LOCOST: State-Space Models for Long Document Abstractive Summarization

Florian Le Bronnec[*,1,2], **Song Duong**[*,1,6], **Mathieu Ravaut**[3,4], **Alexandre Allauzen**[2], **Nancy F. Chen**[3], **Vincent Guigue**[5], **Alberto Lumbreras**[6], **Laure Soulier**[1], **Patrick Gallinari**[1,6]

[1]Sorbonne Université, CNRS, ISIR, F-75005 Paris, France
[2]Miles Team, LAMSADE, Université Paris-Dauphine, Université PSL, CNRS, 75016 Paris, France
[3]Institute of Infocomm Research (I2R), A-STAR, Singapore
[4]Nanyang Technological University, Singapore
[5]AgroParisTech, UMR MIA-PS, Palaiseau, France
[6]Criteo AI Lab, Paris, France

Abstract

State-space models are a low-complexity alternative to transformers for encoding long sequences and capturing long-term dependencies. We propose LOCOST: an encoder-decoder architecture based on state-space models for conditional text generation with long context inputs. With a computational complexity of $\mathcal{O}(L \log L)$, this architecture can handle significantly longer sequences than state-of-the-art models that are based on sparse attention patterns. We evaluate our model on a series of long document abstractive summarization tasks. The model reaches a performance level that is 93-96% comparable to the top-performing sparse transformers of the same size while saving up to 50% memory during training and up to 87% during inference. Additionally, LOCOST effectively handles inputs exceeding *600K* tokens at inference time, setting new state-of-the-art results on full-book summarization and opening new perspectives for long input processing.

1 Introduction

Nowadays the design of efficient models for long texts remains an open challenge despite the recent progress achieved in natural language processing (NLP). The introduction of transformer architectures (Vaswani et al., 2017) indeed came as a major bump in performance and scalability for text generation. However the quadratic complexity in the input length still restricts the application of large pretrained models to long texts. For instance, BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) are limited to a context size of 512 and 1024 tokens respectively, which amounts to 2-3 paragraphs of standard text.

To mitigate this issue, a straightforward approach is to leverage sparse-attention patterns (Child et al., 2019) to better cope with long texts.

[*]Authors contributed equally to this work. Corresponding authors: `florian.le-bronnec@dauphine.psl.eu`, `s.duong@criteo.com`

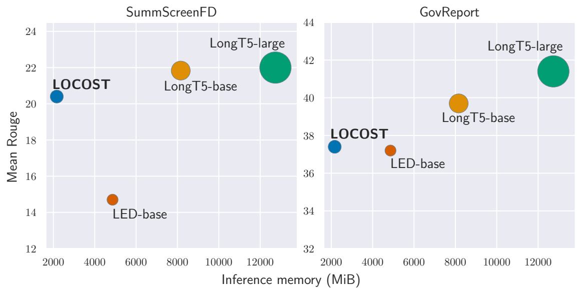


Figure 1: Mean ROUGE score with inference memory usage on long-document summarization with input length 16K (left: SummScreenFD dataset, right: GovReport dataset). The size of the circles represents the training memory usage. LOCOST demonstrates competitive performances compared to state-of-the-art sparse transformers of the same size, while being significantly more memory-efficient at both training and inference.

As key examples, Guo et al. (2022) and Zaheer et al. (2020) extended the context capacity of encoder-decoder models (Raffel et al., 2020; Zhang et al., 2020) and showed drastic increases in the performance on long text summarization, motivating the quest to incorporate longer contexts. However, in practice, even the best sparse-transformers need heavy computational resources to handle sequences of length larger than 8K tokens (see Figure 4).

Deep state-space models (SSMs) (Gu et al., 2022b) have been proposed for sequence processing, with complexity $\mathcal{O}(L \log L)$, initially for computer vision and audio and more recently for text. Their recurrent architectures are designed for capturing long-range dependencies (Gu et al., 2020). Up to now, their applications have been restrained to either unconditional autoregressive generation, i.e., with a decoder-only (Fu et al., 2023; Goel et al., 2022) ; or sequence classification, i.e., with an encoder-only (Gu et al., 2022b,a; Nguyen et al., 2022). Tackling conditional text generation with SSMs as required e.g. for summarization remains yet unexplored.

In this paper, we propose LOCOST an encoder-decoder architecture to explore the performance of SSMs for conditional text generation tasks, through the lens of abstractive summarization. We demonstrate that SSMs can be competitive with transformer-based models while drastically reducing their memory requirements. We opt for a *lightweight* architecture design, comparable to the average base transformers (roughly 250M parameters) in order to process extremely long sequences on standard compute resources. Our experimentations with extremely long sequences yield state-of-the-art results on the challenging BookSum-Book. With an increase of up to 2 points in average ROUGE score compared to sparse attention baselines, our model is able to process entire books, without truncation, and on a single GPU. Our contributions are threefold:

- We propose a new encoder-decoder architecture based on state-space models. By bypassing the self-attention mechanism used in transformers, the model enjoys a complexity of $\mathcal{O}(L \log L)$ instead of $\mathcal{O}(L^2)$ as in traditional transformers.
- Compared with the best-performing sparse transformers of the same size, the model achieves 93-96% of the best performance on various long document abstractive summarization while being up to 50% more memory-efficient during training and up to 87% at inference time, see Figure 1.
- The model is able to process entire input sequences of up to ***600K tokens***, a length far out of reach for sparse transformers. This allows the model to achieve a new state-of-the-art on a challenging full-book summarization task.

To the best of our knowledge, this is the first encoder-decoder that performs competitively with sparse transformers with no attention in the encoder. Furthermore, this work represents the first successful attempt at processing extremely long texts e.g. entire books without any truncation, all in a single pass. The proposed model opens new perspectives for addressing long texts with lesser resources.[*]

2 Related Work

In this section, we first review memory-efficient transformers and existing alternatives to the attention mechanism. Then, we discuss recent literature on state-space models.

[*]Code and checkpoints available at `https://github.com/flbbb/locost-summarization`.

Memory efficiency for transformers. Reducing the memory consumption of transformers is an active research field. Optimization at the hardware level (Dao et al., 2022) helped to improve the scaling of the attention computation on recent GPUs. A line of work considers retrieving-augmented transformers, like (Borgeaud et al., 2022; Wang et al., 2023), that use additional modules to enhance the language modeling backbone. While crucial in developing memory-efficient architectures, we consider these last two topics as being orthogonal to our work that focuses on the models' architecture. Profuse literature focuses on tailoring the models' architecture for long inputs. Since the computational complexity of attention comes from the computation of the self-attention matrix, a straightforward way to reduce its cost is to approximate it using sparse-attention patterns. These patterns typically incorporate a combination of local attention and a set of carefully selected tokens. For instance, in addition to global tokens, BigBird (Zaheer et al., 2020) considers random tokens, while LSG (Condevaux and Harispe, 2023) considers sparse tokens through various strategy of sparsification. LongT5 (Guo et al., 2022) chunks the sequence into blocks and averages their representations, which gives a number of global tokens equal to the number of blocks. An overview of the complexity of various sparse-transformers can be found in Table 1.

In contrast, we propose an alternative, computationally efficient architecture, without the need of costly self-attention blocks nor sparse-attention patterns.

