zero, dominant elements. Base on the pooled hidden state cross, we propose predictable sparse attention, which predicts dominant elements of attention matrix, to avoid computing the full attention matrix.

4.1 Sparse Attention

In this section, we describe the predictable sparse attention by a weighted directed sparse graph, in which the vertexes are the L query/key vectors of the input sequence, its directed edges represent that the head vertex attends to the tail vertex in the attention mechanism, and two weights (i.e. attention score and confidence) are assigned to each edge. Figure 2 illustrates a sub-graph with a single key vertex and its in-neighbors. The attention matrix is the adjacency matrix of the graph. For the multi-head attention, each head has a graph computed independently. The i-th output vector of the pro-

posed predictable sparse attention is defined as

$$\mathcal{A}(\mathbf{X}, \mathbf{C})_i = (\prod_{h=1}^{H} (\Psi(\frac{\vec{q}_i^h \mathbf{K}_{N_i^h}^{h^T}}{\sqrt{d}}) \odot \vec{s}_{i \to N_i^h}^h) \mathbf{V}_{N_i^h}^h) \mathbf{W}_o \tag{7}$$

where row vector $\vec{q}^h$, and matrices $\mathbf{K}^h, \mathbf{V}^h \in \mathbb{R}^{L \times d}$ are respectively the query, key, and value projected from $\mathbf{X}$ or $\mathbf{C}$ following Formula 6, N_i^h represents the out-neighbors set of the i-th vertex in the h-th head's directed graph, when N_i^h is written at subscript, it means only extracting the matrix's rows corresponding to the vertexes in N_i^h, and $\vec{s}_{i \to N_i^h}^h$ represents the confidence vector consisting of the confidence scores of the edges pointing from i-th vertex to the vertexes in N_i^h in the h-th head's directed graph.

4.2 Sparse Attention Matrix Estimation

The challenge of the predictable sparse attention is to find out which elements in the attention matrix are dominant under the condition of not computing the full attention matrix. We utilize pooled hidden state cross, because semantic combination vectors contain the information of the relevance of token pairs.

4.2.1 Attention Confidence

We introduce attention confidence to help the model learning the sparse structure of an attention matrix. Specifically, we define the confidence of the i-th query vector $\vec{q}_i$ attending to the j-th key vector $\vec{k}_j$ as

$$s_{i \to j} = \rho(i \,|\, \bar{I}_j, \sigma^2) \tag{8}$$

in which, ρ denotes the probability density function of Gaussian distribution, $\bar{I}_j$ is the index of the dominant query vector, which attends to the key vector $\vec{k}_j$ with a dominant attention score (i.e. edge $\bar{I}_j \to j$ corresponds to a dominant element in the attention matrix), σ^2 is a hyper-parameter representing the variance. We have this definition because of the observation that the query vectors far away from the dominant query vector have decreasing probabilities of attending to the key vector.

4.2.2 Reparameterization

The key to make a sparse attention matrix predictable is how to back-propagate gradients through the predicted discrete indices. In this paper, discrete indices are reparameterized. A dominant index matrix $\bar{\mathbf{I}} \in \mathbb{R}^{L \times M}$ is predicted based on the pooled hidden state cross $\mathbf{C}$:

$$\bar{\mathbf{I}} = \sigma(\mathbf{C}\mathbf{W}_I + \vec{b}_I) \cdot L_{max} \tag{9}$$

where $\mathbf{W}_I \in \mathbb{R}^{D \times M}$ and $\vec{b}_I \in \mathbb{R}^M$ are the learnable weight and bias respectively, $\sigma(\cdot)$ is the sigmoid function, L_{max} is the maximum sequence length that the model supports. Since there may be multiple dominant query vectors for a key vector, the hyper-parameter M presumes the maximum number of dominant query vectors for a single key vector.

Given the sparse graph described by an index matrix $\mathbf{I} \in \mathbb{N}^{L \times M}$, whose value in the j-th row m-th column I_{jm} indicates that there is a directed edge pointing from the I_{jm}-th query vector to the j-th key vector, the confidence score of each edge can be calculated as follows.

$$s_{I_{jm} \to j} = \rho(I_{jm} | \bar{I}_{jm}, \sigma^2) \tag{10}$$

where $\bar{I}_{jm}$ is the m-th predicted dominant index of the j-th key vector predicted by Formula 9. Therefore, applying the chain rule, the gradient of confidence scores from a loss function can continue to be propagated through Formula 9 and Formula 10 to matrix $\mathbf{C}$.

4.2.3 Learning Index

The index matrix $\mathbf{I} \in \mathbb{N}^{L \times M}$ decides which edges are considered and which edges are ignored in the sparse graph. In this paper, two types of index matrix are adopted, a predicted index matrix $\mathbf{I}_p = \lfloor \bar{\mathbf{I}} \rfloor$ and a random index matrix $\mathbf{I}_r \sim \mathcal{U}_{[0, N-1]}$. The process for learning sparse attention matrix can be viewed as a process for searching right indices, it is a process of exploring new knowledge and exploiting existing knowledge. Therefore, in training, the sparse graph is decided by the union of $\mathbf{I}_p$ (exploitation) and $\mathbf{I}_r$ (exploration), and, in inference, only $\mathbf{I}_p$ is used.

4.2.4 Gradient Truncation

When back-propagation, the gradient of attention confidence is truncated into the range $(-\infty, 0\,]$ for stable convergence. Because, for gradient descent algorithm, positive gradients will decrease the confidence values on edges, it means that the gradients prevent the model from considering these edges in sparse attention mechanism, and tune the model's parameters to change its predicted dominant-indices. But due to the discreteness of indices, changing the predicted dominant-indices

Models	ListOps	Text	Retrieval	Image	Pathfinder	Avg.	Avg. (w/o Text)
Transformer	37.45	64.96	78.38	43.19	74.61	59.72	58.41
Local Attention	15.82	52.98	70.65	41.46	66.63	49.51	48.64
Sparse Trans.	17.07	63.58	72.53	44.24	71.71	53.83	51.39
Longformer	35.63	62.85	68.32	42.22	69.71	55.75	53.97
Linformer	35.70	53.94	77.83	38.56	76.34	56.47	57.11
Reformer	37.27	56.10	73.03	38.07	68.50	54.59	54.22
Sinkhorn Trans.	33.67	61.20	65.88	41.23	67.45	53.89	52.06
Synthesizer	36.99	61.68	80.04	41.61	69.45	57.95	57.02
BigBird	36.05	64.02	76.41	40.83	74.87	58.44	57.04
Linear Trans.	16.13	65.90	72.09	42.34	75.30	54.35	51.47
Performer	18.01	65.40	75.43	42.77	77.05	55.73	53.32
Fnet	35.33	65.11	59.61	38.67	**77.80**	55.30	52.85
Nyström	37.15	65.52	79.56	41.58	70.94	58.95	57.31
Luna-256	37.25	64.57	79.29	47.38	77.72	61.24	60.41
FSAT (ours)	**46.85**	**65.95 / 80.24**	**81.11**	**49.97**	77.32	**64.24 / 67.10**	**63.81**

Table 1: Experimental results on the Long Range Arena benchmark. Except for Retrieval task, results of models from Local Attention to Performer are cited from Tay et al. (2020b). Fnet, Nyström, and Luna-256 are more recent works, results are from their papers. Average accuracy without the Text task is reported separately. For the Text task, the results of FSAT using different feature mapping (fully-connect structure or depth-wise separable convolution layer) are reported, see the discussion in the body text.

slightly larger or smaller does not ensure moving closer to the correct dominant indices. On the contrary, negative gradients indicate hitting the correct dominant-indices, and the model should be tuned using these gradients.

