Efficient Hybrid Long Sequence Modeling with State Space Augmented Transformers

Simiao Zuo[*◇], **Xiaodong Liu**[*◇], **Jian Jiao**[◇], **Denis Charles**[◇], **Eren Manavoglu**[◇], **Tuo Zhao**[‡] and **Jianfeng Gao**[◇]
[◇]Microsoft [‡]Georgia Institute of Technology
{simiaozuo,xiaodl}@microsoft.com tourzhao@gatech.edu

Abstract

Transformer models have achieved superior performance in various natural language processing tasks. However, the quadratic computational cost of the attention mechanism limits its practicality for long sequences. There are existing attention variants that improve the computational efficiency, but they have limited ability to effectively compute global information. In parallel to Transformer models, state space models (SSMs) are tailored for long sequences, but they are not flexible enough to capture complicated local information. We propose SPADE, short for **S**tate s**P**ace **A**ugmente**D** Transform**E**r. Specifically, we augment a SSM into the bottom layer of SPADE, and we employ efficient local attention methods for the other layers. The SSM augments global information, which complements the lack of long-range dependency issue in local attention methods. Experimental results on the Long Range Arena benchmark and language modeling tasks demonstrate the effectiveness of the proposed method. To further demonstrate the scalability of SPADE, we pre-train large encoder-decoder models and present fine-tuning results on natural language understanding and natural language generation tasks.

1 Introduction

With the rise of large language models, the difficulties of modeling long sequences have gained increasing attention. For instance, ChatGPT is capable of handling context that comprises up to 8k tokens, while GPT-4 (OpenAI, 2023) scales this ability up to 32k tokens. Conventional Transformer-based models rely on the attention mechanism (Vaswani et al., 2017), which computes a dependency score for every pair of tokens in the input sequence. Thus, full attention has a quadratic time and space complexity with respect to the length of the sequence. However, such complexity proves computationally prohibitive for tasks that require modeling long sequences such as text summarization (Nallapati et al., 2016) and question-answering (Kwiatkowski et al., 2019). In fact, we find that training a Transformer model (250M parameters) takes up over 80G of GPU memory when modelling an input sequence of length 8k.

In addition, Transformer models that rely on full attention run the risk of overfitting due to the lack of structural biases (Lin et al., 2022). The attention mechanism does not impose any structural prior over the input. As a result, order information (such as sinusoidal encoding) is required to train a Transformer model successfully. Transformer models, equipped with full attention mechanism, prove overly flexible resulting in overfitting to the noise. This significantly impacts the models' practicality in long sequence modeling, where the dependency signal is often weak, and signal-to-noise ratio is low (i.e., in long sequences, a majority of input tokens are useless). Empirical evidence shows that Transformer models without structural biases have a classification accuracy rate of 57.5% on a two-way classification task, nearly 30% less than state-of-the-art approaches that are equipped with powerful structural biases (see Section 4.1 for details).

[*]Simiao Zuo and Xiaodong Liu made equal contributions. This work was done when Simiao Zuo did a research internship at Microsoft during his PhD study at Georgia Tech.

Several methods have been proposed to address the problem of full attention's quadratic complexity and the need to introduce structural biases to Transformer models. One of the primary approaches is employing *approximation methods*, which approximates full attention using fast algorithms with linear complexity. Low-rank approximations (Wang et al., 2020b) or kernel methods (Peng et al., 2021), for instance, could simplify and speed up calculation of the attention score matrix (i.e., $\text{softmax}(\mathbf{Q}\mathbf{K}^\top/\sqrt{d})$ in Eq. 1). Nevertheless, although these methods mitigate the computational complexity of full attention, they inherit the problems associated with the lack of structural bias.

To incorporate structural biases into Transformer, *partial attention* methods have been proposed. The methods can be further categorized in to *sparse attention* (Beltagy et al., 2020) and *clustering* (Kitaev et al., 2020). In the former approach, individual tokens attend to only a subset of tokens determined by pre-defined sparsity patterns. In the latter, tokens are divided into clusters and intra-cluster attention is performed. However, introducing these structural biases restricts the models' ability to capture global information. Local window attention, for instance, assumes that each token depends only on its direct neighbors, causing long-range and global information to be lost.

State space models (SSMs) introduce structural biases tailored for computing global information (Gu et al., 2022b), in contrast to partial attention. Specifically, SSMs design fixed global dependency patterns that facilitate effective and efficient computation, and can be seen as linear recurrent neural networks with specifically designed fixed weights. Moreover, efficient algorithms have been developed to train these models. However, it should be noted that SSMs can be restrictive, because they do not capture local information as effectively as attention-based models, which explicitly compute dependencies among input tokens.

We propose SPADE, short for **S**tate s**P**ace **A**ugmente**D** Transform**E**r. SPADE is a multi-layer Transformer model that can effectively and efficiently capture complicated dependencies. Specifically, we augment a state space model (SSM) into the bottom layer of the model to integrate inputs with global information. Because the SSM only provides coarse global information, at the subsequent top layers of SPADE, we employ local attention methods to capture more complicated and refined local information. With this approach, the SSM induces a strong structural bias that augments global information and complements the long-range dependency issue in local attention methods. SPADE is flexible to accommodate different building blocks. For example, we can use SSMs such as S4 (Gu et al., 2022b) and S5 (Smith et al., 2022) in the bottom layer; while for subsequent layers, local attention algorithms such as sliding window attention and chunk attention can be applied.

We demonstrate the efficiency and effectiveness of SPADE on various tasks. First, we show that SPADE outperforms existing approaches on the Long Range Arena (Tay et al., 2021b) benchmark, which is designed to test models' ability in modeling long sequences. Second, we show that in autoregressive language modeling, SPADE is not only significantly faster than the vanilla Transformer (Vaswani et al., 2017), but also yields better performance. Third, we demonstrate the scalability of SPADE by conducting language model pre-training and fine-tuning experiments. Specifically, we pre-train an encoder-decoder model similar to T5 (Raffel et al., 2020). And we fine-tune the model on various tasks, including natural language understanding and natural language generation benchmarks. In all the settings, SPADE outperforms the baselines. Our code[1] and pre-trained model checkpoints[2] are publicly available.

