ERNIE-SPARSE: Learning Hierarchical Efficient Transformer Through Regularized Self-Attention

Anonymous ACL submission

Abstract

Sparse Transformer has recently attracted a lot of attention since the ability for reducing the quadratic dependency on the sequence length. We argue that two factors, *information bottleneck sensitivity* and *inconsistency between different attention topologies*, could affect the performance of the Sparse Transformer. This paper proposes a well-designed model named ERNIE-SPARSE. It consists of two distinctive parts: (i) **H**ierarchical **S**parse **T**ransformer (**HST**) to sequentially unify local and global information. (ii) **S**elf-**A**ttention **R**egularization (**SAR**) method, a novel regularization designed to minimize the distance for transformers with different attention topologies. To evaluate the effectiveness of ERNIE-SPARSE, we perform extensive evaluations. Firstly, we perform experiments on a multi-modal long sequence modeling task benchmark, Long Range Arena (LRA). Experimental results demonstrate that ERNIE-SPARSE significantly outperforms a variety of strong baseline methods including the dense attention and other efficient sparse attention methods and achieves improvements by 2.77% (57.78% vs. 55.01%). Secondly, to further show the effectiveness of our method, we pretrain ERNIE-SPARSE and verified it on 3 text classification and 2 QA downstream tasks, achieve improvements on classification benchmark by 0.83% (92.46% vs. 91.63%), on QA benchmark by 3.24% (74.67% vs. 71.43%). Experimental results continue to demonstrate its superior performance.

1 Introduction

Transformer (Vaswani et al., 2017) architecture is a key component for many pretrained language models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019), ERNIE (Sun et al., 2020b), ALBERT (Lan et al., 2019), ELECTRA(Clark et al., 2020), T5 (Raffel et al., 2020). Self-attention is one of the most important modules in transformer. It eliminates the sequential dependency constraints of recurrent neural networks by introducing interactions between each token pair to capture contextual information. However, the self-attention's computational complexity and memory cost grow quadratically with sequence length, which comes with complexity $O(N^2)$ for processing contexts of N inputs.

One way to optimize self-attention complexity is introducing sparsity into attention layers (Child et al., 2019; Qiu et al., 2019; Beltagy et al., 2020) by having each token attend to only a subset of tokens in the whole sequence. Recent sparse-attention works (Child et al., 2019; Beltagy et al., 2020; Ainslie et al., 2020; Zaheer et al., 2020) introduce global tokens that can attend to the whole sequence. Those global tokens are used as a form of memory to strengthen global information. While this method reduces the complexity of full self-attention, there are two issues with Sparse Transformer that affect performance.

The first issue is *information bottleneck sensitivity* as shown in Figure 1. Information bottleneck is a phenomenon caused by the low number of global tokens — the model had to encapsulate the entire input sequence into those global tokens. If the size of the information bottleneck becomes smaller, performance can suffer. The second issue is *inconsistency between different attention topologies*. Normally, a sparse pattern is pre-defined and fixed in practical training. Due to that sparse pattern splitting the input into several blocks and interaction for input is partially connected, minor changes, e.g., shifting the attention input by a few positions, will lead to the inconsistency in the attention topology, as shown in Figure 2. Compared to the effect for vanilla attention, the sparse attention mechanism will amplify the impact of the minor change.

To resolve the aforementioned issues, we propose a well-designed efficient model ERNIE-SPARSE. Our method proposes two major techniques including **H**ierarchical **S**parse **T**ransformer

(**HST**) and **S**elf-**A**ttention **R**egularization (**SAR**). The first technique HST is designed for *information bottleneck sensitivity* problem based on hierarchical attention mechanism. HST introduces special tokens to represent local information and global information in two sequential steps, providing an additional way for modeling global information. The second technique SAR, is designed for *inconsistency between different attention topologies*. We introduce a training regularization strategy to force models with different sparse attention topologies to have consistent outputs.

To evaluate ERNIE-SPARSE's ability to model long documents, we first perform extensive evaluations on a long sequence modeling task benchmark (Tay et al., 2020c): Long Range Arena (LRA). Experiment results demonstrate that our model significantly outperforms existing methods and provides a new robust solution to the long range sequence modeling. Our proposed method achieves improvements by average 2.77 points (57.78% vs. 55.01%) on five long sequence tasks of LRA. We also conduct experiments on 3 long document classification and 2 question answering tasks using pretraining and finetuning paradigm, further shows the effectiveness of ERNIE-SPARSE. Extensive experiments on those 10 tasks demonstrate the effectiveness of our approach in both cold start and pretrain-then-finetune paradigm.

2 Related Work

Sparse Transformer Sparse attention is widely adopted to solve the long range sequence modeling problem. A simple version is that split the sequence into blocks and perform attention only within block (Qiu et al., 2019; Liu et al., 2018; Parmar et al., 2018). This mechanism is also called local attention because only local tokens within the block can attend to each other. To improve the connection between tokens, global attention is introduced by (Child et al., 2019; Beltagy et al., 2020; Ainslie et al., 2020; Zaheer et al., 2020). The global attention mechanism mainly relies on specifying some tokens as global tokens. Those global tokens are used as a form of memory to strengthen global information. However, we observed that the mechanism of global and local attention in sparse attention would cause an information bottleneck, which would affect the information flow between local tokens. This phenomenon was also observed in paper (Alon and Yahav, 2021) for Graph Neural Network (Gori et al., 2005). To this end, we propose to use hierarchical mechanism which will be discussed below.

Hierarchical Transformer Hierarchical learning has been suggested by many researchers and it has been empirically shown to be effective in numerous diverse tasks of natural language processing (Zhang et al., 2019; Liu and Lapata, 2019; Rohde et al., 2021; Wu et al., 2021). In this paper, we propose to apply a hierarchical mechanism in Sparse Transformer and provide a new perspective from information flow to describe its beneficiary for Sparse Transformer.

Regularization Method Although state-of-the-art deep neural networks have achieved high performance on various NLP tasks, the architectures used for these tasks have been shown to be unstable to small modifications of input texts (Ebrahimi et al., 2018; Jin et al., 2020; Sun et al., 2020a). Adversarial training (Szegedy et al., 2014; Goodfellow et al., 2015; Miyato et al., 2019; Liu et al., 2020) is proposed to to act as a regularization method to improve model performance by constructing adversarial input. However, current regularization methods are mostly designed for dense models and don't take into account the characteristic of sparse model. Recently, (Liang et al., 2021) proposed a simple regularization strategy which forces the output distributions of different sub models generated by dropout to be consistent with each other. Inspired by this work, we propose a regularization method built upon sparse attention topology.

3 Method

In this section, we first revisit the mechanism of Sparse Transformer in Section 3.1, and describe the two components of sparse attention (i.e. global attention and local attention). In addition, we provide a new perspective for understanding Sparse Transformers from information flow. At the end of Section 3.1, we discuss the problems existing in the mechanism of the Sparse Transformer. Next, we introduce the motivation, formulation and benefits of the HST in Section 3.2. Finally, a regularization method, SAR, is illustrated in Section 3.3.

3.1 Revisiting Sparse Transformer

Given a sequence of input $x = (t_1, ..., t_n)$ of length n, dense attention can be formulated as:

$$\begin{pmatrix} \mathbf{Q} \\ \mathbf{K} \\ \mathbf{V} \end{pmatrix} = \mathbf{H} \begin{pmatrix} \mathbf{W}_\mathbf{q} \\ \mathbf{W}_\mathbf{k} \\ \mathbf{W}_\mathrm{v} \end{pmatrix} + \begin{pmatrix} \mathbf{b}_\mathbf{q} \\ \mathbf{b}_\mathrm{k} \\ \mathbf{b}_\mathrm{v} \end{pmatrix},$$
$$\text{Attention} = \text{Softmax}\left(\mathbf{Q}\mathbf{K}^T\right)\mathbf{V},$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n\times d}$ is linearly mapped from $\mathbf{H} \in \mathbb{R}^{n\times d}$, d is the size of the hidden vector, $\mathbf{H} = (h_1^l, ..., h_n^l)$ for layer $l \in [1, L]$, L is the number of model layers, $\mathbf{W}_{\{\mathbf{q},\mathbf{k},\mathbf{v}\}}$ are mapping weights, and $\mathbf{b}_{\{\mathbf{q},\mathbf{k},\mathbf{v}\}}$ are bias terms. For simplicity, the scale factor for $\mathbf{Q}$ is omitted in the equation. Computation complexity for dense transformer is $O(n^2)$. In dense attention, tokens are connected to each other while in sparse attention, tokens are only partially connected.

As shown in Figure 1 (a), sparse attention mainly consists of global attention and local attention (Liu et al., 2018; Parmar et al., 2018; Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020). In this Figure, $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are divided into 4 blocks respectively, (t_1, $t_2 \sim t_3$, $t_4 \sim t_5$, $t_6 \sim t_7$). The light blue part denotes the global attention meaning the global tokens from this part can attend to all other tokens. The dark blue part expresses local attention denoting that the local tokens can only attend to the tokens within the same block. It can be seen from Figure 1 (b) that there is a bottleneck in the information flow of Sparse Transformer. For sparse attention mechanism, tokens in different attention blocks within the same layer are invisible to each other. For example, t_3 can't attend to t_4 and vice versa. For these tokens, the only way to attend to each other is through global tokens as relay nodes. As shown in Figure 1 (b), the information of t_3 and t_4 first flows to t_1 at layer l, and then distributes back to t_3 and t_4 at layer $l + 1$. This mechanism causes the current layer local token information to be carried by only a few global tokens leading to a bottleneck in the flow of information. According to our observation in section 5.1, with the information bottleneck size decreases, the performance has a downward trend.

Another problem caused by sparse attention mechanism is inconsistency between the sparse attention topology change. As shown in Figure 2, we plot the attention topology of layer l for $[t_1, t_2, \cdots, t_8]$ in left figure, and for $[t_2, \cdots, t_8, t_1]$ in right. We see that even only t_1 is shifted by one place, the topology pattern changes much, where dense attention will be less affected as it has a fully connected attention topology. To this end, we propose HST and SAR and will further describe them below.

3.2 HST: Hierarchical Sparse Transformer

HST mainly contains three key elements, i.e. representative tokens insertion, hierarchical attention and the usage for those representative tokens in the last layer for downstream task.

As aforementioned, the input sample consists of n tokens $(t_1, t_2, ..., t_n)$. We set g as the number of global tokens and w as the block size meaning the number of tokens for each local attention. We propose to insert m representative tokens into input sequence, where $m = (n-g)/w$. Those tokens are inserted to the start position of each block. Similar to BERT (Devlin et al., 2018), we use [CLS] for those representative tokens. Thus the encoder input is as follows:

$$\mathbf{H}^0 = \underbrace{\mathbf{E}(t_1); \cdots; \mathbf{E}(t_g);}_{global}$$
$$\cdots, \underbrace{\mathbf{E}(r_i); \mathbf{E}(t_{g+w(i-1)+1}); \cdots; \mathbf{E}(t_{g+wi})}_{i\text{-th local}}, \cdots \tag{1}$$

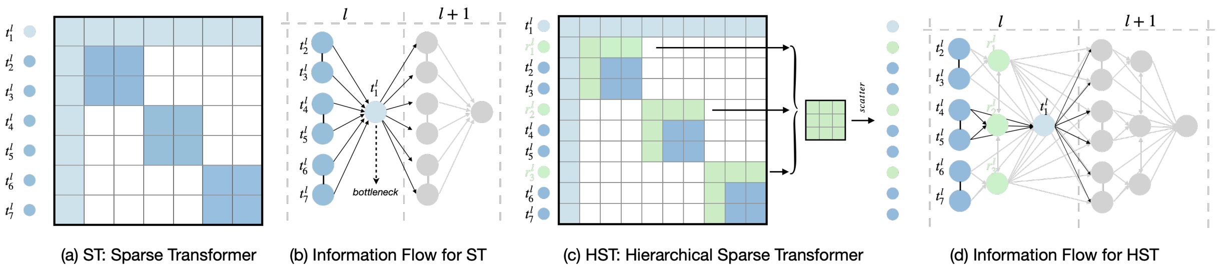
where $i = [1, 2, \cdots, m]$, $\mathbf{E}(t) \in \mathbb{R}^d$ is the embedding lookup for token t. $\mathbf{E}(r) \in \mathbb{R}^d$ is the representation token embedding. Then we use Sparse Transformer to encode the sequence as follows:

$$\mathbf{H}_s^l = \text{SparseTransformer}\left(\left[\mathbf{H}^{l-1}\right]\right),$$

where $\mathbf{H}_s^l \in \mathbb{R}^{n\times d}$ is the hidden output from one layer Sparse Transformer. To better interact globally, representative tokens are extracted from $\mathbf{H}_s^l$ for dense attention as shown in Figure 1 (c). Formally, the hierarchical attention is calculated by:

$$\mathbf{R}_s^l = \left(r_1^l \cdots r_m^l\right),$$
$$\mathbf{R}^l = \text{Attention}\left(\mathbf{R}_s^l\right),$$

where $\mathbf{R}_s^l \in \mathbb{R}^{m\times d}$ is the matrix of representative token's hidden states for layer l, $r_1^l \cdots r_m^l$ are extracted from $\mathbf{H}_s^l$. Attention is the dense attention described in 3.1. After dense attention, these representative token hidden vectors in $\mathbf{R}^l$ are distributed back to $\mathbf{H}_s^l$ so we get the final hidden vectors $\mathbf{H}^l$ of l-th layer. The whole process can be seen more clearly in Figure 1 (c), after hierarchical attention (green matrix), the green dots (denoting representative tokens) are distributed back to the list of tokens.



(a) ST: Sparse Transformer
(b) Information Flow for ST
bottleneck
(c) HST: Hierarchical Sparse Transformer
scatter
(d) Information Flow for HST

Figure 1: **The comparison between Sparse Transformer (ST) and Hierarchical Sparse Transformer (HST). (a)** Sparse Transformer mainly consists of global attention and local attention. **(b)** The bottleneck that existed in ST: all the sequence information is compressed into a fixed size vector. **(c)** In HST, representative tokens are inserted into local attention for hierarchical attention. **(d)** Information flow demonstration for HST: interaction between representative nodes can increase the path of global information interaction.

The information flow of the hierarchical attention is shown in Figure 1 (d) showing that richer global information interaction path are created. For example, t_3 and t_5 can complete an interaction by representative nodes in addition to global tokens.

Note that the $\mathrm{Attention}$ module will introduce additional weights $\mathbf{W_q}$, $\mathbf{W_k}$ and $\mathbf{W_v}$ mentioned in Section 3.1, so how to initialize these three weights needs to be discussed. One option is to randomly initialize these three weights, or we can use the weight of the $\mathrm{SparseTransformer}$ to warm start, or we should also consider whether to share the weights in $\mathrm{Attention}$ with $\mathrm{SparseTransformer}$. Those details will be discussed in Experiment section.

For the downstream task, it is efficient to use the representative tokens as follow:

$$\mathbf{O}^L = \mathrm{Pooling}\left(\mathbf{R}^L\right),$$
$$\mathcal{P}\left(y \mid x\right) = \mathrm{Softmax}(\mathbf{O}^L \mathbf{W}_o),$$

where $\mathrm{Pooling}$ can be MAX, MEAN, $\mathbf{O}^L \in \mathbb{R}^d$, $\mathbf{W}_o \in \mathbb{R}^{d \times c}$, c is the downstream task class number. y is the downstream task label, $\mathcal{P}\left(y \mid x\right)$ denotes the predicted probability. Note that $\mathrm{Pooling}$, can be replaced as CLS_g, where CLS_g is the first global token.

3.3 SAR: Self-Attention Regularization

Different with dense attention, sparse attention doesn't guarantee interaction between all tokens as the design of sparse attention is to allow tokens can only attend to some of other tokens in one layer. This leads to a problem of inconsistency: even shifting by one place to attention input will cause much sparse topology change, as shown in Figure 2 and discussed in detail in Section 3.1. To this end, we introduce a regularization method to leverage the outputs of Sparse Transformer.

Our method is designed based on the motivation that for the same input, the outputs of transformers with different attention topologies should be consistent. As shown in Figure 2, the left shows a L layers Sparse Transformer with the default sparse attention topology, and the right shows a transformer with a shifted attention topology. Concretely, given the input data x at each training step, we feed x to go through the forward pass of the network twice, one with default sparse attention topology, and the other with the shifted attention topology. Therefore, we can obtain two distributions of the model predictions, denoted $\mathcal{P}_1\left(y \mid x\right)$ and $\mathcal{P}_2\left(y \mid x\right)$. Thus the distributions of $\mathcal{P}_1\left(y \mid x\right)$ and $\mathcal{P}_2\left(y \mid x\right)$ are different for the same input data pair (x, y). Then at this training step, in order for those two distributions to be close to each other, our SAR method tries to minimize the bidirectional Kullback-Leibler (KL) divergence between these two output distributions. Formally speaking,

$$\begin{aligned}\mathcal{L}_{SAR} = &\frac{1}{2}\left[\mathcal{D}_{KL}\left(\mathcal{P}_1(y \mid x) \| \mathcal{P}_2\left(y \mid x\right)\right)\right. \\ &\left.+\mathcal{D}_{KL}\left(\mathcal{P}_2\left(y \mid x\right) \| \mathcal{P}_1(y \mid x)\right)\right]\end{aligned}$$

Assume that the learning objective of target task is negative log-likelihood, the loss of the two forward passes is:

$$\mathcal{L}_{NLL} = -\log \mathcal{P}_1\left(y \mid x\right) - \log \mathcal{P}_2\left(y \mid x\right),$$

as a result, the total loss should be :

$$\mathcal{L} = \mathcal{L}_{NLL} + \alpha \mathcal{L}_{SAR},$$

where α is the loss coefficient of $\mathcal{L}_{SAR}$. Our method naturally makes transformer outputs of different attention topologies close to each other.