4.3 Complexity Analysis

In terms of computational complexity, the proposed predictable sparse attention has lower computational cost in time and memory usage. Specifically, the computational complexity for pooled hidden state cross includes the feature mapping $\mathcal{O}(LD^2)$, and fast Fourier transform $\mathcal{O}(LD \log L)$. The computational complexity involved in Formula 7 includes computing attention probability $\mathcal{O}(LMD)$, computing attention confidence $\mathcal{O}(LDM)$, and matrix multiplications about the value matrix and projection matrices $\mathcal{O}(LMD + LD^2)$. The overall computational complexity is $\mathcal{O}(LD^2 + LD \log L + LMD)$. Since M is always small (e.g. $M = 4$), for long sequences, this complexity is much smaller than $\mathcal{O}(L^2D + LD^2)$ which is the complexity of standard multi-head attention. In memory usage, sparse attention has no need to store the full attention matrix, thus the memory complexity is reduced from $\mathcal{O}(L^2H + LD)$ to $\mathcal{O}(LMH + LD)$.

5 Experiments

We conduct experiments to study the performance of the proposed approach on long sequence modeling tasks.

5.1 Long-context Sequence Modeling

As the primary goal, we evaluate the proposed Fourier Sparse Attention for Transformer (FSAT) on multiple tasks requiring long-context perception. We test our models on the Long Range Arena (LRA) benchmark (Tay et al., 2020b), since it is specifically designed for evaluating the performance of efficient Transformers on various long sequence tasks, and there are quite a number of baseline models evaluated on this benchmark.

5.2 Datasets and Baselines

The LRA benchmark includes five tasks of different kinds and modalities (natural language, vision, and math) in order to simulate meaningful real-world tasks under the long-context scenario.

- **ListOps** This task requires models to compute the output value of mathematical expression with a hierarchical structure and operators. The sequence lengths are up to 2K.
- **Text** A byte-level text classification task to probe the model's reasoning ability with compositional, unsegmented characters. Character

Model	Steps per second ↑				Peak Memory Usage ↓			
	1K	2K	3K	4K	1K	2K	3K	4K
Transformer	1.0	1.0	1.0	1.0	1.00	1.00	1.00	1.00
Local Attention	1.1	1.7	3.2	5.3	0.49	0.29	0.19	0.14
Linformer	1.2	1.9	3.7	5.5	0.44	0.21	0.18	0.1
Reformer	0.5	0.4	0.7	0.8	0.56	0.37	0.28	0.24
Sinkhorn Trans.	1.1	1.6	2.9	3.8	0.55	0.31	0.21	0.16
Synthesizer	1.1	1.2	2.9	1.4	0.76	0.75	0.74	0.74
BigBird	0.9	0.8	1.2	1.1	0.91	0.56	0.4	0.3
Linear Trans.	1.1	1.9	3.7	5.6	0.44	0.22	0.15	0.11
Performer	1.2	1.9	3.8	5.7	0.44	0.22	0.15	0.11
FSAT (ours)	1.1	1.5	2	2.5	0.53	0.27	0.21	0.16

Table 2: The time cost and memory consumption on the Long Range Arena benchmark on byte-level text classification with various input lengths (1K, 2K, 3K and 4K). The speed and memory consumption are shown through the rate with respect to the vanilla Transformer.

sequences are truncated or padded to a fixed maximum length of 4K in this task.

- **Retrieval** A byte-level document retrieval task tests model's ability to compress long sequences into representations suitable for similarity-based matching.
- **Image** An image classification task evaluates a model's performance of perceiving 2D spatial relations between input pixels. Images are flattened to sequences of length 1K pixels.
- **Pathfinder** A binary image classification task tests if a model can capture long-range spatial dependencies by judging if two points are connected by a path consisting of dashes in an image with distractor paths. 32×32 images are flattened to sequences of length 1K pixels.

We compare our model with a number of promising models, including vanilla Transformer(Vaswani et al., 2017), a local attention baseline, Sparse Transformer(Child et al., 2019), Longformer(Beltagy et al., 2020), Linformer(Wang et al., 2020), Reformer(Kitaev et al., 2020), Sinkhorn Transformer(Tay et al., 2020a), Synthesizer(Tay et al., 2021), Big Bird(Zaheer et al., 2020), Linear Transformer(Katharopoulos et al., 2020), Performer(Choromanski et al., 2020a,b), and more recent models, such as FNet(Lee-Thorp et al., 2021), Nsyträmformer(Xiong et al., 2021), and Luan(Ma et al., 2021).