2 Background

2.1 Attention Mechanism

Suppose the input to the layer is $\mathbf{X} \in \mathbb{R}^{L \times d}$, where L is the sequence length and d is the embedding dimension, then the attention mechanism outputs

$$\text{Attn}(\mathbf{X}) = \text{softmax}(\mathbf{Q}\mathbf{K}^\top/\sqrt{d})\mathbf{V}, \text{ where } \mathbf{Q} = \mathbf{X}\mathbf{W}_q,\ \mathbf{K} = \mathbf{X}\mathbf{W}_k,\ \mathbf{V} = \mathbf{X}\mathbf{W}_v. \tag{1}$$

[1] https://github.com/microsoft/EfficientLongSequenceModeling

[2] https://github.com/namisan/mt-dnn

Here $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d\times d}$ are learnable weights. The attention mechanism can simultaneously compute the alignment between any pair of input tokens, such that it models long-range dependencies better than recurrent neural networks. Specifically, denote the attention score matrix $\mathbf{A} = \text{softmax}(\mathbf{Q}\mathbf{K}^\top/\sqrt{d}) \in \mathbb{R}^{L\times L}$. Then, $\mathbf{A}_{ij}$ captures the alignment between the i-th and the j-th input tokens.

2.2 State Space Models

Continuous time state space model. A continuous time latent state space model maps a 1-dimensional input signal $u(t)$ to a d_s-dimensional latent state $x(t)$, after which $x(t)$ is mapped to a 1-dimensional output signal $y(t)$. Concretely,

$$x'(t) = \mathbf{A}x(t) + \mathbf{B}u(t), \quad y(t) = \mathbf{C}x(t). \tag{2}$$

Here, $\mathbf{A} \in \mathbb{R}^{d_s\times d_s}$, $\mathbf{B} \in \mathbb{R}^{d_s}$ and $\mathbf{C} \in \mathbb{R}^{d_s}$.

Existing works leverage Eq. 2 to model long sequences. For example, Gu et al. (2020) claim that randomly initialized parameters $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ cannot model long-range dependencies well. Subsequently, a class of matrices (termed HiPPO, high-order polynomial projection operators) are proposed to initialize $\mathbf{A}$. The HiPPO matrices are designed such that the state $x(t)$ at time t can memorize the history of the input $u(t)$ up to time t.

Discrete time state space model. In practice, we often work with discrete sequences such as natural language inputs $(u_0, u_1, \cdots, u_L)$, where L is the sequence length. To facilitate modeling discrete data, the model in Eq. 2 can be discretized (using the bilinear method) by a step size Δ, such that

$$\begin{aligned} &x_k = \overline{\mathbf{A}}x_{k-1} + \overline{\mathbf{B}}u_k, \quad y_k = \overline{\mathbf{C}}x_k, \\ &\text{where } \overline{\mathbf{A}} = (\mathbf{I} - \Delta/2\cdot\mathbf{A})^{-1}(\mathbf{I} + \Delta/2\cdot\mathbf{A}),\ \overline{\mathbf{B}} = (\mathbf{I} - \Delta/2\cdot\mathbf{A})^{-1}\Delta\mathbf{B}, \quad \overline{\mathbf{C}} = \mathbf{C}. \end{aligned} \tag{3}$$

We unroll the above recurrent representation, after which we have

$$y_k = \overline{\mathbf{C}}\overline{\mathbf{A}}^k\overline{\mathbf{B}}u_0 + \cdots + \overline{\mathbf{C}}\overline{\mathbf{A}}\overline{\mathbf{B}}u_{k-1} + \overline{\mathbf{C}}\overline{\mathbf{B}}u_k.$$

This can be written as a convolutional representation

$$y = \overline{\mathbf{K}} * u, \text{ where } \overline{\mathbf{K}} \in \mathbb{R}^L = \left(\overline{\mathbf{C}}\overline{\mathbf{B}}, \overline{\mathbf{C}}\overline{\mathbf{A}}\overline{\mathbf{B}}, \cdots, \overline{\mathbf{C}}\overline{\mathbf{A}}^{L-1}\overline{\mathbf{B}}\right). \tag{4}$$

Here, $\overline{\mathbf{K}}$ is the convolutional kernel, "$*$" is the discrete convolution operator, u represents the input sequence $(u_0, u_1, \cdots, u_L)$, and y represents the corresponding output sequence $(y_0, y_1, \cdots, y_L)$.

In Eq. 4, the output y can be computed efficiently given that the convolution kernel $\overline{\mathbf{K}}$ is known (e.g., using Fast Fourier Transform). However, computing the kernel is non-trivial. Most of existing algorithms have $O(L^2)$ time and space complexity.

Structured State Space Sequence model (S4). Gu et al. (2022b) develop the S4 model to efficiently compute Eq. 4. Specifically, $\mathbf{C}$ in Eq. 2 is randomly initialized, and $\mathbf{A}$ and $\mathbf{B}$ are initialized as

$$\begin{aligned} &\mathbf{A} = \mathbf{A}^{(d_s)} - \mathbf{P}\mathbf{P}^\top, \quad \mathbf{B}_i = (2i+1)^{1/2}, \\ &\text{where } \mathbf{P}_i = (i+1/2)^{1/2},\ \mathbf{A}_{ij}^{(d_s)} = -\begin{cases} (i+1/2)^{1/2}(j+1/2)^{1/2}, & i > j, \\ 1/2, & i = j, \\ -(i+1/2)^{1/2}(j+1/2)^{1/2}, & i < j. \end{cases} \end{aligned} \tag{5}$$

Subsequently, the convolution kernel $\overline{\mathbf{K}}$ in Eq. 4 can be computed efficiently with near linear time and space complexity. Then, for an input u, the S4 output $y = \overline{\mathbf{K}} * u$ can be computed efficiently.

More state space models. Despite the superior performance of S4, it suffers from known issues such as difficulty in scaling. Other state space models are subsequently developed, such as MEGA (Ma et al., 2022), S4-D (Gu et al., 2022a), H3 (Fu et al., 2022), S5 (Smith et al., 2022), GSS (Mehta et al., 2022) and Hyena (Poli et al., 2023). Through experiments, we demonstrate that the proposed model is flexible to accommodate variants of SSMs.