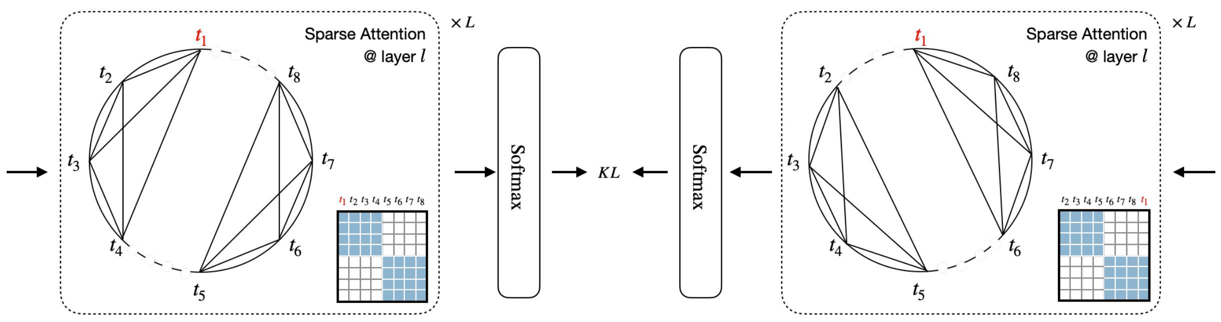


Figure 2: **The overall framework of our proposed SAR**. The procedure of SAR that regularizes the model outputs of transformers with different attention topologies.

4 Experiments

To evaluate our approach and show its performance, we first conduct experiments on a long context sequence modeling benchmark, LRA, which is consisted of 5 multi-modal tasks including logical inference, natural language and image tasks. LRA is a benchmark for cold start models and does not require the model to be pre-trained, so LRA is a suitable benchmark for testing the model structure designs. To further evaluate the ability of ERNIE-SPARSE in the pretrain-then-finetune paradigm, we also follow (Beltagy et al., 2020) and (Zaheer et al., 2020) to pretrain ERNIE-SPARSE and test the pretrained ERNIE-SPARSE on 3 natural language classification and 2 question answering tasks.

4.1 Long-Context Sequence Modeling

We first evaluate the effectiveness and efficiency of ERNIE-SPARSE on the LRA benchmark recently introduced by (Tay et al., 2021), which is designed to evaluate efficient transformer models under the long-context scenario. They collect five tasks in this benchmark which are ListOps (Nangia and Bowman, 2018), byte-level text classification (Maas et al., 2011), byte-level document retrieval (Radev et al., 2013), image classification on sequences of pixels (Krizhevsky et al., 2009) and Pathfinder (Linsley et al., 2018). This benchmark consists of sequences ranging from 1K to 16K tokens, encompassing a wide range of data types and modalities such as text, natural, synthetic images, and mathematical expressions requiring similarity, structural, and visual-spatial reasoning. We run each experiment five times with different random seeds and report the average accuracy.

The result of ERNIE-SPARSE on the LRA tasks are reported in Table 1. First, we note that ERNIE-SPARSE achieves strong results on all tasks consistently compared to the transformer model and significantly outperforms all the other baseline methods and achieve best score in terms of the average accuracy. By taking a closer look at the accuracy for each task, ERNIE-SPARSE wins over baseline models on four out of five tasks. Notably, ERNIE-SPARSE can work well on both image and text and the math inference data sets.

4.2 Pretraining and Finetuning

4.2.1 Pretraining

In the training task, we follow (Liu et al., 2019) and pretrain ERNIE-SPARSE using the Mask Language Model (MLM) training object. This task involves predicting tokens that are randomly masked out. For the training samples, we train ERNIE-SPARSE with maximum sequence length of 4096 and train it for 1 million steps. Samples with sequence length less than 4096 will be concatenated as one sample to improve training efficiency. For those samples longer than max sequence length, we truncate them to 4096. Statistics of pretraining data can be found in A.1. Following (Beltagy et al., 2020), ERNIE-SPARSE warm-starts from the public RoBERTa checkpoint and we compare the performance in MLM task in terms of bits per character (BPC). As shown in Table 2, BigBird, Longformer, ERNIE-SPARSE are all better than RoBERTa whose max sequence length is 512. Among those methods, ERNIE-SPARSE performs best.

4.2.2 Text Classification

To test ERNIE-SPARSE on downstream tasks, we first select three text classification tasks: Arxiv paper categories classification (He et al., 2019), IMDB reviews classification (Maas et al., 2011) and Hyperpartisan news detection (Kiesel et al., 2019). Arxiv dataset consists of paper text con-

Models	ListOps	Text	Retrieval	Image	Pathfinder	Avg
Local Attention	15.82	52.98	53.39	41.46	66.63	46.06
Linear Trans. (Katharopoulos et al., 2020)	16.13	**65.90**	53.09	42.34	75.30	50.55
Reformer (Kitaev et al., 2020)	37.27	56.10	53.40	38.07	68.50	50.67
Sparse Trans. (Child et al., 2019)	17.07	63.58	59.59	44.24	71.71	51.24
Sinkhorn Trans.(Tay et al., 2020b)	33.67	61.20	53.83	41.23	67.45	51.39
Linformer (Wang et al., 2020)	35.70	53.94	52.27	38.56	76.34	51.36
Performer (Choromanski et al., 2021)	18.01	65.40	53.82	42.77	77.05	51.41
Synthesizer (Tay et al., 2020a)	36.99	61.68	54.67	41.61	69.45	52.88
Longformer (Beltagy et al., 2020)	35.63	62.85	56.89	42.22	69.71	53.46
Transformer (Vaswani et al., 2017)	36.37	64.27	57.46	42.44	71.40	54.39
BigBird (Zaheer et al., 2020)	36.05	64.02	59.29	40.83	74.87	55.01
ERNIE-SPARSE	**37.75**	64.47	**62.64**	**45.28**	**78.77**	**57.78**

Table 1: Experimental results on the long range arena (LRA) benchmark. The highest acc for each dataset is highlighted in bold and the second place is underlined.

Setting	BPC ↓
RoBERTa (Liu et al., 2019)	1.85
Longformer (Beltagy et al., 2020)	1.71
BigBird (Zaheer et al., 2020)	1.68
RoBERTa$_{reproduce}$	2.05
ERNIE-SPARSE	**1.67**

Table 2: MLM BPC for ERNIE-SPARSE and other models.

Models	Arxiv [2]	IMDB	Hyp	
#Example	30043	25000	645	/
#Classed	11	2	2	
Len. at P_{50}	14733	215	516	
Len. at P_{95}	43951	748	1990	
Metric	**F1**	**F1**	**F1**	**Avg**
RoBERTa	86.86	95.00	87.80	89.88
Longformer	87.65	95.49	87.19	90.11
BigBird	87.50	95.20	92.20	91.63
ERNIE-SPARSE	**89.05**	**95.53**	**92.81**	**92.46**

Table 3: Performance of various models on development set of benchmark natural language understanding tasks. Len. at P_k means the k-th percentile example length in the corresponding dataset.

tent collected from https://arxiv.org/ and there are 11 classes denoting the paper category. IMDB is a widely used sentiment analysis dataset containing 25,000 movie reviews labeled as positive or negative. Hyperpartisan news dataset contains document for political standpoint classification. To align the experimental settings, we used the dataset split published by (Beltagy et al., 2020) [1]. The experiment was repeated 5 times for both datasets. ERNIE-SPARSE's hyperparameters are recorded in the appendix A.2.1. Note that all linear transformation weights of hierarchical attention are shared with the weights of the previous Sparse Transformer attention. Table 3 summarizes the results of ERNIE-SPARSE. From this table, it shows that both ERNIE-SPARSE and BigBird performs better than limited length RoBERTa, with ERNIE-SPARSE performs the best. For Arxiv, ERNIE-SPARSE surpasses baseline by a large margin. For IMDB and Hyperpartisan, the performance gain continues demonstrating that ERNIE-SPARSE is capable of utilizing information from long document input.

4.2.3 Question Answering

For QA tasks, we choose WikiHop (Welbl et al., 2018) and TriviaQA (Joshi et al., 2017). WikiHop is a large scale QA dataset which has over 40K samples. It encourages models for text understanding across multiple documents. Specifically, WikiHop provides several contexts and questions and ask the model to select the best answer from candidates. Meanwhile, TriviaQA is a bigger scale QA dataset that contains over 650K question-answer pairs. It requires the model to identify the answer span given a context and a question. The dataset is distantly supervised, which means that there is

[1]https://github.com/allenai/longformer/blob/master/scripts/hp-splits.json

[2]The reported numbers from (Zaheer et al., 2020) on Arxiv dataset are not comparable with ours because they did not release the train/test split of the data. So we re-tested RoBERTa and BigBird's open source checkpoint on our Arxiv train/test split. We open source our Arxiv dataset split: https://github.com/anonymous

Models	WikiHop	TriviaQA		
#Example	43738	110647		/
Len. at P_{50}	1313	4576		
Len. at P_{95}	3685	5166		
Metric	**Acc**	**F1**	**EM**	**Avg**
RoBERTa	72.40	74.30	-	-
Longformer	75.18	74.53	67.00	72.23
ETC	73.70	-	-	-
BigBird	72.30	73.40	68.60	71.43
ERNIE-SPARSE	**75.76**	**76.47**	**71.80**	**74.67**

Table 4: Model comparison for WikiHop and TriviaQA.

Setting	Image	Text	Avg
	Acc	Acc	
MEAN	45.28	63.19	54.23
MAX	**46.51**	**63.45**	**54.98**
CLS_g-Only	42.88	61.58	52.23

Table 5: Ablation for the pooling method of representative tokens in ERNIE-SPARSE for downstream tasks.

no golden span, thus we find all identical answers in provided documents. From Table 4, we can see that the length at 50 percentile is over 1K and 4K for WikiHop and TriviaQa respectively, denoting that those two datasets consist of very long context.

In ERNIE-SPARSE model, following (Beltagy et al., 2020), we concatenate the answer / question / candidates with special tokens along with the context. The results of WikiHop and TriviaQA are shown in Table 4. For those two datasets, we record the reprodueced results of (Beltagy et al., 2020) and (Zaheer et al., 2020) and the score of ERNIE-SPARSE in the bottom row. From this table, we see that ERNIE-SPARSE achieves the best results over all those methods.

4.3 Ablation

Effect of pooling method for representative tokens In the experiment, we have explored three different pooling methods while keeping other settings unchanged. The results are shown in Table 5. MEAN and MAX represent mean pooling and max pooling, respectively. CLS_g-Only refers to use the first token CLS only. As presented in Table 5, CLS_g-Only is the worst on average score, and MEAN performs on par with MAX. Take a closer look at the score each task, it shows that MAX performs best at both Image and Text, indicating that MAX is a stable method for predicting strong results and for image tasks, meanwhile the MEAN can be considered as a candidate for hyper-parameters. For CLS, the score drops significantly at both tasks, indicating that the contextual global information is more important for image tasks.

Effect of proposed components: HST and SAR Table 6 shows the performance of ERNIE-SPARSE by ablating each proposed component. All models are reported with mean results of 5 runs, the hyperparameters keep the same as the official recommendation. From the last column in Table 6, we see that the HST improves ERNIE-SPARSE with 1.0 percent point on average (#1 vs. #0). As the default setting is weight not sharing for linear mapping weight in hierarchical attention with the sparse attention layer, we add a group of experiment #3 to explore the effect of weight sharing on experimental results. We see that with weight sharing, the results drop by 0.27 points on average by (#3 vs #2). By comparing #2 to #0, we see that SAR is beneficial in modeling the long sequence bringing 1.44 points improvement on average. Except for those observation on average scores, let us take a closer to the how HST and SAR perform in different types of data. Since HST is expected to be more useful on tasks that require global information, let's look at ListOps, Text and Image first. It can be seen that after removing HST, the average decrease is 1-2 points (ListOps, Text and Image in #1 vs. #0). Especially, for ListOps, the task need the model to see all characters in order to do valid logical inference, HST have shown its importance (35.97 vs. 37.75). Except for the global information, some tasks require the model has more stability. For example, the image task Pathfinder, dropped 3.5 points after removing SAR (75.25 vs. 78.77) showing the effect of SAR. Except for those component discussed above, the relation between SAR and R-Drop (#4) will be discuss later.

Effect of SAR vs. R-Drop In this study, we specifically investigate the importance of Dropout (Srivastava et al., 2014) in those experiments. As the Dropout technique is commonly used in deep neural network training and (Liang et al., 2021) use Dropout to constrain the outputs of two subnetworks, we ablate Dropout to compare SAR and R-Drop (Liang et al., 2021) as shown by (#4 vs. #0) in Table 6. #1 is the SAR-only version ERNIE-SPARSE result, which means for getting the regularization version model output $\mathcal{P}_2(y \mid x)$ in 3.3, we roll the sparse attention input (which is equivalent to shift sparse attention topology) and use the

Models	ListOps	Text	Retrieval	Image	Pathfinder	Avg
#0 ERNIE-SPARSE	37.75	64.47	62.64	45.28	78.77	57.60
#1 w/o HST	35.97	63.25	62.24	42.76	78.79	56.60
#2 w/o SAR	37.36	63.54	61.87	42.77	75.25	56.16
#3 w/o SAR + ws in HST	37.41	62.55	60.32	42.62	76.55	55.89
#4 w/o SAR + R-Drop	37.68	61.73	60.59	43.48	78.11	56.32

Table 6: Performance of ERNIE-SPARSE by ablating each proposed components. **ws** means the linear weights for hierarchical and sparse attention are shared.

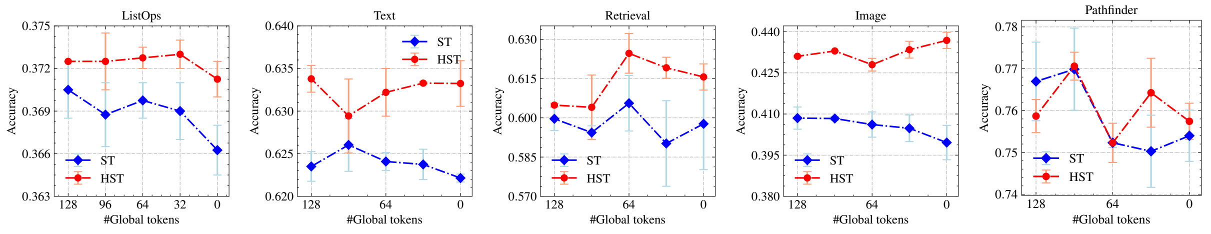


Figure 3: Phenomenon of bottleneck and the effectiveness of **HST**: Accuracy across global token size in the LRA benchmark. A downward trend can be observed for ST on all datasets when bottleneck size decrease (blue line). With our proposed HST method, the performance is less affected by the bottleneck size (red line). Especially, task Text and Image is even better with the #Global tokens decreasing from 64 to 0. Error bar denotes standard deviation.

Dropout at the same time, #4 means that we only use Dropout to get the regularization version model output $\mathcal{P}_2(y \mid x)$. We see that SAR achieve the best for the average score (#4 vs. #0). Moreover, for specific task in Table 6, our method is more efficient in 3 out of 5, indicating not only SAR's effectiveness, but also that SAR and Dropout are compatible and complementary.

5 Discussion

5.1 Bottleneck Analysis

In this section, we analyze the impact of bottleneck size on performance of Sparse Transformer and HST respectively. All the hyperparameter configuration follows (Tay et al., 2020c). We ran each experiment twice and average the results. The experimental results can be seen in Figure 3. As discussed in section 3.1, the number of global tokens determines the size of a bottleneck in Sparse Transformer and we recorded the trend of Sparse Transformer and HST in LRA dataset by changing the size of bottleneck. In this figure, blue and red line denotes Sparse Transformer and HST respectively. As shown in Figure 3, with the information bottleneck size decreases, the performance has a downward trend.

This trend indicates that Sparse Transformer is more prone to the bottleneck size. For our method, the performance of HST in Figure 3 is less affected by the bottleneck size by using representative token for richer global information interaction. As can be seen in this figure, the task score of HST (red line) in each subfigure can be maintained in the same interval without a downward trend, Especially, task Text and Image is even better with the #Global tokens decreasing from 64 to 0.

6 Conclusion

In this paper, we propose ERNIE-SPARSE. Firstly, the advantages and disadvantages of Sparse Transformer are analyzed from the perspective of information flow and we introduce HST to provide a richer path for interaction between input tokens. Secondly, we propose a self-attention regularization method, SAR, to improve the consistency of sparse transformers with different attention topologies. Experiments on various downstream tasks demonstrate that ERNIE-SPARSE outperforms a variety of strong baseline methods including the full-rank attention and other efficient sparse and dense attention methods.

References

Joshua Ainslie, Santiago Ontañón, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. 2020. Etc: encoding long and structured data in transformers. arXiv e-prints, pages arXiv–2004.

Uri Alon and Eran Yahav. 2021. On the bottleneck of graph neural networks and its practical implications. In International Conference on Learning Representations.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150.

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

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

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.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers, pages 31–36. Association for Computational Linguistics.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005., volume 2, pages 729–734. IEEE.

Jun He, Liqun Wang, Liu Liu, Jiao Feng, and Hao Wu. 2019. Long document classification from local word glimpses via recurrent attention learning. IEEE Access, 7:40707–40718.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 8018–8025. AAAI Press.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pages 1601–1611. Association for Computational Linguistics.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 5156–5165. PMLR.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David P. A. Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. In Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019, pages 829–839. Association for Computational Linguistics.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. Technical report.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. CoRR, abs/1909.11942.

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-drop: Regularized dropout for neural networks.

Drew Linsley, Junkyung Kim, Vijay Veerabadran, Charles Windolf, and Thomas Serre. 2018. Learning long-range spatial dependencies with horizontal gated recurrent units. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 152–164.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net.

Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. 2020. Adversarial training for large neural language models. CoRR, abs/2004.08994.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. arXiv preprint arXiv:1905.13164.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA, pages 142–150. The Association for Computer Linguistics.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2019. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. IEEE Trans. Pattern Anal. Mach. Intell., 41(8):1979–1993.

Nikita Nangia and Samuel R. Bowman. 2018. Listops: A diagnostic dataset for latent tree learning. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 2-4, 2018, Student Research Workshop, pages 92–99. Association for Computational Linguistics.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pages 4052–4061. PMLR.

Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. 2019. Blockwise self-attention for long document understanding. arXiv preprint arXiv:1911.02972.