5.3 Implementation details

We run our experiments on the LRA benchmark with the configurations based on Tay et al. (2020b) open source codebase. Specifically, we follow the original data preprocessing, data split, and keep roughly equivalent model parameters for a fair comparison with the baselines reported in Tay et al. (2020b). An exception is that we reproduce the experiments of the Retrieval task for a longer training of 30K steps because models are not fully converged in 5K training steps. Ma et al. (2021); Lee-Thorp et al. (2021); Xiong et al. (2021) also pointed the same issue. We also re-run the vanilla Transformer using our Pytorch implementation.

For the proposed FSAT, the default value of the hyper-parameter M is 4, and the variance σ^2 is empirically set to L_{max} which is the maximum sequence length of each task. To ensure roughly equivalent model parameters, we reduce the dimension of FFN layer from 4 times of the hidden size to 2 times for FSAT to offset the increased parameters of non-linear feature mapping functions (Formula 1). In our code, sparse matrix multiplications involved in FSAT model are implemented via scatter operations at batch-level for better efficiency. Median results of 5 runs are reported in the tables.

5.4 Results

Table 1 summarizes the performance of a number of models on the LRA benchmark. As we can see, the proposed model clearly outperforms all previously published approaches, and achieves new state-of-the-art performance on four of the five datasets, and a 5.8% absolute improvement over average performance, which validates the effectiveness of the proposed FSAT model. It is noteworthy that we separately report the average accuracy

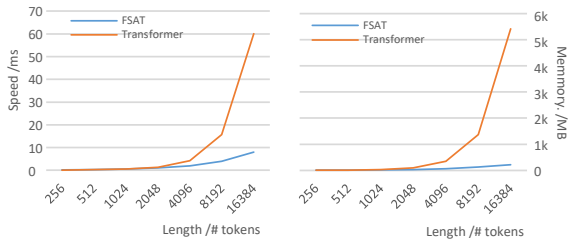

Figure 3: An illustration of the trend of time and memory cost, as the input sequence length increases. The average memory consumption (MB) and training time (ms) for a single sequence is counted.

without the Text task, this is because we find that convolution layers have a significant impact on this task. In the experiment, we adopt a depth-wise separable convolution layer with a kernel size of 5 as the feature mapping function when computing hidden crosses, the accuracy significantly increases from 65.95% to 80.24%. Therefore, considering the particularity of the Text task, we report its result separately. We suspect that the small data size may be the reason of its particularity.

5.5 Efficiency

The time and memory efficiency of our model and competing approaches are summarized in Table 2. Compared with the vanilla Transformer, our FSAT significantly reduces the computational complexity with faster training speed and lower memory usage, which demonstrates that directly predicting the sparse structure of attention matrix is an effective way for building efficient Transformer architecture. The limitation of FSAT is that extra operations (e.g. gathering, slicing) of linear complexity are involved, so that FSAT can not bring parallel superiority into full play. Even so, among all compared Transformer variants, FSAT achieves promising results in time and memory efficiency. The trend of computation complexity increasing is shown in Figure 3. This is in line with expectations, FSAT has a linear rate of increase, its advantage is especially obvious for sequences longer than 4K tokens. This demonstrates the potential of the proposed predictable sparse attention for the tasks with much longer sequences, e.g. 3D feature volume.