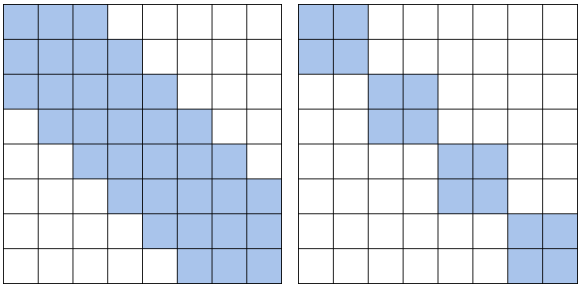

Figure 1: Illustration of window attention (left) and chunk attention (right). For window attention, the window size is 2 (on each side); for chunk attention, the chunk size is 2.

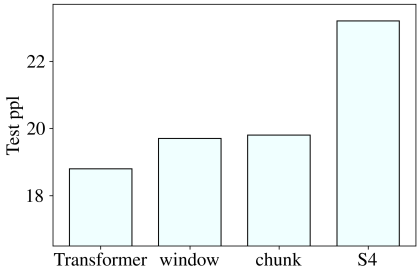

Figure 2: Performance of Transformer with full attention, window attention, chunk attention, and S4. We use language modeling experiments (see Section 4.2), and the sequence length is 3k.

3 Method

We first conduct experiments to demonstrate that SSMs do not model local information well. Then, we present SPADE, which effectively combines global and local information by augmenting SSMs into the Transformer architecture.

3.1 Attention vs. State Space Models

The motivation behind SPADE is that even though SSMs perform well on several long sequence classification tasks (Gu et al., 2022b), they perform poorly on language modeling, which is a fundamental task in natural language processing. To demonstrate such an observation, we compare S4 with Transformer with full attention and Transformer with local (window and chunk) attention. In local attention, each token can only attend to its neighboring tokens (see Figure 1 for illustrations). We conduct experiments on token-level language modeling. In this setting, local information is more important than global information. This is because in practice, we rarely see words (tokens) that are thousands of positions apart exhibit strong dependencies (Sukhbaatar et al., 2019).

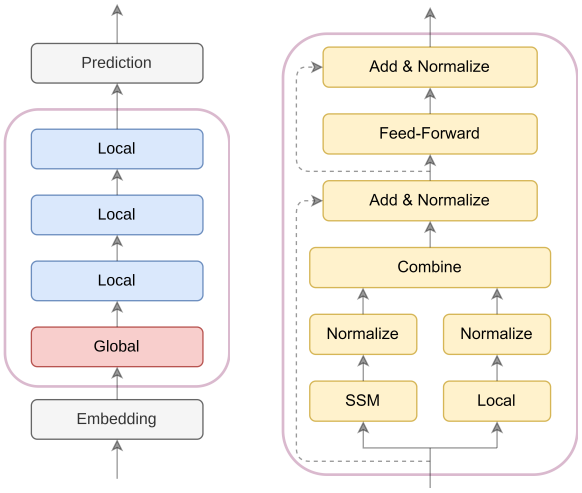

Figure 3: Demonstration of SPADE with 4 layers. Left: overview; Right: details of global layer.

Experimental results are illustrated in Figure 2. We see that both Transformer with full attention and Transformer with local attention (e.g., window and chunk) outperforms S4. Notice that replacing full attention with local attention does not significantly hurt model performance, indicating that local information is more important in this setting. We remark that SSMs such as S4 produces a fixed dependency pattern, e.g., the convolution kernel in Eq. 4. Moreover, unlike attention, SSMs do not explicitly compute dependencies among tokens. Therefore, SSMs are not refined enough to capture local information, such that they perform poorly on language modeling tasks.

3.2 SPADE: State Space Augmented Transformer

We propose SPADE, which is a multi-layer Transformer model that can capture complicated global and local information. The overall architecture of SPADE is shown in Figure 3 (left). The proposed model employs a hierarchical structure. Specifically, at the bottom layer of SPADE (termed the *global* layer), we capture global dependencies using a SSM. Because the SSM only provides coarse global information, the subsequent *local* layers facilitate the

model to handle more refined and complicated local dependencies. In other words, the SSM induces a strong structural bias that augments global information to the inputs.

To instantiate the local layer, we replace the full attention in the conventional Transformer layer with off-the-shelf efficient local attention methods. SPADE is flexible to accommodate different approaches, such as window attention and chunk attention (see Figure 1 for illustrations).

In the global layer (Figure 3, right), given the input $\mathbf{X}$ to the layer, we have the output $\mathbf{Y}$ as

$$\mathbf{Y} = \mathrm{FFN}\left(\mathrm{LN}(\mathbf{X}_a)\right) + \mathbf{X}_a, \text{ where } \mathbf{X}_a = \mathbf{W}\left[\mathrm{LN}(\mathbf{X}_{\text{local}}), \mathrm{LN}(\mathbf{X}_{\text{global}})\right] + \mathbf{X},$$
$$\mathbf{X}_{\text{local}} = \mathrm{Local}\left(\mathrm{LN}(\mathbf{X})\right), \ \mathbf{X}_{\text{global}} = \mathrm{SSM}\left(\mathrm{LN}(\mathbf{X})\right).$$

Here, $\mathrm{LN}(\cdot)$ denotes layer normalization (Ba et al., 2016), $\mathrm{FFN}(\cdot)$ denotes a two-layer feed-forward neural network, and $\mathbf{W}$ is a trainable weight that combines local and global representations. Notice that we apply normalization to $\mathbf{X}_{\text{local}}$ and $\mathbf{X}_{\text{global}}$ to align their scales.

SPADE is flexible to accommodate different building blocks. For example, in the experiments we use S4 (Gu et al., 2022b), S5 (Smith et al., 2022) and Hyena (Poli et al., 2023) as the SSM in the bottom layer. In the top layers, local attention algorithms such as sliding window attention and chunk attention can be used.

We remark that because of the sequential nature of SSMs (Eq. 3), the global layer can encode positional information of the inputs. Therefore, we do not need additional fixed-length positional embedding techniques (Devlin et al., 2019).

4 Experiments

In the experiments, we implement all the models using *PyTorch* (Paszke et al., 2019) and *Fairseq* (Ott et al., 2019). Training details such as hyper-parameter settings are deferred to the appendix.