Dragomir R. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The ACL anthology network corpus. Lang. Resour. Evaluation, 47(4):919–944.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21:140:1–140:67.

Tobias Rohde, Xiaoxia Wu, and Yinhan Liu. 2021. Hierarchical learning for generation with long source sequences. arXiv preprint arXiv:2104.07545.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip S. Yu, and Caiming Xiong. 2020a. Adv-bert: BERT is not robust on misspellings! generating nature adversarial samples on BERT. CoRR, abs/2003.04985.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020b. Ernie 2.0: A continual pre-training framework for language understanding. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8968–8975.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2020a. Synthesizer: Rethinking self-attention in transformer models. CoRR, abs/2005.00743.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020b. Sparse sinkhorn attention. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, 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. 2020c. Long range arena: A benchmark for efficient transformers. arXiv preprint arXiv:2011.04006.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning. CoRR, abs/1806.02847.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008.

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

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. Trans. Assoc. Comput. Linguistics, 6:287–302.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Hi-transformer: Hierarchical interactive transformer for efficient and effective long document modeling. arXiv preprint arXiv:2106.01040.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems, 32.

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. Advances in Neural Information Processing Systems, 33.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 9051–9062.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. arXiv preprint arXiv:1905.06566.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, pages 19–27. IEEE Computer Society.

A Appendix

A.1 Pretraining

A.1.1 Pretraining Dataset

For ERNIE-SPARSE pretraining, we use Wikipedia (English Wikipedia dump[3]; 12GB), BookCorpus (Zhu et al., 2015) (4.6GB), Realnews (Zellers et al., 2019) (7.4GB) and Stories (Trinh and Le, 2018) (11GB). For pretraining, we also sample 5% training data as the validation set to monitor the training process. Table 7 shows statistics of the pretraining data.

A.1.2 Pretraining Hyperparameters

We split any document longer than 4096 into multiple documents and we joined multiple documents that were much smaller than 4096. During the pre-training phase, we only use mask language model for training tasks. Specifically, we mask 15% of tokens in these four datasets, and train ERNIE-SPARSE to predict the mask. We warm start ERNIE-SPARSE from RoBERTa's checkpoint. The hyperparameters for these ERNIE-SPARSE are given in Table 8. We use a learning rate warmup over the first 10,000 steps, and polynomial decay of the learning rate. Notably, attention weight in HST are shared with sparse attention.

A.2 Tasks

To evaluate ERNIE-SPARSE, we chose three benchmarks, including LRA, and text classification, as well as question answering. The latter two need to follow the pretraining and finetuning paradigm. Table 11 lists the data distribution, task type, evaluation metric of each dataset.

A.2.1 Hyperparameters for LRA

Table 10 gives the detail list of hyperparameters used to get results shown in Table 1.

A.2.2 Hyperparameters for Classification and QA

Table 9 gives the detail list of hyperparameters used to get results shown in Table 3 and Table 4.

[3]https://dumps.wikimedia.org/enwiki/

Source	Tokens	Avg doc len
Wikipedia	3.0B	515
BookCorpus	1.2B	23K
Realnews	1.8B	3.0K
Stories	2.7B	8.7K

Table 7: Pretraining data statistics.

Parameter	**ERNIE-SPARSE**
α of L_{SAR}	0
learning rate	$3e{-}5$
batch size	256
weight decay	0.1
warmup steps	10k
total steps	1m
max seq length	4096
embedding dim	768
#head	12
#layer	12
activation layer	gelu
dropout	0.1
attn dropout	0.1

Table 8: Hyperparameters for the ERNIE-SPARSE for Pretraining.

A.3 Complexity Analysis

In this section, we analyze the complexity of ERNIE-SPARSE. Note that the two techniques of ERNIE-SPARSE, HST and SAR, are general and can be applied with any type of sparse patterns. Considering that only HST of the two techniques affects computation complexity, we will only analyze the performance of HST here. In order to analyze the complexity, we will follow the following process: firstly, we select a sparse attention and secondly, we analyze the HST complexity to compare the additional computation complexity brought by HST. For the sparse pattern we choose to use the recently proposed pattern from BigBird (Zaheer et al., 2020) for comparison, and the computation complexity is $O(n)$. For HST layer, as it is a dense attention for m representation tokens, the computation complexity is $O(m^2)$. Though the computation complexity for HST is quadratic, this will not bring over computation overhead as the number of representative tokens $m \ll n$ in practice.

Hyperparameter	**Arxiv**	**IMDB**	**Hyperpartisan**	**WikiHop**	**TriviaQA**
HST pooling	mean	mean	mean	mean	mean
α of $\mathcal{L}_{SAR}$	10	0.1	0	10	3
#roll tokens of $\mathcal{L}_{SAR}$	8	8	0	8	8
learning rate	6e-5	1e-5	3e-5	3e-5	3e-5
batch size	48	64	16	48	32
epoch	10	20	20	30	10
warmup	10%	10%	10%	200 (steps)	10%
max seq len	4096	2048	1024	4096	4096
#global token	128				
local window size	192				
#random token	192				
Optimizer	Adam				
Weight Sharing Option	ON				

Table 9: Hyperparameters of classification and question answering tasks.

Hyperparameter	**ListOps**	**Text**	**Retrieval**	**Image**	**Pathfinder**
Hyperparameters for HST and SAR					
HST pooling	$\{\mathrm{MIN}, \mathrm{MEAN}, \mathrm{MAX}\}$				
α of $\mathcal{L}_{SAR}$	$\{0.5, 5, 10\}$				
#roll tokens of $\mathcal{L}_{SAR}$	$\{2, 8\}$				
Weight Sharing Option	OFF				
Fixed hyperparameters provided by LRA (Tay et al., 2021)					
learning rate	5e-2	5e-2	5e-2	5e-4	1e-3
batch size	32	32	32	256	512
weight decay	1e-1	1e-1	1e-1	0	0
warmup	1000	8000	8000	175	312
max seq length	2000	1000	4000	1024	1024
embedding dim	512	256	128	32	64
#head	8	4	4	1	2
#layer	4	4	4	1	4
Q/K/V dim	512	256	128	32	32
MLP dim	1024	1024	512	64	64
dropout	0.1	0.1	0.1	0.3	0.2
attn dropout	0.1	0.1	0.1	0.2	0.1
lr decay	root square	root square	root square	cosine	cosine

Table 10: The upper part is the hyperparameter related to ERNIE-SPARSE, while the lower part is the fixed hyperparameter provided by LRA and cannot be changed.