Attention-free transformers. Some variants of transformers already avoid the standard attention mechanism. For example Katharopoulos et al. (2020); Hua et al. (2022) approximate the softmax similarity in the attention by a more efficient computation. More recently, mixing architectures were introduced in (Liu et al., 2021). They are the main component of the FNet (Lee-Thorp et al., 2022) model, an encoder that replaces self-attention with a Discrete Fourier Transform (DFT). FNet has a complexity of $\mathcal{O}(L \log L)$ and is an encoder-only model, thus restricted to classification and regression tasks.

Our proposed model also bypasses attention in the encoder, reaching the same computational complexity as encoders such as FNet, while being a much more versatile model, specifically designed for conditional text generation.

Encoder architecture	Complexity per layer
Transformer (full)	$\mathcal{O}(L^2)$
LED	$\mathcal{O}(Lw)$
BigBird	$\mathcal{O}(Lw + L(g+r))$
LSG	$\mathcal{O}(Lw + L(g+s))$
LongT5 (TGlobal)	$\mathcal{O}(Lw + L\lfloor L/c \rfloor)$
LOCOST	$\boldsymbol{\mathcal{O}(L\log(L))}$

Table 1: Computational complexity per encoder layer as a function of the input length L, the local window size w (typically set to 256 tokens), the number of global tokens g, random tokens r, sparse tokens s and the chunk size c. LOCOST has a much lower complexity than other sparse-attention baselines.

State-space models (SSMs). Deep learning implementations of SSMs consist of emerging architectures, first presented in (Gu et al., 2020). These architectures are particularly appealing for processing long sequences due to their reduced complexity compared to transformers, and their stronger theoretical guarantees compared to RNNs (Gu et al., 2022b), more details in Section 3. In practical applications, SSMs have found success in both classification and unconditional autoregressive generation for language modeling. Gu et al. (2022b) proposed a classification model that significantly improved the Long-Range Arena benchmark (Tay et al., 2021), which includes classification tasks involving images, synthetic sequences, and texts. Other studies have applied SSMs to video classification (Nguyen et al., 2022) and text classification (Wang et al., 2022). Regarding language modeling, many researchers have leveraged the natural causal formulation of SSMs, employing a decoder-only architecture for tasks like audio generation (Goel et al., 2022) and, more recently, autoregressive language modeling (Fu et al., 2023).

In this work, we tackle the more challenging task of conditional text generation and study the performance of SSMs, used as an encoder-decoder architecture, on long document abstractive summarization. With our proposed architecture, we demonstrate the abilities of our model to process input sequences of up to 600K tokens, while being competitive to sparse-transformers on long document abstractive summarization.

3 Background

For contextualization, we leverage state-space models instead of self-attention. Throughout the paper, L denotes the sequence length, H the embedding dimension and N the dimension of the state-space hidden state (to be introduced in Section 3). Before delving into our model in Section 4, we describe below the main components of the state-space architecture and elaborate on their potential for long sequence processing.

State-space models. For unidimensional inputs $\boldsymbol{u} = (u_0, ..., u_{L-1}) \in \mathbb{R}^L$, deep SSMs (Gu et al., 2022b) are based on the recurrent equation:

$$\begin{cases} \boldsymbol{x}_{j+1} = \boldsymbol{A}\boldsymbol{x}_j + \boldsymbol{b}u_{j+1}, \\ y_{j+1} = \boldsymbol{c}^\top \boldsymbol{x}_{j+1} + du_{j+1}, \end{cases} \tag{1}$$

where $\boldsymbol{x}_j$ is the SSM hidden state and y_j the output of the SSM. The state matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ carries and transforms the hidden state through the iterations along with $\boldsymbol{b} \in \mathbb{R}^N$, $\boldsymbol{c} \in \mathbb{R}^N$, and $d \in \mathbb{R}$ which are learned parameters.

State-space convolution. By unrolling the recurrence above, the output sequence $\boldsymbol{y} \in \mathbb{R}^L$ can be expressed as: $y_j = \sum_{l=0}^{j} \boldsymbol{c}^\top \boldsymbol{A}^{j-l}\boldsymbol{b}u_l + du_j$, $\forall l \in \{1, ..., L\}$. Let $*$ denote the causal convolution operator (details about this operator are in Appendix A). Then, we can define a convolution kernel $\boldsymbol{\kappa} \in \mathbb{R}^L$ that depends on $\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}$. A SSM layer is therefore parametrized by $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$, d through $\boldsymbol{\kappa}$ and its output is defined by $\boldsymbol{y}$ as in the following equation:

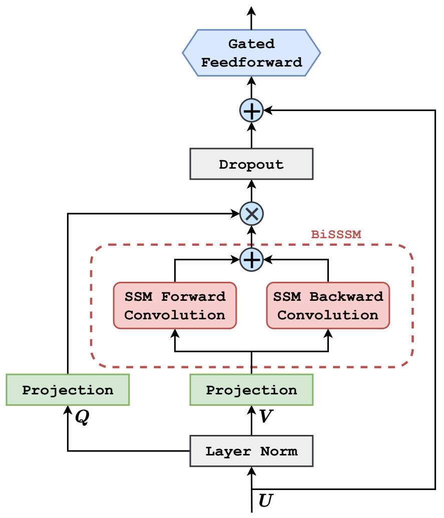
$$\begin{cases} \boldsymbol{y} = \boldsymbol{\kappa} * \boldsymbol{u} + d\boldsymbol{u}, \\ \boldsymbol{\kappa} = \left(\boldsymbol{c}^\top \boldsymbol{b}, \boldsymbol{c}^\top \boldsymbol{A}\boldsymbol{b}, \ldots, \boldsymbol{c}^\top \boldsymbol{A}^{L-1}\boldsymbol{b}\right). \end{cases} \tag{2}$$

For multidimensional $\boldsymbol{u} \in \mathbb{R}^{L\times H}$, we simply compute H convolutions with one kernel $\boldsymbol{\kappa}_h$ for each dimension.

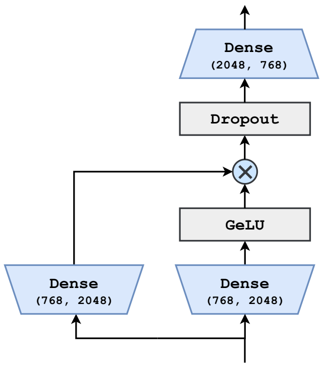
SSMs efficiency. Due to the linear time-dependency between hidden states, as shown in Equation (1), we can compute the whole output $\boldsymbol{y}$ directly as a convolution, without iteration over the time dimension, as opposed to RNNs. A naive implementation of (2) would incur a quadratic complexity in the input length L, matching the complexity of transformers and thus be prohibitive for long sequences. However, thanks to the FFT, this computation can be performed in $\mathcal{O}(L\log L)$ (see Appendix A for more details).

4 Model

In this section, we present the LOCOST model. We first introduce the bidirectional deep state-space model, then show how to use it to enable global contextualization of the tokens. Then, we present



(a) The LOCOST layer.



(b) Gated feedforward net.

Figure 2: The embedded sequence is contextualized via a gated bidirectional SSM before passing through a gated feedforward net.

the architecture of the LOCOST layer with an efficient contextualization that can be used as a drop-in replacement for the self-attention mechanism in transformers.