4.1 Long Range Arena

Dataset and models. We evaluate SPADE on Long Range Arena (LRA, Tay et al. 2021b), which is a benchmark tailored for evaluating models' ability in modeling long sequences. Dataset details are presented in Appendix B.

Following the setting in Ma et al. 2022, we use small models (less than 2M parameters) for all the tasks. We limit the computational budget such that all the models are trained with similar speed for the same amount of time. To aggregate local information, we consider two approaches: window attention and chunk attention. We use S4 Gu et al. as the SSM in the bottom layer.

Dataset	**Listops**	**Text**	**Retrieval**	**Image**	**Pathfinder**	**Path-X**	**Avg.**
Sequence length	2k	4k	8k	1k	1k	16k	—
Transformer (full attention)	36.37	64.27	57.46	42.44	71.40	✗	53.66
S4 (Gu et al., 2022b)	58.35	76.02	87.09	87.26	86.05	88.10	80.48
MEGA (chunk) (Ma et al., 2022)	58.76	90.19	90.97	85.80	94.41	93.81	85.66
SPADE (S4+window)	59.70	87.55	90.13	**89.11**	**96.42**	94.22	86.19
SPADE (S4+chunk)	**60.50**	**90.69**	**91.17**	88.22	96.23	**97.60**	**87.40**

Table 1: Experimental results on LRA. The best results are shown in **bold**. For Path-X, "✗" indicates unavailable results due to computational constraints. See Table 11 in the appendix for comparison with other baselines.

Results. Experimental results are summarized in Table 1 (results of other baselines are shown in Table 11 in the appendix). We see that both variants of SPADE (window and chunk) significantly outperform all the baselines in terms of average accuracy. For example, the window attention variant outperforms the best-performing baseline (MEGA-chunk) by 0.5%, and the chunk attention variant has a 1.8% performance gain. Therefore, SPADE is more suitable to model long sequences than existing approaches.

4.2 Language Modeling

Dataset and models. We further evaluate our model by conducting language modeling experiments on Wikitext-103. The dataset contains English-language Wikipedia articles, and the total number of tokens is 103M. In all the experiments, we set the input sequence length to 3k and train for 286k steps. During testing, we set the sequence length to 3k and the context window size to 400.

Model	Test ppl
Transformer (Vaswani et al., 2017)	18.8
Transformer (relative) (Shaw et al., 2018)	18.7
Transformer (window)	19.7
Transformer (chunk) (Hua et al., 2022)	20.9
MEGA (chunk) (Ma et al., 2022)	19.8
S4 (Gu et al., 2022b)	23.2
S5 (Smith et al., 2022)	23.0
Hyena (Poli et al., 2023)	23.2
SPADE (S4+chunk)	19.5
SPADE (S4+window)	18.5
SPADE (S5+window)	**18.3**
SPADE (Hyena+window)	18.5

Table 2: Experimental results on Wikitext-103. We present model perplexity on the test set.

We use window attention as the local information extractor in SPADE. And we equip our model with three variants of SSMs: S4 (Gu et al., 2022b), S5 (Smith et al., 2022) and Hyena (Poli et al., 2023). For SPADE and Transformer-based baselines, we follow the settings in Baevski & Auli (2019), where we use a large-scale Transformer model with 16 layers and about 250M parameters. For SSM-only baselines (e.g., S4), we also scale the models such that they contain approximately the same number of parameters.

Results. Experimental results are presented in Table 2. From the results, we see that the proposed model achieves significant performance improvement and outperforms all the baselines. We remark that SPADE with window attention is not only significantly faster than the Transformer with full attention, but also yields a better performance.

Remark. We remark that we do not need to train the S4 in the bottom layer of SPADE (S4+window) to achieve the performance in Table 2. That is, we initialize the parameters in S4 using Eq. 5, and the parameters are frozen during training. This is because even without training, the initialization of S4 yields intriguing theoretical properties, which facilitates S4's ability to capture global information.

5 Language Model Pre-Training

We implement model pre-training using *Fairseq*, and we implement model fine-tuning using *MT-DNN* (Liu et al., 2019a; 2020b). Note that all our experiments only use single task fine-tuning. Details such as pre-training settings and hyper-parameter settings are deferred to the appendix.

5.1 Pre-Training Details

To demonstrate the scalability of the proposed method, we pre-train an encoder-decoder variant of SPADE. The model architecture is the same as T5_{base} (Raffel et al., 2020), except that we use post-layernorm instead of pre-layernorm to improve model performance (Liu et al., 2020a; Xiong et al., 2020). The embedding dimension is 768, the hidden dimension of the FFN is 3072, the number of attention heads is 12, and both the encoder and the decoder have 12 layers. We add a S4 module to the bottom layer of SPADE, and the parameters of the S4 are fixed after initialization (Eq. 5). We use the window attention as the local information extractor, where we set the window size to 128. The model contains about 290M parameters.

We consider two pre-training settings. In the first setting, we follow the pre-training settings in BERT (Devlin et al., 2019), and we term the trained model $\textbf{SPADE}_{\textbf{base}}$. In the second setting, we follow the pre-training settings in RoBERTa (Liu et al., 2019b), and we term the trained model $\textbf{SPADE}_{\textbf{base++}}$. To facilitate fair comparisons, we also pre-train a T5 model using the second setting (the model yields better performance than the first setting), and we term the pre-trained model $\textbf{T5}_{\textbf{base}}$ **(re-imp)**.

We remark that because the S4 module is not trained after proper initialization, and we do not use fixed-length positional embedding, our pre-trained model can be fine-tuned with

any sequence length. For example, we can set the sequence length to 16k during fine-tuning, which is longer than the sequence length used in pre-training.

5.2 Natural Language Understanding

We fine-tune the pre-trained models on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019), which is a collection of natural language understanding tasks. Dataset details are presented in Appendix B. We do not consider the long sequence setting in these tasks. In the experiments, all models are fine-tuned under the sequence length of 512.

	RTE Acc	**MRPC** Acc/F1	**CoLA** Mcc	**SST-2** Acc	**STS-B** P/S Corr	**QNLI** Acc	**QQP** Acc/F1	**MNLI-m/mm** Acc	**Avg. Score**
$T5_{base}$	76.9	90.8/–	55.5	92.8	86.5	91.9	90.9/–	84.4/83.5	—
$T5_{base}$ (re-imp)	78.0	91.7/88.6	61.5	93.6	88.2	92.9	91.2/87.9	87.0/86.9	85.1
$SPADE_{base}$	77.9	92.2/89.0	63.2	94.0	87.9	92.8	91.6/88.2	87.1/87.2	85.4
$SPADE_{base++}$	**80.5**	**92.3/89.2**	**64.7**	**95.9**	**89.2**	**93.9**	**91.7/88.4**	**89.6/89.2**	**86.8**

Table 3: Experimental results on GLUE development set. The best results are shown in **bold**. *$T5_{base}$* results are from Raffel et al. 2020. We pre-train a T5 model $T5_{base}$ (re-imp) to facilitate fair comparisons).

Experimental results are presented in Table 3. We see that both variants of SPADE significantly outperforms $T5_{base}$. For example, $T5_{base}$ has a 83.9 average accuracy on the MNLI dataset (average of MNLI-m and MNLI-mm); while $SPADE_{base}$ has a 87.2 average accuracy (+3.3) and $SPADE_{base++}$ has a 89.4 average accuracy (+5.5). Recall that the sequence length is set to 512, which is the standard setting instead of the long-sequence setting. Therefore, the results indicate that SPADE is universal in that it is suitable to model both long and short sequences.

5.3 Natural Language Generation

	arXiv Length	R-2	**CNN/DailyMail** Length	R-2
$LongT5_{base}$	4k	18.54	4k	20.11
$LongT5_{large}$	16k	21.63	4k	20.51
$LongT5_{xl}$	16k	21.92	4k	21.40
$SPADE_{base++}$	16k	21.65	4k	20.40
	MediaSum Length	R-2	**MultiNews** Length	R-2
$LongT5_{base}$	4k	18.35	4k	17.37
$LongT5_{large}$	4k	19.04	8k	18.44
$LongT5_{xl}$	4k	19.66	8k	19.43
$SPADE_{base++}$	4k	19.03	8k	19.63

Table 4: Experimental results (ROUGE-2) on test sets. LongT5 results are from Guo et al. 2022.

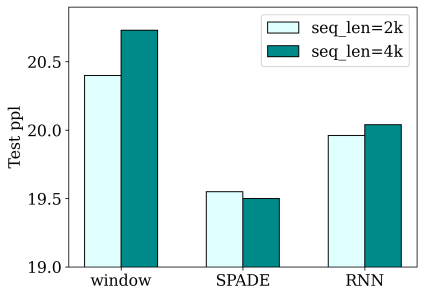

Figure 4: Performance of models with different global information extractors under two sequence length. We conduct language modeling experiments on Wikitext-103 with window attention (window=128).

We fine-tune the pre-trained models on several abstractive summarization datasets. Dataset details are presented in Appendix B. We use ROUGE-2 as the evaluation metric.

We compare SPADE with LongT5 (Guo et al., 2022), which is a state-of-the-art model tailored for long sequences. Experimental results are summarized in Table 4. From the results, we see that our model significantly outperforms LongT5. Note that $SPADE_{base++}$ have about 290M parameters, while $LongT5_{large}$ contains about 770M parameters and $LongT5_{xl}$ contains about 3B parameters. From the results, we see that in all the tasks, our base-sized models have on par or better performance compared with $LongT5_{large}$. On the MultiNews dataset, our model even outperforms $LongT5_{xl}$, which is over ten times larger than our model.

6 Analysis

6.1 Effectiveness of Global Information Extractors

Recall from Eq. 3 that SSMs are essentially linear recurrent neural networks which can be computed much more efficiently than conventional RNNs. In Figure 4, we explore model variants with different global information extractors. Specifically, we equip Transformer with window attention with either S4 or LSTM (Hochreiter & Schmidhuber, 1997).

From the results, we see that RNNs can indeed capture global information. Also, we see that SSMs are more effective than RNNs. This is because RNNs suffer from known numerical problems such as forgetting and gradient explosion/vanishing (Pascanu et al., 2013). On the other hand, the theoretical properties of SSMs (Gu et al., 2020; 2021; 2022b) can greatly alleviate such issues.

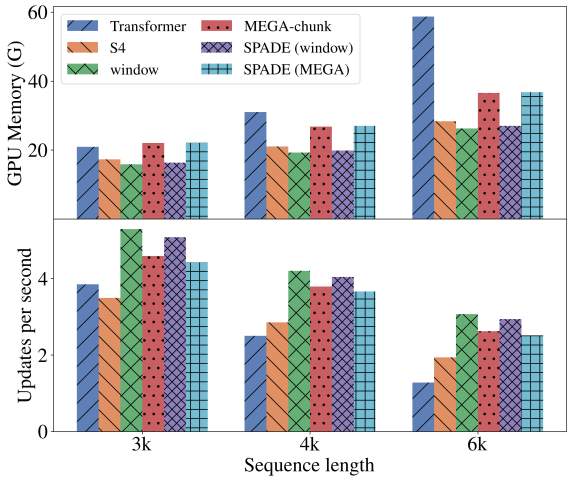

Figure 5: Efficiency comparison of differnet models on language modeling tasks.

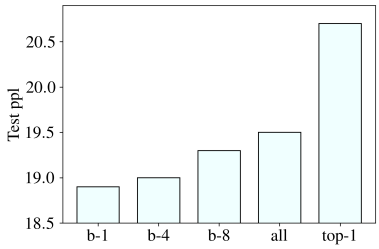

Figure 6: Performance vs. location of SSMs. We conduct language modeling experiments with window attention (window=256). By default, the model has 16 layers, where the bottom layer is a global layer and the rest are local layers. Here, "*b-k*" means the bottom-k layers are global layers, "*all*" means all layers are global layers, and "*top-1*" means the top-1 layer is a global layer.

6.2 Efficiency Comparison

We compare the efficiency of SPADE with other models: Transformer with full attention, Transformer with window attention, MEGA-chunk, and S4. The results are illustrated in Figure 5. We see that SPADE is efficient in terms of both training speed and GPU memory usage. For example, when the sequence length is 6k, Transformer uses about 60GB of GPU memory, whereas SPADE with window attention only uses 27GB. Moreover, notice that SPADE also trains significantly faster than the vanilla Transformer under all settings. Notice that S4 may be less efficient than the vanilla Transformer (e.g., when the sequence length is 3k). This is because in Gu et al. 2022b, each layer of the model contains multiple S4 modules and expensive non-linear components. Therefore, the per-layer computational cost can exceed full attention when the sequence is not extremely long.