A.4 More details for SAR

Training with doubled steps. As discussed in Section 3.3, we use the SAR as a regularization term in the total loss. To achieve this training goal, we firstly halve the batchsize and then copy the batch samples to keep the total batchsize is unchanged. In order to ensure sufficient convergence of SAR, we double the training steps. Therefore, it is necessary to see the effect for changing training steps.

To conduct those experiments, we choose Image and Text in LRA as the ablation dataset. Using HST as the baseline model which is denoted as $\mathrm{Baseline}$ in Figure 4, and we double the training steps for HST which is denoted as $\mathrm{Baseline} + \mathrm{DoubleStep}$. Finally, we apply SAR on $\mathrm{Baseline}$. All of those results are recorded in Figure 4. We can see that $\mathrm{Baseline} + \mathrm{DoubleStep}$ shows the trend for overfitting, while $\mathrm{Baseline} + \mathrm{DoubleStep} + \mathrm{SAR}$ has a better convergence curve. For the additional training cost of SAR, we record the training speed for Text. For the $\mathrm{Baseline}$, the speed is 59 samples per second, and 56 samples per second after applying SAR. The additional cost is from the KL-divergence loss backward computation. We can see the cost is 1.03 times, which is a negligible cost.

How to Shift the sparse attention topology. SAR is devised upon self-attention and requires different attention topologies for same model input x, which is defined in 3.1. Actually, to get a shifted

Corpus	**Task**	**Split**	**#Sample**	**Length in percentile**				**#Label**	**Metrics**
				50%	**90%**	**95%**	**max**		
Long Range Arena (LRA)									
ListOps	Logical Reasoning	Train	96k	954	1646	1800	1999	10	Acc
		Dev	2k	960	1648	1813	1999		
		Test	2k	947	1657	1803	1999		
Text	Sentiment Classification	Train	25k	979	2615	3431	13704	2	Acc
		Dev	25k	962	2543	3333	12988		
Retrieval	Retrieval	Train	147k	7648	13467	20495	72885	2	Acc
		Dev	18k	7665	13359	19928	72885		
		Test	17k	7702	15955	22427	50012		
Image	Category Classification	Train	45k	-	-	-	1024	10	Acc
		Dev	5k	-	-	-	1024		
		Test	10k	-	-	-	1024		
PathFinder	Image Reasoning	Train	160k	-	-	-	1024	2	Acc
		Dev	20k	-	-	-	1024		
		Test	20k	-	-	-	1024		
Text Classification									
Arxiv	Category Classification	Train	33k	14733	34209	43951	1121751	11	Micro F1
		Test	3.3k	14710	32417	40965	850540		
IMDB	Sentiment Classification	Train	25k	215	569	748	3084	2	Micro F1
		Test	25k	212	550	724	2778		
Hyperpartisan	News Classification	Train	516	536	1517	1990	5560	2	Micro F1
		Dev	65	520	1535	1971	2637		
		Test	65	637	1771	1990	5560		
Question Answering									
TriviaQA	Question Answering	Train	110k	4576	5027	5166	10091	Span	Macro F1 & EM
		Dev	14k	4577	5026	5169	10210		
WikiHop	Question Answering	Train	43k	1313	3001	3685	19747	Candidates	Acc
		Dev	5.1k	1413	3184	3871	17004		

Table 11: Downstream tasks statistics. Samples of tasks Image and PathFinder are all 32×32 images.

version attention topology, a quick implementation is that we can roll the attention input H, as shown in Figure 2. Furthermore, we can choose any layer for applying the rolling operator to get the shifted version attention topology. However, note that if the SAR is applied from the first layer, the rolling operator should be applied after the word embedding adding the position embedding.

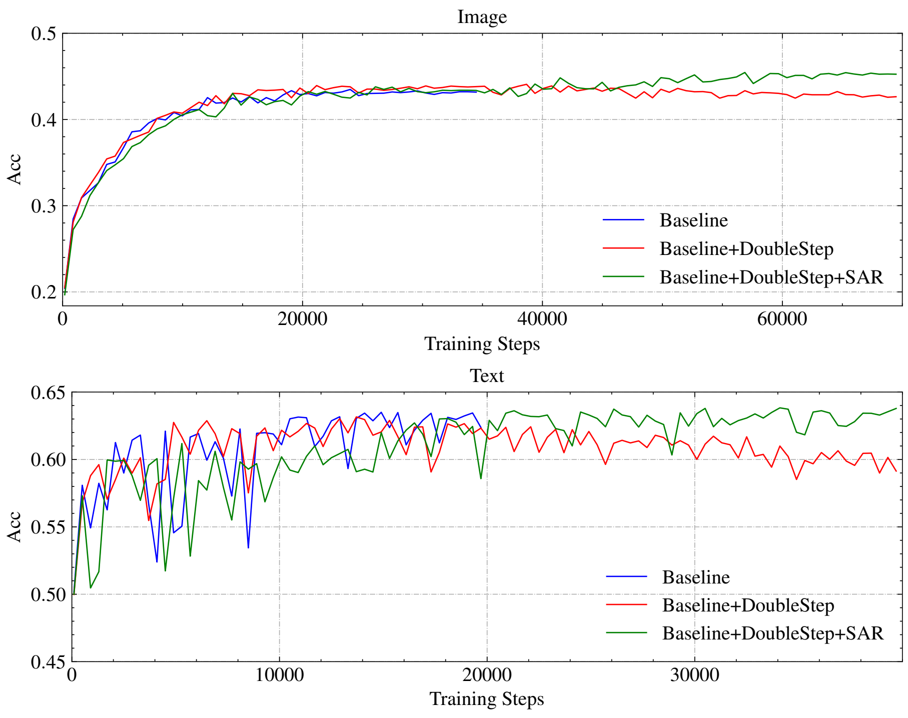


Figure 4: Results of SAR and Baseline on task Image and Text with doubled training steps.