4.1 Capturing local and global contexts

Intuition. In deep SSMs, information from previous tokens flows up to the current token through the hidden states $\boldsymbol{x}$. The convolution view provides another angle: each output $\boldsymbol{y}_j$ is a weighted sum of the previous tokens $\boldsymbol{u}_0, \dots, \boldsymbol{u}_j$, whose weights are given by $\boldsymbol{\kappa}$.

Bidirectional contextualization. To aggregate information from both directions, we consider bidirectional convolutions. A first kernel, $\overleftarrow{\boldsymbol{\kappa}}$ performs the regular causal convolution $\overleftarrow{\boldsymbol{\kappa}} * \boldsymbol{u}$. A second kernel $\overrightarrow{\boldsymbol{\kappa}}$ is used to compute the cross-correlation with $\boldsymbol{u}$. The results of these two operations are summed out (similar to bi-recurrent encoder). The overall

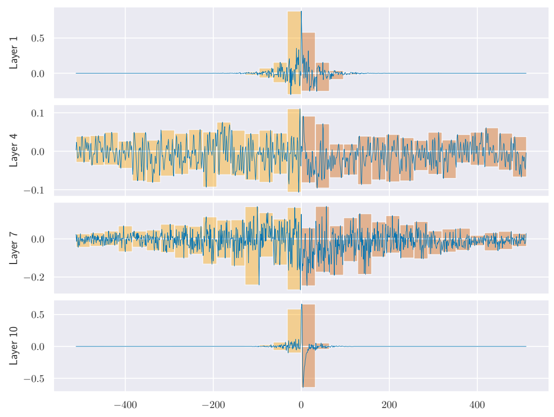


Figure 3: Visualization of the kernels corresponding to the first dimension for several layers of the pre-trained model. Bins show the average decay of the forward and backward kernels. This illustrates the different scales of each kernel. Layers 1 and 10 capture short and extra-short range contextualizations, while Layers 4 and 7 model extra-long and long contexts, respectively.

operation is described by the following equation:

$$\begin{aligned}\boldsymbol{y}_j &= \sum_{l\le j} \overleftarrow{\boldsymbol{\kappa}}_{j-l}\odot \boldsymbol{u}_l + \sum_{l\ge j}\overrightarrow{\boldsymbol{\kappa}}_{l-j}\odot \boldsymbol{u}_l + \boldsymbol{d}\odot\boldsymbol{u}_j \\ &= \mathrm{BiSSM}(\boldsymbol{U})_j. \qquad (3)\end{aligned}$$

In this equation, $\boldsymbol{U} \in \mathbb{R}^{L\times H}$ is the embedding matrix of the input text: $(\boldsymbol{u}_0, \dots, \boldsymbol{u}_{L-1})$. The kernels $\overrightarrow{\boldsymbol{\kappa}}, \overleftarrow{\boldsymbol{\kappa}}$ are computed as in Equation (2), with their respective parameters $(\overrightarrow{\boldsymbol{A}}, \overrightarrow{\boldsymbol{c}}, \overrightarrow{\boldsymbol{b}})$ and $(\overleftarrow{\boldsymbol{A}}, \overleftarrow{\boldsymbol{c}}, \overleftarrow{\boldsymbol{b}})$. The element-wise product is denoted by $\odot$ and we consider multidimensional inputs, with one kernel per dimension.

The output $\boldsymbol{y}_j$ is now contextualized as a weighted sum of previous $\boldsymbol{u}_{\le j}$ and subsequent $\boldsymbol{u}_{\ge j}$ inputs. For scalar inputs, more insights on how far in the future or in the past a scalar input u_l contributes to the scalar output y_j are given by the spectral radii $\rho(\overrightarrow{\boldsymbol{A}})$ and $\rho(\overleftarrow{\boldsymbol{A}})$. Indeed the sensitivity of an output y_j with respect to an input u_l is bounded by the following quantity:

$$\left|\frac{\partial y_j}{\partial u_l}\right| \le \begin{cases} \rho(\overleftarrow{\boldsymbol{A}})^{j-l}|\overleftarrow{\boldsymbol{c}}^\top \overleftarrow{\boldsymbol{b}}| & \text{if} \quad l < j, \\ \rho(\overrightarrow{\boldsymbol{A}})^{l-j}|\overrightarrow{\boldsymbol{c}}^\top \overrightarrow{\boldsymbol{b}}| & \text{if} \quad l > j. \end{cases}$$

For multidimensional inputs, using a state-space kernel for each dimension enables a fine-grained adjustment of the spectral radii independently for each of them. A small value corresponds to modeling local contexts, while a large value captures global ones.

Some of the corresponding kernel weights of this convolution can be visualized on Figure 3. A more complete visualization can be found in Appendix C.

4.2 Architecture

Encoder. Our encoder consists of a stack of LOCOST layers, illustrated in Figure 2a. It is computed as follows:

- Embedding matrix $\boldsymbol{U} \in \mathbb{R}^{L\times H}$ is first projected onto $\boldsymbol{Q}, \boldsymbol{V} \in \mathbb{C}^{L\times H}$.
- $\boldsymbol{V}$ is contextualized through a BiSSM.
- A pointwise multiplication $\boldsymbol{Q} \odot \mathrm{BiSSM}(\boldsymbol{V})$ acts as a first gate before passing the output through a feedforward layer.
- This feedforward layer employs a second gating mechanism (see Figure 2b). For this component, we use gated GeLU that has shown to be efficient by Shazeer (2020).

The architecture of the LOCOST layer (Figure 2a) resembles that of a transformer layer except that the self-attention mechanism is replaced by a gated bidirectional state-space model. We follow Gu et al. (2022a) for the parametrization and initialization of the state-space models (more details in Appendix E).

Decoder. Since our focus is on long input summarization, the generation output length is very short compared to the input. For decoding, we follow the practice of other efficient architectures (Zaheer et al., 2020; Beltagy et al., 2020; Guo et al., 2022) and use a vanilla transformer decoder equipped with dense self- and cross-attention. A full description of hyperparameters of the model is provided in Appendix B.

Complexity. The LOCOST layer takes $\mathcal{O}(H^2L + HNL + HL\log L)$ time and $\mathcal{O}(HNL)$ space to compute. We refer to Appendix D for more details.

5 Experiments

To validate our experiments, we focus on the long document abstractive summarization task as it represents a typical conditional generation problem with long input requirements.

5.1 Experimental setup

Approach. We evaluate LOCOST following a classical pre-training then fine-tuning approach. For fine-tuning, we used the official train, validation and test splits of each dataset. We train all models until convergence and select the best model based on the validation Mean ROUGE (mean of ROUGE-1/2/LSum) for test evaluation.

Metrics. We evaluate LOCOST both with reference-based and reference-free metrics. For reference-based summarization evaluation, we use the traditional n-gram overlap summarization metrics **ROUGE-1/2/Lsum** (Lin, 2004). We average them into a single score to compare with other baselines. We also report **BERTScore** (BS) (Zhang* et al., 2020), a model-based metric. For reference-free evaluation, we report the **BLANC** (BL) score (Vasilyev et al., 2020), a metric that has been shown to correlate well with human evaluations. We also assess the throughput (samples per second) and the memory usage (MiB of GPU RAM) of LOCOST compared with other state-of-the-art sparse transformers.

Inference. In all of our experiments, we intentionally favored simplicity and opted for greedy decoding.

5.2 Pre-training

Pre-training objective. To pre-train the model, we leverage the gap-sentences generation (GSG) unsupervised pre-training objective, which was introduced by PEGASUS (Zhang et al., 2020) and is well-suited for sequence-to-sequence generation. Unlike BART (Lewis et al., 2020) or T5 (Raffel et al., 2020) pre-training objectives, GSG endows the model with zero-shot summarization capabilities. GSG was successfully applied by subsequent generation models such as LongT5 (Guo et al., 2022) and PEGASUS-X (Phang et al., 2022). Namely, a document D is split into its M sentences: $D = \{s_1, \dots, s_M\}$. Given a ratio α, GSG then identifies $K = \lfloor \alpha M \rfloor$ sentences from D that maximize the ROUGE-1 (noted R-1) with the rest of the document:

$$U = \arg\underset{j}{\text{top-K}}\ R\text{-}1\Big(\bigcup_{i\neq j}\{s_i\}, s_j\Big) \tag{4}$$

The resulting subset $U \subseteq \{1, \dots, M\}$ splits the document into a pseudo summary $\hat{Y} = \{s_i\}_{i\in U}$ and a pseudo-source $\hat{D} = \{s_i\}_{i\notin U}$, which are used for pre-training with the standard cross-entropy loss.

Pre-training data. We pre-train the model exclusively on the C4 dataset (Raffel et al., 2020), in BF16 for 1M steps, using an input sequence length of 4,096 and an output sequence length of 910.

Pre-training optimization. The learning rate scheduler we use is identical to T5, employing an inverse square root function, with the warm-up steps set to 10,000. We set the GSG-ratio $\alpha = 0.2$ and do not employ dropout during this phase. We follow closely the same pre-training as LongT5 (Guo et al., 2022).

5.3 Fine-tuning

Fine-tuning datasets. We evaluate LOCOST on a series of long-input abstractive summarization tasks. A table of statistics for all the datasets can be found in Appendix F.

- **arXiv** (Cohan et al., 2018) Articles extracted from *arXiv* using the core body document as the input sequence and the abstract as the target sequence.
- **PubMed** (Cohan et al., 2018) Similar to arXiv, but articles come from *PubMed*, a medical database.
- **GovReport** (Huang et al., 2021) A long-document summarization dataset of US government reports with their executive summaries.
- **SummScreenFD** (Chen et al., 2022) A long-document summarization dataset of TV series transcripts of entire episodes with human-written recaps of the episodes.
- **BookSum (-Chapter & -Book)** (Kryscinski et al., 2022) A collection of chapters from various books with a summary for each of them. We also consider the book-level version where the model has to summarize entire books.

Fine-tuning optimization. We fine-tune in BF16 using a constant learning rate of 5×10^{-4} and a dropout rate of 0.1 for all datasets. We experiment with lengths ranging from 4,096 to 32,768 for the input and 512 for the output, except for GovReport and BookSum-Book where we use 1024.

Baselines. We consider both competitive sparse transformers, including LED (Beltagy et al., 2020), BigBird (Zaheer et al., 2020), LongT5 (Guo et al., 2022) and LSG (Condevaux and Harispe, 2023), as well as dense encoder-decoders like BART (Lewis et al., 2020), T5 (Raffel et al., 2020) and PEGASUS (Zhang et al., 2020). For a fair comparison, **we only compare to sparse transformers architectures of equivalent size (roughly 250M parameters)**.

5.4 Results

Long-input summarization. Table 2 and 3 present our experimental results. Across all datasets, LOCOST reaches up to 96% of state-of-the-art Mean ROUGE while being up to 3 times more memory-efficient than the best model LongT5 during both training and inference for 16K long inputs, e.g. on GovReport or SummScreenFD. The model is also twice as efficient as the local-attention transformer LED and up to 17 times more efficient than dense transformer BART at inference time. LOCOST significantly improves Mean ROUGE over LED and BigBird on all datasets while performing competitively with respect to LSG. On all datasets, the results for LongT5 and LED have been obtained by fine-tuning from pre-trained checkpoints, following recommended configurations in (Guo et al., 2022) and (Beltagy et al., 2020) respectively. The results for BigBird has been reported from the original paper. LSG results are obtained from evaluating the publicly fine-tuned checkpoints on arXiv and PubMed and from our fine-tuning on BookSum-Chapter. GovReport and SummScreenFD results are reported from the SCROLLS test leaderboard (Shaham et al., 2022).

Throughput and Memory usage. We measure the memory consumption of T5, LED, LongT5 and LOCOST on input lengths ranging from 1K to 500K tokens, at training and inference time. Results are presented on Figure 4. Compared to LongT5, the best-performing baseline, LOCOST is able to process up to $2\times$ longer sequences during training and $16\times$ longer at inference time. This correlates also with a higher throughput during both training and inference, as shown in Table 4.

Qualitative evaluation: GPT-3.5 preference. Since our input texts are very long, performing a full human-based evaluation would be very costly and time consuming. Instead, we perform a mock human evaluation using GPT-3.5 [*]. This practice has been used and has shown success in summary evaluation (Shen et al., 2023; Gilardi et al., 2023; Chiang and Lee, 2023). We ask the model to rate the generated summary on four dimensions: *relevance*, *consistency*, *fluency*, and *coherence*. More

[*]We use *gpt-3.5-turbo-16k* model for evaluation.

	arXiv				PubMed				BookSum-Chapter			
Model	R-1 / R-2 / R-L	BS	BL	%	R-1 / R-2 / R-L	BS	BL	%	R-1 / R-2 / R-L	BS	BL	%
PEGASUS$_{\text{base}}$	34.8 / 10.2 / 22.5	–	–	64.5	40.0 / 15.2 / 25.2	–	–	35.1	–	–	–	–
LED$_{\text{base}}$	41.3 / 16.4 / 37.7	60.1	10.6	91.3	42.9 / 19.6 / 39.7	63.1	15.9	89.2	17.3 / 2.6 / 15.7	41.7	4.4	46.3
BigBird$_{\text{base}}$	41.2 / 16.4 / 37.0	–	–	90.4	43.7 / 19.3 / 39.9	–	–	89.8	–	–	–	–
LSG$_{\text{base}}$	43.6 / 17.4 / 39.8	62.4	10.3	96.4	45.3 / 20.8 / 42.0	65.4	16.3	94.7	31.8 / 6.3 / 30.1	54.1	5.3	89.3
LongT5$_{\text{base}}$	45.2 / 18.4 / 41.0	64.2	11.2	100	47.9 / 22.5 / 44.2	67.3	15.1	100	35.7 / 7.2 / 33.6	56.9	3.9	100
LOCOST	43.8 / 17.0 / 39.7	63.2	10.9	96.1	45.7 / 20.1 / 42.0	65.6	14.7	94.5	34.3 / 6.1 / 32.4	55.4	3.2	95.2

Table 2: Results on arXiv, PubMed and BookSum-Chapter with a input length of 4K, 4K and 8K tokens respectively. % denotes the relative performance on the Mean ROUGE score w.r.t. LongT5, the best performing sparse-transformer at the given size, which is indicated as 100%. BS stands for BERTScore and BL for BLANC.

		GovReport				SummScreenFD					
Model	L	R-1	R-2	R-L	%	R-1	R-2	R-L	%	MEM$_{\text{train}}$	MEM$_{\text{inf}}$
BART$_{\text{base}}$	1K	47.9	18.6	22.7	74.9	27.2	4.9	16.7	74.5	✗	17.6×
LED$_{\text{base}}$	16K	56.2	26.6	28.8	93.7	24.2	4.5	15.4	67.3	1.0×	2.3×
LongT5$_{\text{base}}$	16K	57.7	30.0	31.4	100	34.8	9.6	21.1	100	2.9×	3.8×
LOCOST	16K	56.5	26.8	28.9	94.2	33.4	8.1	19.7	93.5	1.4×	1.0×

Table 3: Results on the test set of SCROLLS for GovReport and SummScreenFD. L denotes the considered input length. % denotes the relative performance on the Mean ROUGE score w.r.t. the reference LongT5. We reported baselines' results from the official SCROLLS test leaderboard. GovReport and SummScreen exhibit challenging long contexts sizes even for sparse transformers, as reported by the memory usage during training (MEM$_{\text{train}}$) and inference (MEM$_{\text{inf}}$) of the different architectures on 16K inputs. ✗ means out-of-memory.

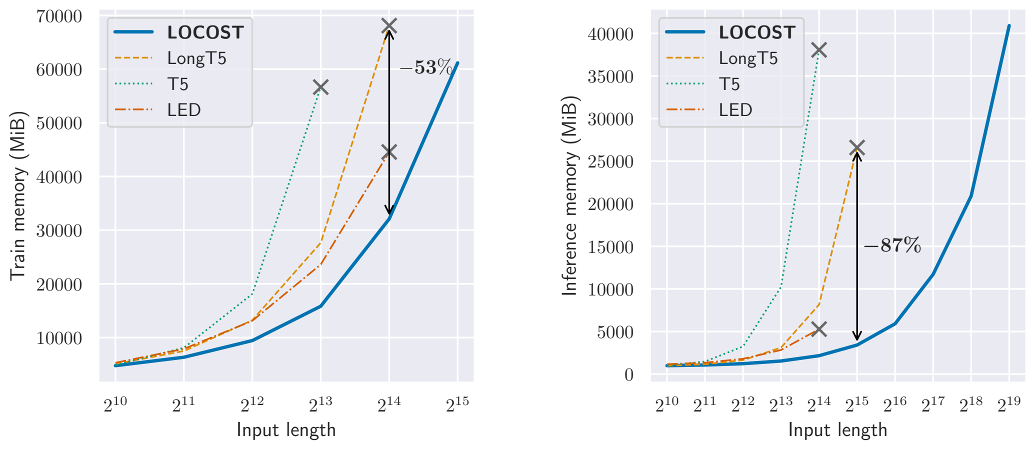


Figure 4: Memory consumption during a typical training (forward + backward) (**left**) and inference iteration (only forward) (**right**). Batch size = 1. Ending cross means out-of-memory or architectural limitations after this point.

details are given in Appendix I.

We perform evaluation on 500 samples randomly taken from PubMed. The results are shown in Table 5. LOCOST produces summaries at a competitive level with respect to LongT5 (93-97%).

5.5 Extrapolating to longer sequences

Because the lengths of the inputs considered during training are often limited due to complexity issues, a desirable property for a model would be to extrapolate at inference time to sequences much longer than the ones used during training.

We train LOCOST on a maximum input length of 4,096 and evaluate it on the test set of arXiv with a maximum input length of 8,192 tokens. As shown in Table 6, this experiment confirms that LOCOST is indeed able to extrapolate to longer sequences than those employed in training. Note that LongT5 leverages relative positional encodings, enabling extrapolation capability. However, as previously mentioned, this comes at the expense of an increased complexity compared to LOCOST. In the next section, we push this idea further by considering extra-long sequences.

5.6 Extra-long sequences: towards full-book summarization

Effect of increasing contexts during training. As shown previously, LOCOST exhibits a strong capability to generalize well on sequences longer

Model	Length 4K		Length 16K	
	Inference (samples/s)	Training (samples/s)	Inference (samples/s)	Training (samples/s)
LED_{base}	3.57	1.69	1.67	0.45
T5_{base}	2.27	1.49	0.34	✗
$\text{LongT5}_{\text{base}}$	2.94	2.94	1.49	0.64
LOCOST	3.03	3.03	1.69	0.81

Table 4: Throughput comparison for different models at 4K and 16K input length.

Model	Rel	Cons	Flu	Coh
$\text{LongT5}_{\text{base}}$	4.6	4.7	3.7	3.7
LOCOST	4.3	4.4	3.6	3.5

Table 5: GPT3.5 evaluation on PubMed with 4K input length using gpt-3.5-turbo-16k. **Rel** stands for *relevance*, **Cons** for *factual consistency*, **Flu** for *fluency* and **Coh** for *coherence*.

Model	L	arXiv-4K	arXiv-8K	Gain (%)
		Mean-R	Mean-R	
$\text{LongT5}_{\text{base}}$	4K	34.8	35.5	2.0
LOCOST	4K	33.5	34.3	2.4

Table 6: Extrapolating to longer sequences experiments. L is the training sequence size. Gain represents the relative Mean ROUGE (Mean-R) improvement from evaluating on 4K to 8K maximum input length. The ROUGE increase asserts that both models are able to generalize to input lengths unseen during training.

than the ones seen during training. Due to the reduced memory usage at both train and inference time, we conduct in this section an analysis of its performances when facing extremely long texts e.g. *summarizing entire books*. We consider the book-level setting of BookSum. We train multiple instances of LOCOST for 100 epochs on truncated books with a context length ranging from 1K to 32K and select the best model on Mean ROUGE on the validation set. We evaluate these models on the test set with untruncated books, and report the results in Figure 5. We found that increasing the input length during training leads to an overall increase in the test Mean ROUGE scores as more contexts are being considered for optimization. Once more, this confirms the generalization capability of LOCOST on extra-long sequence lengths.

Results on full-book summarization. Based on the observations above, we put our best model LOCOST-32K to the test and compare it with LongT5 and current state-of-the-art models on BookSum-Book. For LongT5, we fine-tune the available checkpoint on the maximum possible input length during training (16K) and report its performance on the longest possible input length at inference time (32K). For the other models, the results come from the original papers, in which the models initially produce individual summaries for each paragraph of the book and then rank them according to the model's level of confidence. Results are shown in Table 7. Despite being the model with the least number of parameters, LOCOST achieves state-of-the-art Mean ROUGE compared to LongT5 and even large variants of BART, T5 and PEGASUS. LOCOST is also the only model capable of processing the full documents without truncation and handle sequence lengths of up to 600K tokens. This reveals that effectively processing full contexts without truncation can lead to strong performance enhancement.

[*]For a fair comparison with already existing results, we used ROUGE-L instead of ROUGE-Lsum on BookSum-Book.

Model	#Params	BookSum-Book			
		R-1	R-2	R-L[*]	Mean-R
$\text{BART}_{\text{large}}$	406M	38.7	7.6	13.6	20.0
T5_{large}	737M	**39.9**	8.0	14.0	20.6
$\text{PEGASUS}_{\text{large}}$	568M	36.0	7.2	12.9	18.7
$\text{LongT5}_{\text{base}}$	247M	33.9	7.2	15.6	18.9
LOCOST	234M	38.6	**8.1**	**16.2**	**21.0**

Table 7: Results on BookSum-Book. While being the smallest model, LOCOST achieves state-of-the-art on Mean ROUGE when summarizing entire books.

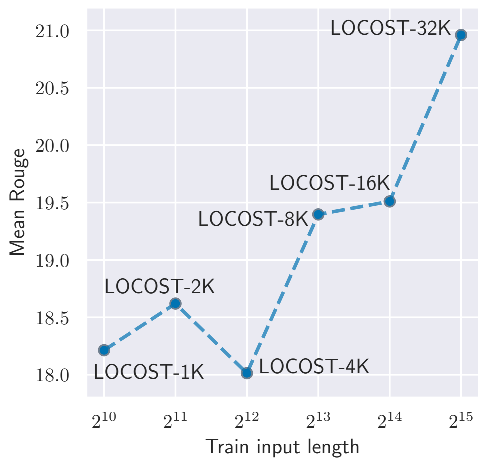


Figure 5: LOCOST trained on increasing sequence lengths evaluated on BookSum-Book dataset *without truncation*, with texts reaching up to 600K tokens.

6 Conclusion

Our paper explores a new encoder-decoder architecture dedicated to handle long input texts. By replacing the self-attention block by SSMs, we design a low complexity and lightweight model able to process long sequences up to 600K tokens at inference time on a single GPU. Our model achieves competitive results on summarization datasets. Moreover, surpassing the limits of existing sparse transformer alternatives, new state-of-the-art results are obtained on the BookSum-Book dataset. To the best of our knowledge, LOCOST is the first model able to process entire books without truncation, all in a single pass. These results offer exciting possibilities for abstractive text-processing tasks requiring extra-long sequences.

7 Limitations

Though we investigated lightweight models for computational reasons, scaling the architecture to a larger size could be studied. We focused on long document abstractive summarization, we leave for future work the study of SSMs on other long inputs abstractive tasks. Although replacing self-attention with state-space encoders drastically reduces the computational complexity, the use of dense cross-attention in the decoder still limits the output sequence length in terms of computation during training.

8 Ethics Statement

We performed pre-training on a subset of the C4 dataset, which has been identified to include inappropriate content like hate speech and explicit material, as noted in the studies conducted by Luccioni and Viviano (2021) and also exhibits negative biases towards certain ethnicities (Dodge et al., 2021). It is important to investigate potential solutions for mitigating these problems through more meticulous preprocessing in order to prevent the emergence of such undesirable attributes in future research. Nevertheless, it is worth mentioning that despite these concerns, the C4 dataset serves as a benchmark within the community, and the reported results solely focus on the quality of the summaries, thereby avoiding any unethical implications. In this paper, we consider a relatively small size for LOCOST. We believe our work could be reproducible with limited resources. We tracked the GPU power consumption during pre-training. The average power usage was 190W per GPU. We trained for 140 hours on 16 GPUs. Given the local CO_2 intensity of 58 gCO_2/kWh [*], we can estimate that approximately 25kg of CO_2 have been emitted during the pre-training, to be compared with the average emissions of 4.6t of CO_2 par capita in 2019[*].

9 Acknowledgements

This work has been partly funded through project ACDC ANR-21-CE23-0007 and ANR-23-PEIA-0008, PEPR IA, project "Principes théoriques et algorithmiques de l'apprentissage frugal (SHARP)". This project was provided with computing AI and storage resources by GENCI at IDRIS thanks to the grants 20XX-AD011014060, 20XX-AD011014022 and 20XX-A0151014638 on the supercomputer Jean Zay's V100/A100 partition. This work has also been partly funded through the Singapore International Pre-Graduate Award (SIPGA).

References

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

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. 2022. SummScreen: A dataset for abstractive screenplay summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8602–8615. Association for Computational Linguistics.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of*

[*]https://www.eea.europa.eu/data-and-maps/daviz/co2-emission-intensity-13/

[*]https://data.worldbank.org/indicator/EN.ATM.CO2E.PC

the Association for Computational Linguistics (Volume 1: Long Papers), pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.

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

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. 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)*, pages 615–621. Association for Computational Linguistics.

Charles Condevaux and Sébastien Harispe. 2023. LSG Attention: Extrapolation of pretrained Transformers to long sequences. In *PAKDD 2023 - The 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Osaka, Japan.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305. Association for Computational Linguistics.

Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Re. 2023. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations*.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30).

Karan Goel, Albert Gu, Chris Donahue, and Christopher Re. 2022. It's raw! Audio generation with state-space models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7616–7633. PMLR.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. Hippo: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems*, volume 33, pages 1474–1487. Curran Associates, Inc.

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

Albert Gu, Karan Goel, and Christopher Re. 2022b. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736. Association for Computational Linguistics.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. 2022. Transformer quality in linear time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9099–9117. PMLR.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. 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*, pages 1419–1436, Online. Association for Computational Linguistics.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Wojciech Kryscinski, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. BOOKSUM: A collection of datasets for long-form narrative summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6536–6558. Association for Computational Linguistics.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2022. 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*, pages 4296–4313, Seattle, United States. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics.

Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. 2021. Pay attention to MLPs. In *Advances in Neural Information Processing Systems*.

Alexandra Luccioni and Joseph Viviano. 2021. What's in the box? an analysis of undesirable content in the Common Crawl corpus. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 182–189. Association for Computational Linguistics.

Eric Nguyen, Karan Goel, Albert Gu, Gordon Downs, Preey Shah, Tri Dao, Stephen Baccus, and Christopher Ré. 2022. S4ND: Modeling images and videos as multidimensional signals with state spaces. In *Advances in Neural Information Processing Systems*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jason Phang, Yao Zhao, and Peter J Liu. 2022. Investigating efficiently extending transformers for long input summarization. *arXiv preprint arXiv:2208.04347*.

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. *Journal of Machine Learning Research*, 21(140):1–67.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. 2022. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021. Association for Computational Linguistics.

Noam Shazeer. 2020. GLU variants improve transformer. *CoRR*, abs/2002.05202.

Chenhui Shen, Liying Cheng, Yang You, and Lidong Bing. 2023. Are Large Language Models Good Evaluators for Abstractive Summarization? *arXiv e-prints*, page arXiv:2305.13091.

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 *International Conference on Learning Representations*.

Oleg Vasilyev, Vedant Dharnidharka, and John Bohannon. 2020. Fill in the BLANC: Human-free quality estimation of document summaries. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 11–20. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Junxiong Wang, Jing Nathan Yan, Albert Gu, and Alexander M Rush. 2022. Pretraining without attention. *arXiv preprint arXiv:2212.10544*.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory. *arXiv preprint arXiv:2306.07174*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Denis Charles, Eren Manavoglu, Tuo Zhao, and Jianfeng Gao. 2022. Efficient long sequence modeling via state space augmented transformer. *arXiv preprint arXiv:2212.08136*.

A Convolution

A.1 Causal convolution

In this section indices of sequence are represented by bracketed numbers. The *causal* convolution between sequences $\boldsymbol{u}, \boldsymbol{\kappa} \in \mathbb{R}^L$ denoted as $*$ presented in section 3 is defined as:

$$(\boldsymbol{\kappa} * \boldsymbol{u})[j] = \sum_{l=0}^{j} \kappa[j-l]u[l]. \tag{5}$$

A.2 Convolution and DFT

We are going to detail the link between convolution and the Discrete Fourier Transform. For that purpose, we need another tool, the *circular convolution*.