We remark that even though we add a S4 module to the bottom layer of SPADE, such an additional module does not induce much computational overhead. We see that both the training speed and the memory usage of SPADE with window is only marginally different from those of window-attention Transformer. We have similar observations for the chunk attention variant.

6.3 Location and Number of Global Layers

Recall that in SPADE, the bottom layer is equipped with a SSM and serves as the global layer, while the rest are local layers (see Figure 3). In Figure 6, we empirically justify this design choice.

We investigate the possibility of incorporating more global layers: we set the bottom 1 (the default choice), 4, 8, and 16 (all) layers as global layers. From the results, we see that model

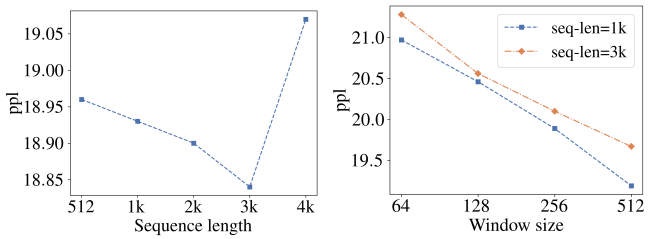

Figure 7: Performance with different configurations. Left: Transformer with full attention using different sequence length; Right: Transformer with window attention using different window size and sequence length.

	Sequence length			
	2k	3k	4k	6k
128	19.55	19.42	19.50	19.55
256	18.90	18.95	18.99	19.00
512	18.63	18.64	18.52	18.67

Table 5: Performance of SPADE with different sequence length and window size. We conduct language modeling experiments on Wikitext-103.

performance decreases as we use more global layers. This is because the SSM in the bottom layer captures and filters out global information, such that subsequent SSMs only introduce noise to the intermediate representations.

We also investigate whether the global layer can be the top instead of the bottom layer in SPADE. From Figure 6, we see that model performance drops significantly. This is because as a global information extractor, the global layer encodes positional information, on which the local attention modules rely. Therefore, using the global layer as the top layer is akin to using Transformer models without positional encoding, which will yield unsatisfactory performance.

6.4 Different Configurations

We examine how performance changes when the sequence length and window size change.

From Figure 7 (left), we see that when we increase the sequence length from 512 to 3k, performance of Transformer with full attention increases. However, when we further increase the sequence length to 4k, model performance drastically drops. This is because in long sequences, the signal-to-noise ratio is low, such that the full attention may easily fit to the noise. From Figure 7 (right), we see that performance of Transformer with window attention increases when we increase the window size. Moreover, model performance is better with shorter sequences for the same window size. Such findings indicate that performance of window attention depends on the proportion of information within its perception.

From Table 5, we see that for the same sequence length, performance of SPADE increases when we increase the window size. Also, we see that performance of SPADE marginally decreases when we increase the sequence length from 4k to 6k. Recall from Figure 7 (left) that performance of Transformer with full attention drastically deteriorates when we increase the length from 3k to 4k. Such a result indicates that the proposed model is more suitable to model long sequences.

7 Conclusion and Discussion

We propose SPADE, a state space augmented Transformer model that targets long sequence modeling. SPADE is a multi-layer Transformer model, where the bottom layer is a global layer and the rest are local layers. In the global layer, we use a SSM to augment coarse global information, which are subsequently refined by the following local layers. SPADE is flexible to accommodate different SSMs such as S4 and S4 as the global information extractor. We instantiate the local layers with off-the-shelf efficient attention methods, such as sliding-window attention. The proposed model has near linear time and space complexity, facilitating it to handle long sequences. We conduct extensive experiments on the Long Range Arena (LRA) benchmark and language modeling datasets to demonstrate the effectiveness and efficiency of SPADE. We also pre-train encoder-decoder models to demonstrate the scalability of SPADE, and we perform fine-tuning experiments on natural language understanding (GLUE) and natural language generation (summarization) tasks. In all the experiments, SPADE exhibits superior performance.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 268–284, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.19.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *ArXiv preprint*, abs/1607.06450, 2016.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, and Danilo Giampiccolo. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

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

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. The fifth pascal recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC'09*, 2009.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001.

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

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. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 615–621, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2097.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pp. 177–190, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-33427-0, 978-3-540-33427-9. doi: 10.1007/11736790_9.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1074–1084, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1102.

Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 1–9, Prague, 2007. Association for Computational Linguistics.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus, 2019.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 572–585, 2021.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022a.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 724–736, Seattle, United States, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.55.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer quality in linear time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9099–9117. PMLR, 2022.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1419–1436, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.112.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 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*, pp. 5156–5165. PMLR, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

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

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. FNet: Mixing tokens with Fourier transforms. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4296–4313, Seattle, United States, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.319.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022.

Drew Linsley, Junkyung Kim, Vijay Veerabadran, Charles Windolf, and Thomas Serre. Learning long-range spatial dependencies with horizontal gated recurrent units. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 152–164, 2018.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5747–5763, Online, 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.463.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4487–4496, Florence, Italy, 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1441.

Xiaodong Liu, Yu Wang, Jianshu Ji, Hao Cheng, Xueyun Zhu, Emmanuel Awa, Pengcheng He, Weizhu Chen, Hoifung Poon, Guihong Cao, and Jianfeng Gao. The Microsoft toolkit of multi-task deep neural networks for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 118–126, Online, 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.16.

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

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 2441–2453, 2021.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. Mega: Moving average equipped gated attention. *ArXiv preprint*, abs/2209.10655, 2022.

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

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*, 2022.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028.

Nikita Nangia and Samuel Bowman. 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: Student Research Workshop*, pp. 92–99, New Orleans, Louisiana, USA, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-4013.

Tan M. Nguyen, Vai Suliafu, Stanley J. Osher, Long Chen, and Bao Wang. Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 29449–29463, 2021.

OpenAI. Gpt-4 technical report. *arXiv*, 2023.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. Random feature attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*, 2023.

Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2555–2565, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.232.

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

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, 2013. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 331–335, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1032.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 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*, pp. 9438–9447. PMLR, 2020.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10183–10192. PMLR, 2021a.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 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, 2021b.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Shuohang Wang, Luowei Zhou, Zhe Gan, Yen-Chun Chen, Yuwei Fang, Siqi Sun, Yu Cheng, and Jingjing Liu. Cluster-former: Clustering-based sparse transformer for long-range dependency encoding. *ArXiv preprint*, abs/2009.06097, 2020a.

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

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019. doi: 10.1162/tacl_a_00290.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. 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*, pp. 10524–10533. PMLR, 2020.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nystöm-based algorithm for approximating self-attention. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 35, pp. 14138. NIH Public Access, 2021.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Hang Zhang, Yeyun Gong, Yelong Shen, Weisheng Li, Jiancheng Lv, Nan Duan, and Weizhu Chen. Poolingformer: Long document modeling with pooling attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12437–12446. PMLR, 2021.

Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17723–17736, 2021a.

Chenguang Zhu, Yang Liu, Jie Mei, and Michael Zeng. MediaSum: A large-scale media interview dataset for dialogue summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, pp. 5927–5934, Online, 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.474.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 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*, pp. 19–27. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.11.

A Related Works

In Eq. 1, we have $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L \times d}$, such that computing the attention $\mathrm{Attn}(\mathbf{X})$ introduces $O(L^2)$ time and space costs. Such quadratic costs are prohibitive when the sequence length L is large. There are various attempts to reduce the quadratic time and space complexity of the vanilla attention.

One approach is to employ *sparse attention*. That is, each token only attends to a subset of all the tokens according to pre-defined patterns, e.g., neighboring tokens within a fixed size window. Some examples include Sparse Transformer (Child et al., 2019), BlockBERT (Qiu et al., 2020), Longformer (Beltagy et al., 2020), ETC (Ainslie et al., 2020), BigBird (Zaheer et al., 2020), HEPOS (Huang et al., 2021), and Poolingformer (Zhang et al., 2021).

Another approach is to use *low-rank projection*. For example, in Linformer (Wang et al., 2020b), the attention mechanism in Eq. 1 becomes $\mathrm{Attn}(\mathbf{X}) = \mathrm{softmax}(\mathbf{Q}(\mathbf{EK})^\top / \sqrt{d})(\mathbf{FV})$. Here, the two additional parameters satisfy $\mathbf{E}, \mathbf{F} \in \mathbb{R}^{r \times L}$, where r is the projection rank such that $r \ll L$. Similar methods include Nyströmformer (Xiong et al., 2021), Synthesizer (Tay et al., 2021a), Transformer-LS (Zhu et al., 2021a), and Luna (Ma et al., 2021). However, these approaches face difficulty when handling causal tasks, such as auto-regressive language modeling. Specifically, in Eq. 1, we mask out the upper triangular part in the attention score matrix $\mathbf{A} \in \mathbb{R}^{L \times L}$ such that each token can only attend to its previous tokens. However, this is implausible in Linformer since we project the $L \times L$ matrix to a $L \times r$ matrix.

Kernel-based approaches can be used to approximate the full attention $\mathrm{Attn}(\mathbf{X})$. In these approaches, the quadratic-time softmax attention is replaced by fast linear-time kernel approximations (e.g., Gaussian and arc-cosine kernel). Some examples include Linear Transformer (Katharopoulos et al., 2020), Performer (Choromanski et al., 2021), Random Feature Attention (Peng et al., 2021), and FMMformer (Nguyen et al., 2021). Both low-rank projection and kernel-based approaches approximate the full attention, and thus, they often suffer from non-negligible approximation error.

We can also adopt *clustering-based approaches*, where we divide $\mathbf{Q}$ or $\mathbf{K}$ into several clusters, and only perform inter-cluster attention. Such methods include Reformer (Kitaev et al., 2020), Clusterformer (Wang et al., 2020a), Sinkhorn Transformer (Tay et al., 2020), Fast Transformer (Vyas et al., 2020), Routing Transformer (Roy et al., 2021), and FLASH (Hua et al., 2022).

B Dataset Details

Long Range Arena. We evaluate the effectiveness of the proposed model on Long Range Arena (LRA, Tay et al. 2021b), which is a benchmark tailored for evaluating models' ability in modeling long sequences. The benchmark contains six tasks: ListOps, which tests the capability of modeling hierarchically structured data (Nangia & Bowman, 2018); byte-level text classification on the IMDB movie review dataset (Text, Maas et al. 2011); byte-level document retrieval on the ACL anthology network (Retrieval, Radev et al. 2013); pixel-level image classification on CIFAR-10 (Image, Krizhevsky et al. 2009); Pathfinder, which tests the capability in modeling spatial dependency (Linsley et al., 2018); and a longer version of Pathfinder (Path-X, Tay et al. 2021b).

GLUE. For natural language understanding, we fine-tune the pre-trained models on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019), which is a collection of natural language understanding tasks. The benchmark includes two single-sentence classification tasks: CoLA (Warstadt et al., 2019) is a linguistic acceptability task; and SST-2 (Socher et al., 2013) is a binary classification task that classifies movie reviews to positive or negative. The benchmark also contains three similarity and paraphrase tasks: STS-B (Cer et al., 2017) is a text similarity task; MRPC (Dolan & Brockett, 2005) is a paraphrase detection task; and QQP is a duplication detection task. Additionally, there are natural language inference tasks: MNLI (Williams et al., 2018); QNLI (Rajpurkar et al., 2016); RTE (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009). Statistics of the GLUE benchmark is summarized in Table 7.

Summarization. For natural language generation, we fine-tune the pre-trained models on several abstractive summarization datasets. The sources and statistics are summarized in Table 6.

Table 6: Statistics and sources of abstractive summarization datasets.

	# Train	# Validation	# Test	Mean	Median	Max	90th percentile
arXiv (Cohan et al., 2018)	203,037	6,436	6,440	10,720	8,519	378,825	20,170
CNN/DailyMail (Nallapati et al., 2016)	287,113	13,368	11,490	982	894	5,268	1,659
MediaSum (Zhu et al., 2021b)	443,596	10,000	10,000	2,302	1,748	125,974	4,128
MultiNews (Fabbri et al., 2019)	44,972	5,622	5,622	2,594	1,902.5	683,544	4,853

Table 7: Statistics of the GLUE benchmark.