5.6 Ablation Study

An ablation study is conducted to verify the necessity of our proposed model components. In Table 3, we report FSAT models with the different number of predicted dominant indices. The results show

	ListOps	Retrieval	Image	Avg.
Transformer	37.45	78.38	43.19	53.01
FSAT-2	39.1	76.76	49.64	55.17
FSAT-4	46.85	81.11	49.97	59.31
- No Trunc.	42.65	74.88	47.74	55.09
- Only $\mathbf{I}_r$	37.54	73.84	40.24	50.54
- Only $\mathbf{I}_p$	17.8	56.81	21.58	32.06
+ DConv-5	45.95	81.45	32.23	53.21
FSAT-8	**47.95**	81.29	49.85	59.70
FAT	46.8	**82.46**	**50.14**	**59.77**

Table 3: Ablation study on three tasks of the LRA benchmark. “”/“+” denotes removing/adding a model component. The best model is in boldface.

that for each key vector in the attention mechanism about 4 predicted dominant query vectors are enough for the model to produce high accuracy. We also remove the sparse attention module, and test the architecture of only integrating the pooled hidden state cross into the attention mechanism, corresponding to Formula 5. We call this architecture Fourier Attention for Transformer (FAT). It can be seen from the results of FAT, better results can be obtained with the pooled hidden state cross in some tasks, which supports our hypothesis that the 2-order token feature may generate more expressive feature representations. It is noteworthy that without the gradient truncation, or only using random indices $\mathcal{I}_r$ or $\mathcal{I}_p$, the performance significantly drops. Besides, the depth-wise convolution based feature mapping performs worse than a fully-connected structure base feature mapping.

6 Conclusion

In this paper, we proposed a new efficient Transformer architecture, FSAT, which directly predicts the sparse structure of attention matrix to avoid computing the quadratic-cost full self-attention. The proposed approach has the advantages in long-range sequence modeling tasks meanwhile reaching a balance between time, memory, and accuracy. We showed the effectiveness of the proposed method in modeling long sequence context using the Long Range Arena benchmark. Experimental results showed that state-of-the-art performance is achieved by our proposed method.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv*

preprint arXiv:2004.05150.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer.

Lin Chen, Hossein Esfandiari, Gang Fu, Vahab S Mirrokni, and Qian Yu. 2021. Feature cross search via submodular optimization. *arXiv preprint arXiv:2107.02139*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, David Belanger, Lucy Colwell, et al. 2020a. Masked language modeling for proteins via linearly scalable long-context transformers. *arXiv preprint arXiv:2006.03555*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020b. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

Nazrul Hoque, Dhruba K Bhattacharyya, and Jugal K Kalita. 2014. Mifs-nd: A mutual information-based feature selection method. *Expert Systems with Applications*, 41(14):6371–6385.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Nojun Kwak and Chong-Ho Choi. 2002. Input feature selection by mutual information based on parzen window. *IEEE transactions on pattern analysis and machine intelligence*, 24(12):1667–1671.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2021. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. 2021. Luna: Linear unified nested attention. *arXiv preprint arXiv:2106.01540*.

Feiping Nie, Heng Huang, Xiao Cai, and Chris Ding. 2010. Efficient and robust feature selection via joint 2, 1-norms minimization. *Advances in neural information processing systems*, 23.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Monica Rogati and Yiming Yang. 2002. High-performing feature selection for text classification. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 659–661.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018. Bi-directional block self-attention for fast and memory-efficient sequence modeling. *arXiv preprint arXiv:1804.00857*.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3531–3539.

Takumi Takahashi, Takuji Tahara, Koki Nagatani, Yasuhide Miura, Tomoki Taniguchi, and Tomoko Ohkuma. 2018. Text and image synergy with feature

cross technique for gender identification. *Working Notes Papers of the CLEF*.

Alex Tamkin, Dan Jurafsky, , and Noah Goodman. 2020. Language through a prism: A spectral approach for multiscale language representations. *arXiv preprint arXiv:2011.04823*.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2021. Synthesizer: Rethinking self-attention for transformer models. In *International Conference on Machine Learning*, pages 10183–10192. PMLR.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020a. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020b. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020c. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. 2015. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963. PMLR.

Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. 2000. Feature selection for svms.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nystr\" omformer: A nystr\" om-based algorithm for approximating self-attention. *arXiv preprint arXiv:2102.03902*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Sepehr Abbasi Zadeh, Mehrdad Ghadiri, Vahab Mirrokni, and Morteza Zadimoghaddam. 2017. Scalable feature selection via distributed diversity maximization. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.