Circular convolution. Let's define $\tilde{\boldsymbol{\kappa}}$ the periodized version of $\boldsymbol{\kappa}$ as: $\forall j \in \mathbb{N}$, $\tilde{\kappa}[j] = \kappa[j \bmod L]$. For index $0 \leq j \leq L-1$, the discrete *circular* convolution between $\boldsymbol{u}$ and $\boldsymbol{\kappa}$ is defined as:

$$(\boldsymbol{\kappa} \circledast \boldsymbol{u})[j] = \sum_{l=0}^{L-1} \tilde{\kappa}[j-l]u[l]. \tag{6}$$

Convolution theorem. The convolution theorem states that (the derivation consists only in permuting the $\sum$ symbols):

$$\boldsymbol{\kappa} \circledast \boldsymbol{u} = \mathcal{F}^{-1}\left(\hat{\boldsymbol{\kappa}} \odot \hat{\boldsymbol{u}}\right), \tag{7}$$

where $\hat{\cdot}$ designates the DFT of a sequence and $\odot$ designates the element-wise multiplication.

Causal convolution with DFT. To compute $\boldsymbol{\kappa} * \boldsymbol{u}$ with a DFT, a trick is to pad $\boldsymbol{\kappa}$ and $\boldsymbol{u}$ with L zeros *before* taking their DFT. Indeed, if we replace $\boldsymbol{\kappa}$ and $\boldsymbol{u}$ with their padded versions (hence vectors of $\mathbb{R}^{2L}$) in eq. (6) we see immediately that it coincides with the *causal* convolution (5). This means that using the Fast Fourier Transform (FFT) algorithm, the causal convolution can be computed in $\mathcal{O}(L \log L)$.

B Hyperparameters

The set of hyperparameters used are presented in Table 8.

C Visualisation of learned kernels

A more complete visualization of the learned kernels can be found in Figure 3 and 7.

Parameter	Value
Embedding dimensions H	768
Vocabulary size	32100
Feedforward dimension	2048
Activation function	GeLU
LayerNorm ε	1×10^{-6}
State-space dimension N	256
Number of encoder layers	12
Number of decoder layer	12
Decoder attention heads	12
AdamW (β_1, β_2)	(0.9, 0.999)
AdamW weight decay	0
Pre-training LR schedule	$\frac{2\times10^4}{\sqrt{\max(10^4, \text{current step})}}$
Pre-training dropout	0
Finetuning LR	5×10^{-4}
Finetuning LR schedule	constant
Finetuning dropout	0.1

Table 8: LOCOST hyperparameters.

D Computational complexity of a LOCOST layer

Projection onto $\boldsymbol{Q}$ and $\boldsymbol{V}$ takes $\mathcal{O}(LH^2)$ time and $\mathcal{O}(LH)$ space. Computing the SSM kernel $\boldsymbol{\kappa} = \left(\boldsymbol{c}^\top \boldsymbol{b}, \boldsymbol{c}^\top \boldsymbol{A}\boldsymbol{b}, \ldots, \boldsymbol{c}^\top \boldsymbol{A}^{L-1}\boldsymbol{b}\right)$ takes $\mathcal{O}(LHN)$ time and space. Finally, calculating H convolutions in parallel with DFT takes $\mathcal{O}(LH \log L)$ time.

E State-space models implementation details

Parametrization. We chose to follow the parametrization exposed in (Gu et al., 2022a).

- The multi-dimensional state-tensor[*] $\boldsymbol{A} \in \mathbb{C}^{H\times N\times N}$ is made of H diagonal matrices $\boldsymbol{A}_h = \operatorname{diag}(\boldsymbol{\lambda}_h) \in \mathbb{C}^{N\times N}$.
- For $0 \leq h \leq H$ and $0 \leq n \leq N$, $\boldsymbol{\lambda} \in \mathbb{R}^{H\times N}$ is $\lambda_{h,n} = \exp\left(\Delta_h \lambda_{h,n}^{\text{Re}} + i\Delta_h \lambda_{h,n}^{\text{Im}}\right)$.
- $\boldsymbol{\Delta} \in \mathbb{R}^H$ is a time-scaling parameter.
- We use $N = 256$. Most work chose either $N = 64$ or $N = 256$ (Gu et al., 2022a; Fu et al., 2023). Since increasing N from 64 to 256 did only incur a negligible increase in memory consumption, we chose the latter, with the rationale that it should give more expressive power to $\boldsymbol{\kappa}$.

Initialization. As reported in (Gu et al., 2022a) (see their Table 3), SSMs with special initialization are tailored for long inputs processing. This has

[*]Using parameters in $\mathbb{C}$ gives better expressive power to the convolution, see Gu et al. (2022a) for theoretical and empirical justifications.

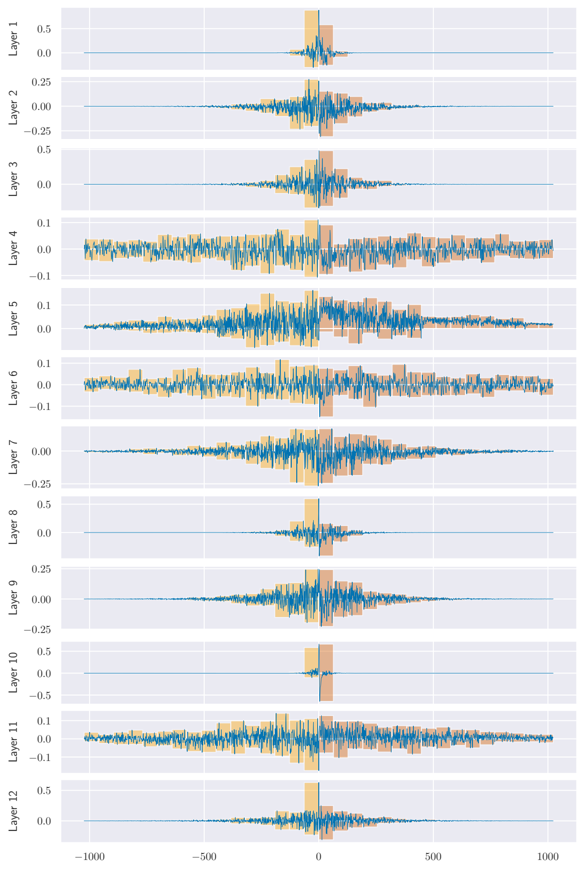


Figure 6: Complete visualization of the kernel of the first dimension of the model through all the 12 layers, includes visualization from Figure 3.

been experimentally confirmed in (Zuo et al., 2022), where they use non-trainable state-space layers to provide long-range contextualization in addition to local attention.

- $\lambda_{h,n}^{\mathrm{Re}}$ is initialized to $-\frac{1}{2}$ and $\lambda_{h,n}^{\mathrm{Im}}$ to πn.
- Δ_h is initialized randomly following $\mathcal{U}([0,1])$.
- $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{C}^{N \times H}$ are initialized randomly following $\mathcal{N}(0,1)$[*].

F Dataset details

Statistics. The statistics of the datasets can be found in Table 9.

License. C4: ODC-BY, arXiv/PubMed: unknown, BookSum: BSD-3-Clause, GovReport: unknown, SummScreenFD: unknown.

Usage. All datasets were solely used for research purposes. Note that they are all in english and we refer to the original publications for more details.

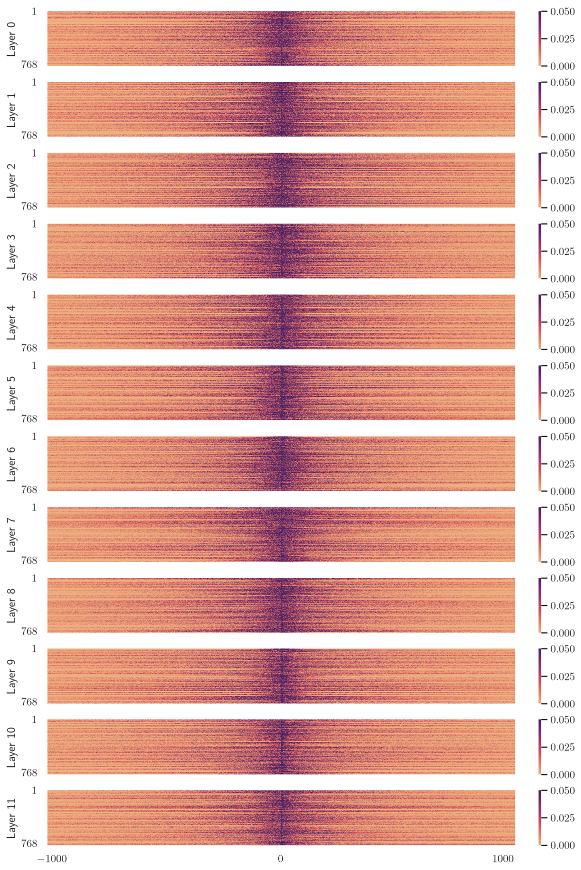


Figure 7: Visualization of the kernel (in absolute value) of size 768×2048 for each of the 12 layers. We clearly show that each layer has kernels of different scales that will model different context ranges.

G Implementation details

Evaluation. For ROUGE score computations, we used the implementation from `https://github.com/google-research/google-research/tree/master/rouge`, released under Apache 2.0 license. BERTScore was computed using the package `https://pypi.org/project/bert-score/` and is released under a MIT license. BLANC using `https://pypi.org/project/blanc/`, released under a MIT license.

Software. Our code is based on Pytorch (Paszke et al., 2019), Huggingface (Wolf et al., 2020) and H3 (Fu et al., 2023). LongT5, LED models and weights are released under the Apache 2.0 license. The license for the LSG model and weights is unknown.

	#Examples per split			Input Length			
Dataset	Train	Validation	Test	Average	Median	Max	90th
arXiv	203,037	6,436	6,440	10,720.18	8,519	378,825	20,170
PubMed	119,924	6,633	6,658	4,747.97	3,883	452,915	8,883
GovReport	17,457	972	973	10,576.06	8,840	240,734	18,834
SummScreenFD	3,673	338	337	9,589.36	9,044	26,447	15,171
BookSum-Chapter	9,600	1,484	1,431	5986.47	4311	204,567	11,804
BookSum-Book	314	45	46	143,562.75	104,381	667,817	305,749

Table 9: Statistics for the summarization datasets. Input length is computed using a SentencePiece tokenizer.

H Sample outputs

Here is a sample summary (gold human abstract + generated summary) from PubMed.

Ground Truth: "Background : the purpose of this study was to perform the creating opportunities for parent empowerment (cope) program for iranian mothers and evaluate its effectiveness on stress, anxiety, and participation of mothers who have premature infants hospitalized in neonatal intensive care units (nicus).materials and methods : a randomized clinical trial was conducted with 90 mothers of premature infants hospitalized in the educational neonatal nicus of state hospitals affiliated to isfahan university of medical sciences. For measuring the variables, state - trait anxiety inventory, the parental stressor scale : neonatal intensive care, and the index of parental participation / hospitalized infant were used. Intervention group received two phases of cope program. This program consisted of information and behavioral activities about the characteristics of premature infants. Sessions interval was from 2 to 4 days. Stress and anxiety were measured three times (before each phase and 2 - 4 days after the second phase). Mothers participation was evaluated 2 - 4 days after the second phase. The t - test, 2, mann - whitney u test, and repeated measurement test were used for data analysis. Results : mothers in the intervention group reported significantly less anxiety and less stress in the nicu after performing each phase of the cope program (p 0.001), whereas at this time, the level of stress in the comparison group increased. Also, cope mothers participated in their infants' care rather than the mothers in the comparison group (p 0.001). Conclusion : cope program was effective for iranian mothers. This study shows that irrespective of any culture, giving early and comprehensible educational - behavioral information may have positive effect on maternal psychological condition and maternal infant interaction."

LOCOST: "Background : the premature birth of an infant and the following neonatal intensive care cause psychological distress and can have a traumatizing effect on parents. The neonatal intensive care unit (nicu) environment has the potential to exacerbate stress for parents. The purpose of this study was to create opportunities for parent empowerment (cope) program in this survey.materials and methods : this is a quasi - experimental study. The study population consisted of 90 mothers with infants hospitalized in the nicu of isfahan, iran. The cope program included information about the behavior of infants, parenting role, and parenting stressor scales. Data were collected through demographic and infant variables questionnaire, state anxiety inventory, and parental stressor scale at the first contact point. Then, the cope program was given to the mothers. Data were analyzed by spss version 16 using descriptive statistics.results:the study resulted in the Absence of any significant difference in the state anxiety inventory and the parental stressor scale results. Moreover, the mean of maternal state anxiety inventory in the comparison group was significantly higher than the intervention group (p 0.001). The parental stressor scale results showed significant difference between the two groups (p 0.001). The state anxiety inventory showed significant difference between the two groups (p 0.001). The parental stressor scale results showed significant difference between the two groups (p 0.001). Conclusions : the cope program was feasible and the mother - infant positive interactions were high. Therefore, this study can be done by providing the parents with strategies that support them."

I GPT-3.5 evaluation

As a proxy for subjective assessment, we make use of *gpt-3.5-turbo-16k* as a mock human evaluator. We ask the model to rate the generated summary on four dimensions: *relevance*, *consistency*, *fluency*, and *coherence* with the following prompt:

"Imagine you are a human annotator now.

You will evaluate the quality of summaries written for an article. Please follow these steps: Carefully read the article, and be aware of the information it contains. Read the proposed summary. Rate the summary on four dimensions: relevance, consistency, fluency, and coherence. You should rate on a scale from 1 (worst) to 5 (best). Definitions are as follows:

Relevance: The rating measures how well the summary captures the key points of the article. Consider whether all and only the important aspects are contained in the summary.

Consistency: The rating measures whether the facts in the summary are consistent with the facts in the original article. Consider whether the summary does reproduce all facts accurately and does not make up untrue information.

Fluency: This rating measures the quality of individual sentences, whether they are well-written and grammatically correct. Consider the quality of individual sentences.

Coherence: The rating measures the quality of all sentences collectively, to fit together and sound natural. The article and the summary are given below:

Article: **{insert article}**

Summary: **{insert summary}**.

Rate the summary in the following format:

Relevance:

Consistency:

Fluency:

Coherence:”