Corpus	Task	# Train	# Dev	# Test	# Labels	Metrics
Single-Sentence Classification						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST	Sentiment	67k	872	1.8k	2	Accuracy
Pairwise Text Classification						
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
Text Similarity						
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr

C Training Details

C.1 Language Model Pre-Training and Fine-Tuning

For language model pre-training, we consider two pre-training settings:

- ◇ $\text{SPADE}_{\text{base}}$: We follow the pre-training settings in BERT (Devlin et al., 2019). Specifically, we train the model on Wikipedia (Devlin et al., 2019) and BookCorpus (Zhu et al., 2015).

- ◇ $\text{SPADE}_{\text{base++}}$: We follow the pre-training settings in RoBERTa (Liu et al., 2019b). Specifically, we train the model on Wikipedia (Devlin et al., 2019), BookCorpus (Zhu et al., 2015), STORIES (Trinh & Le, 2018), CC-News (Liu et al., 2019b), and OpenWebText (Gokaslan et al., 2019).

For language model pre-training and fine-tuning experiments, we use Adam (Kingma & Ba, 2015) as the optimizer. Hyper-parameters for pre-training are detailed in Table 8; and hyper-parameters for fine-tuning are detailed in Table 9.

C.2 Long Range Arena

We follow the model architecture settings in Ma et al. 2022. In all the experiments, we use Adam (Kingma & Ba, 2015) as the optimizer. We use a linear decay learning rate schedule. The rest of the hyper-parameters are detailed in Table 10.

Table 11 shows additional experimental results, where we compare SPADE with existing efficient Transformer models.

Table 8: Hyper-parameters for pre-training.

Parameters	**Base**	**Base++**
Peak Learning Rate	4e-4	2e-4
Batch Size	2,048	2,048
Warmup Steps	10,000	10,000
Total Steps	125,000	2,000,000
Sequence Length	1024	1024
Relative Position Encoding Buckets	32	32
Relative Position Encoding Max Distance	128	128
Adam ϵ	1e-6	1e-6
Adam (β_1, β_2)	(0.9, 0.98)	(0.9, 0.98)
Clip Norm	–	1.0
Dropout	0.1	0.1
Weight Decay	0.01	0.01

Table 9: Hyper-parameters for fine-tuning.

Parameters	**Range**
Learning Rate	{2e-5, 4e-5, 5e-5, 1e-4}
Batch Size	{16, 32}
Maximum Training Epochs	{3, 5, 10}
Dropout	0.1
Warmup Step Rate	0.1
Weight Decay	0.1

C.3 Language Modeling

We follow the settings in Baevski & Auli 2019, including model architecture and hyper-parameters. For the efficient Transformer variants, we set the window size to 512 when using window attention, and we set the chunk size to 512 when using chunk attention.

Table 10: Hyper-parameters for training on LRA.

Task	Batch Size	Learning Rate	Weight Decay	Dropout	Clip Norm	Chunk Size
Listops	64	0.0015	0.0	0.2	1.0	128
Text	100	0.01	0.01	0.2	1.0	128
Retrieval	128	0.004	0.03	0.1	1.0	128
Image	100	0.01	0.02	0.0	1.0	128
Pathfinder	128	0.01	0.01	0.0	1.0	128
Path-X	16	0.01	0.01	0.0	1.0	1024

Table 11: Experimental results on Long Range Arena (LRA). Path-X uses 16k as the input sequence length, and "✗" indicates unavailable results due to computational constraints. All the baseline results, except for MEGA-chunk, are from Gu et al. (2022b). MEGA-chunk results are from Ma et al. (2022).

Dataset	**Listops**	**Text**	**Retrieval**	**Image**	**Pathfinder**	**Path-X**	**Avg.**
Sequence length	2k	4k	8k	1k	1k	16k	—
Random	10.00	50.00	50.00	10.00	50.00	50.00	36.67
Transformer (full) (Vaswani et al., 2017)	36.37	64.27	57.46	42.44	71.40	✗	53.66
Transformer (window)	15.82	52.98	53.39	41.46	66.63	✗	46.71
Sparse Trans. (Child et al., 2019)	17.07	63.58	59.59	44.24	71.71	✗	51.03
Longformer (Beltagy et al., 2020)	35.63	62.85	56.89	42.22	69.71	✗	52.88
Linformer (Wang et al., 2020b)	35.70	53.94	52.27	38.56	76.34	✗	51.14
Reformer (Kitaev et al., 2020)	37.27	56.10	53.40	38.07	68.50	✗	50.56
Sinkhorn Trans. (Tay et al., 2020)	33.67	61.20	53.83	41.23	67.45	✗	51.23
Synthesizer (Tay et al., 2021a)	36.99	61.68	54.67	41.61	69.45	✗	52.40
BigBird (Zaheer et al., 2020)	36.05	64.02	59.29	40.83	74.87	✗	54.17
Linear Trans. (Katharopoulos et al., 2020)	16.13	65.90	53.09	42.34	75.30	✗	50.46
Performer (Choromanski et al., 2021)	18.01	65.40	53.82	42.77	77.05	✗	51.18
FNet (Lee-Thorp et al., 2022)	35.33	65.11	59.61	38.67	77.80	✗	54.42
Nystromformer (Xiong et al., 2021)	37.15	65.52	79.56	41.58	70.94	✗	57.46
Luna-256 (Ma et al., 2021)	37.25	64.57	79.29	47.38	77.72	✗	59.37
FMMformer (Nguyen et al., 2021)	36.74	67.84	81.88	45.10	72.12	✗	60.74
S4 (Gu et al., 2022b)	58.35	76.02	87.09	87.26	86.05	88.10	80.48
MEGA-chunk (Ma et al., 2022)	58.76	90.19	90.97	85.80	94.41	93.81	85.66
SPADE (window)	59.70	87.55	90.13	**89.11**	**96.42**	94.22	86.19
SPADE (chunk)	**60.50**	**90.69**	**91.17**	88.22	96.23	**97.60**	**87.40**