LOCOST: State-Space Models for Long Document Abstractive Summarization

Florian Le Bronnec∗,1,2, Song Duong∗,1,6, Mathieu Ravaut3,4, Alexandre Allauzen2, Nancy F. Chen3, Vincent Guigue5, Alberto Lumbrañas6, Laure Soulier1, Patrick Gallinari1,6

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 $O(L \log L)$, 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.

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 $O(L \log L)$, 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-deco...
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 experiments 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 $O(L \log L)$ instead of $O(L^2)$ 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.

### 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 $O(L \log L)$ 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.

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 \( u = (u_0, \ldots, u_{L-1}) \in \mathbb{R}^L \), deep SSMs (Gu et al., 2022b) are based on the recurrent equation:

\[
\begin{align*}
    x_{j+1} &= Ax_j + bu_{j+1}, \\
    y_{j+1} &= c^\top x_{j+1} + du_{j+1},
\end{align*}
\]

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

State-space convolution. By unrolling the recurrence above, the output sequence \( y \in \mathbb{R}^L \) can be expressed as:

\[
\begin{align*}
    y_j &= \sum_{l=0}^{j} c^\top A^{j-l} b u_l + du_j, \\
    \forall l \in \{1, \ldots, L\}. 
\end{align*}
\]

Let \( \ast \) denote the causal convolution operator (details about this operator are in Appendix A). Then, we can define a convolution kernel \( \kappa \in \mathbb{R}^L \) that depends on \( A, b, c \). A SSM layer is therefore parametrized by \( A, b, c, d \) through \( \kappa \) and its output is defined by \( y \) as in the following equation:

\[
\begin{align*}
    y &= \kappa \ast u + du, \\
    \kappa &= \left( c^\top b, c^\top Ab, \ldots, c^\top A^{L-1} b \right). 
\end{align*}
\]

For multidimensional \( u \in \mathbb{R}^{L \times H} \), we simply compute \( H \) convolutions with one kernel \( \kappa_h \) 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 \( y \) 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 \( O(L \log L) \) (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

<table>
<thead>
<tr>
<th>Encoder architecture</th>
<th>Complexity per layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer (full)</td>
<td>( O(L^2) )</td>
</tr>
<tr>
<td>LED</td>
<td>( O(L w) )</td>
</tr>
<tr>
<td>BigBird</td>
<td>( O(L w + L(g + r)) )</td>
</tr>
<tr>
<td>LSG</td>
<td>( O(L w + L(g + s)) )</td>
</tr>
<tr>
<td>LongT5 (TGlobal)</td>
<td>( O(L w + L \lceil L/c \rceil) )</td>
</tr>
<tr>
<td>LOCOST</td>
<td>( O(L \log(L)) )</td>
</tr>
</tbody>
</table>

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
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 $x$. The convolution view provides another angle: each output $y_j$ is a weighted sum of the previous tokens $u_0, \ldots, u_j$, whose weights are given by $\kappa$.

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

$$y_j = \sum_{l \leq j} \tilde{\kappa}_{j-l} \otimes u_l + \sum_{l \geq j} \hat{\kappa}_{l-j} \otimes u_l + d \otimes u_j$$

$$= \text{BiSSM}(U)_j.$$  \hspace{1cm} (3)

In this equation, $U \in \mathbb{R}^{L \times H}$ is the embedding matrix of the input text: $(u_0, \ldots, u_{L-1})$. The kernels $\tilde{\kappa}, \hat{\kappa}$ are computed as in Equation (2), with their respective parameters $(\tilde{A}, \tilde{c}, \tilde{b})$ and $(\hat{A}, \hat{c}, \hat{b})$. The element-wise product is denoted by $\otimes$ and we consider multidimensional inputs, with one kernel per dimension.

The output $y_j$ is now contextualized as a weighted sum of previous $u_{\leq j}$ and subsequent $u_{\geq j}$ 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 $\rho(\tilde{A})$ and $\rho(\hat{A})$. Indeed the sensitivity of an output $y_j$ with respect to an input $u_l$ is bounded by the following quantity:

$$|\frac{\partial y_j}{\partial u_l}| \leq \begin{cases} \rho(\tilde{A})^{j-l} \frac{|\tilde{c}^\top \tilde{b}|}{|\tilde{b}|} & \text{if } l < j, \\ \rho(\hat{A})^{l-j} \frac{|\hat{c}^\top \hat{b}|}{|\hat{b}|} & \text{if } l > j. \end{cases}$$

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 $U \in \mathbb{R}^{L \times H}$ is first projected onto $Q, V \in \mathbb{C}^{L \times H}$.
- $V$ is contextualized through a BiSSM.
- A pointwise multiplication $Q \odot \text{BiSSM}(V)$ 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 $O(H^2L + HNL + HL \log L)$ time and $O(HNL)$ 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: $D = \{s_1, \ldots, s_M\}$. Given a ratio $\alpha$, 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:

$$U = \arg \top_{j} \bigcup_{i \neq j} \{s_i, s_j\}$$

The resulting subset $U \subseteq \{1, \ldots, M\}$ 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.


- **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 $5 \times 10^{-4}$ 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× longer sequences during training and 16× 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.
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.

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\textsubscript{train}) and inference (MEM\textsubscript{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
Table 4: Throughput comparison for different models at 4K and 16K input length.

<table>
<thead>
<tr>
<th>Model</th>
<th>Length 4K</th>
<th>Length 16K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inference (samples/s)</td>
<td>Training (samples/s)</td>
</tr>
<tr>
<td>LED_base</td>
<td>3.57</td>
<td>1.69</td>
</tr>
<tr>
<td>T5_base</td>
<td>2.27</td>
<td>1.49</td>
</tr>
<tr>
<td>LongT5_base</td>
<td>2.94</td>
<td>2.94</td>
</tr>
<tr>
<td>LOCOST</td>
<td>3.03</td>
<td>3.03</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Rel</th>
<th>Cons</th>
<th>Flu</th>
<th>Coh</th>
</tr>
</thead>
<tbody>
<tr>
<td>LongT5_base</td>
<td>4.6</td>
<td>4.7</td>
<td>3.7</td>
<td>3.7</td>
</tr>
<tr>
<td>LOCOST</td>
<td>4.3</td>
<td>4.4</td>
<td>3.6</td>
<td>3.5</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>#Params</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
<th>Gain (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BART_large</td>
<td>406M</td>
<td>38.7</td>
<td>7.6</td>
<td>13.6</td>
<td>20.0</td>
</tr>
<tr>
<td>T5_large</td>
<td>737M</td>
<td>39.9</td>
<td>8.0</td>
<td>14.0</td>
<td>20.6</td>
</tr>
<tr>
<td>PEGASUS_large</td>
<td>568M</td>
<td>36.0</td>
<td>7.2</td>
<td>12.9</td>
<td>18.7</td>
</tr>
<tr>
<td>LongT5_base</td>
<td>247M</td>
<td>33.9</td>
<td>7.2</td>
<td>15.6</td>
<td>18.9</td>
</tr>
<tr>
<td>LOCOST</td>
<td>234M</td>
<td>38.6</td>
<td>8.1</td>
<td>16.2</td>
<td>21.0</td>
</tr>
</tbody>
</table>

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

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.

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 Lucchioni 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$^\text{a}$.

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.

References


$^\text{b}$https://data.worldbank.org/indicator/EN.ATM.CO2E.PC


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.


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


A Convolution

A.1 Causal convolution

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

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

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 \( \widehat{\kappa} \) the periodized version of \( \kappa \) as: \( \forall j \in \mathbb{N}, \widehat{\kappa}[j] = \kappa[j \mod L] \). For index \( 0 \leq j \leq L - 1 \), the discrete circular convolution between \( u \) and \( \kappa \) is defined as:

\[
(\kappa \circledast u)[j] = \sum_{l=0}^{L-1} \widehat{\kappa}[j - l]u[l].
\]

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

\[
\kappa \circledast u = \mathcal{F}^{-1}(\mathcal{F}(\kappa) \mathcal{F}(u)),
\]

where \( \mathcal{F} \) designates the DFT of a sequence and \( \circledast \) designates the element-wise multiplication.

Causal convolution with DFT. To compute \( \kappa \ast u \) with a DFT, a trick is to pad \( \kappa \) and \( u \) with \( L \) zeros before taking their DFT. Indeed, if we replace \( \kappa \) and \( u \) 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding dimensions ( H )</td>
<td>768</td>
</tr>
<tr>
<td>Vocabulary size</td>
<td>32100</td>
</tr>
<tr>
<td>Feedforward dimension</td>
<td>2048</td>
</tr>
<tr>
<td>Activation function</td>
<td>GeLU</td>
</tr>
<tr>
<td>LayerNorm ( \varepsilon )</td>
<td>( 1 \times 10^{-6} )</td>
</tr>
<tr>
<td>State-space dimension ( N )</td>
<td>256</td>
</tr>
<tr>
<td>Number of encoder layers</td>
<td>12</td>
</tr>
<tr>
<td>Number of decoder layer</td>
<td>12</td>
</tr>
<tr>
<td>Decoder attention heads</td>
<td>12</td>
</tr>
<tr>
<td>AdamW ((\beta_1, \beta_2))</td>
<td>((0.9, 0.999))</td>
</tr>
<tr>
<td>AdamW weight decay</td>
<td>0</td>
</tr>
<tr>
<td>Pre-training LR schedule</td>
<td>( \sqrt{\text{max(10^4, current step)}} )</td>
</tr>
<tr>
<td>Pre-training dropout</td>
<td>0</td>
</tr>
<tr>
<td>Finetuning LR</td>
<td>5 \times 10^{-4}</td>
</tr>
<tr>
<td>Finetuning LR schedule</td>
<td>constant</td>
</tr>
<tr>
<td>Finetuning dropout</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 8: LOCOST hyperparameters.

D Computational complexity of a LOCOST layer

Projection onto \( Q \) and \( V \) takes \( \mathcal{O}(LH^2) \) time and \( \mathcal{O}(LH) \) space. Computing the SSM kernel \( \kappa = (c^\top b, c^\top Ab, \ldots, c^\top A^{L-1}b) \) 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* \( A \in \mathbb{C}^{H \times N \times N} \) is made of \( H \) diagonal matrices \( A_h = \text{diag}(\lambda_h) \in \mathbb{C}^{N \times N} \).
- For \( 0 \leq h \leq H \) and \( 0 \leq n \leq N \), \( \lambda \in \mathbb{R}^{H \times N} \) is \( \lambda_{h,n} = \exp\left(\Delta_h \lambda_{h,n}^{\text{Re}} + i \Delta_h \lambda_{h,n}^{\text{Im}}\right) \).
- \( \Delta \in \mathbb{R}^H \) 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 \( \kappa \).

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.
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.

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

F Dataset details

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


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.
### 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: