Efficient Long Sequence Modeling and Beyond

Anonymous ACL submission

Abstract

Transformers have been successful in almost all modalities and achieved remarkable performance. However, their quadratic complexity limits their application on long sequences. To address the issue, researchers proposed many efficient sequence models, such as efficient Transformers, and state space models. Intending to assist researchers in navigating this vast landscape, this paper presents a detailed survey of a wide range of recent long-sequence models from their shared nature of token mixing and provides a clear taxonomy. It offers a comprehensive overview of existing work, ultimately providing valuable insights for future design.

1 Introduction

Long-range dependency learning has been a central focus in sequence modeling (Hè and Kabic, 2023; Dong et al., 2022; Zhao et al., 2023). Traditional models like LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Chung et al., 2014) cannot be trained on sequences over a few thousand tokens due to both gradients vanishing/exploding and inability to parallelize over the time domain. In response, the Transformer (Vaswani et al., 2017) architecture was introduced to achieve faster training and better performances on various data modalities. Despite its breakthrough in performance, Transformer has quadratic complexity with sequence length, i.e., $\mathcal{O}(L^2)$ for a sequence of length L, limiting its training on short sequences. However, modeling on long sequences is essential for tasks like summarization, high-resolution image processing, biological sequence analysis, *etc.* Much research in recent years has focused on improving Transformer efficiency (Tay et al., 2022). Proposed techniques include approximating the costly full attention matrix through various means like sparsity (Ho et al., 2019; Beltagy et al., 2020; Zaheer et al., 2021) or conditional probability assumptions (Roy et al., 2020). Other innovations like matrix factorization (Tay et al., 2021), optimized memory access patterns, and architectural constraints have also been explored (Choromanski et al., 2020b). In parallel, models based on State Space Models (SSMs) (Gu et al., 2021a, 2020; Gupta et al., 2022) were introduced to handle long sequences and demonstrated promising performance.

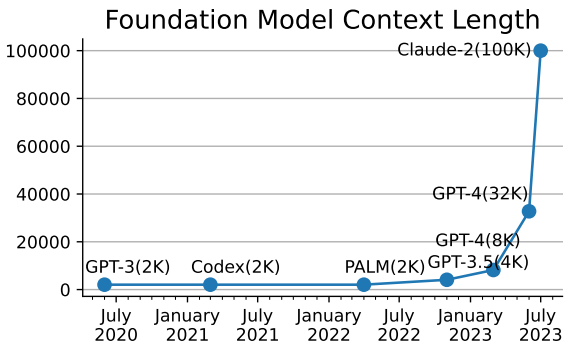


Figure 1: The context lengths of foundation models have been growing recently.

Starting with an introduction to the Transformer, we explain many of the recent efficient models and establish connections between them based on the shared nature of token mixing with a clear taxonomy for the plethora of these developed methods. A conceptual categorization of the methods discussed is illustrated in fig. 2. Additionally, we compare the ability of these models to long-sequence tasks and language modeling at different scales.

2 Background

The Transformer model serves as a strong baseline for the long-sequence models. We start by reviewing the architecture of the transformer and highlight the key elements in the sequence modeling.

Token Mixing and Channel Mixing The most important component in a Transformer Layer is its attention mechanism. Given an input sequence $X \in \mathbb{R}^{L\times d}$ where L is sequence length and d is

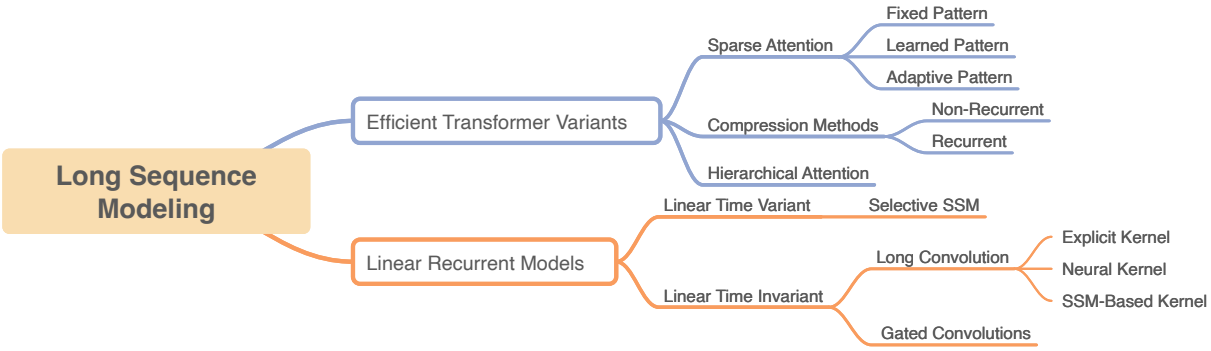


Figure 2: A conceptual categorization of the methods discussed in this paper.

the model dimension, the attention mechanism first obtains the query, key, and value matrices

$$Q = XW_q \quad K = XW_k \quad V = XW_v.$$

The dense interactions between tokens are defined:

$$\boldsymbol{A} = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d}}\right), \tag{1}$$

giving the token mixing matrix $\boldsymbol{A}$, which mix the input long the sequence length dimension

$$\mathrm{Attention}(Q, K, V) = \boldsymbol{A}V. \tag{2}$$

The mixing process, *i.e.,* $\boldsymbol{A}V$, is termed *token mixing*. Afterward, an MLP is applied to each token individually to mix the input along the channel dimension, called *channel mixing*. These two operations are essential in all sequence models and are the main degree-of-freedom in model design.

Data-dependent Token Mixing Note that all entries in the token mixing matrix $\boldsymbol{A}$ is input-dependent. This endows high expressivity to the transformer model while making it data-hungry. In addition, for long sequences $L \gg d$, attention eq. (1) has $\mathcal{O}(L^2)$ computation complexity, limiting its applicability to long sequences. Thus efficient models typically replace the token mixing component of the Transformer model. We find that the mixing matrix $\boldsymbol{A}$ is at the center of all token mixing processes, and efficient models exploit different structured $\boldsymbol{A}$. We thus center our discussion around the mixing matrix $\boldsymbol{A}$ and relate each method discussed below to it.

3 Transformer Variants

3.1 Sparse Attention

Methods in this category assume that the attention matrix is sparse, and reduce the computation by computing a subset of entries in $\boldsymbol{A}$. These methods can be further categorized into *fixed patterns*, *learnable pattern*, and *adaptive patterns*.

Fixed pattern methods have predetermined sparsity patterns. Simple examples include local attention, which applies full attention within each chunk in the input sequence, illustrated in fig. 3a. Sliding window attention allows attention to attend to w previous tokens for every token, illustrated in fig. 3b. These models have linear complexity but cannot learn dependencies outside of their receptive fields. To alleviate the issues, some models include a few global tokens that can attend to the whole sequence, thus achieving global receptive fields while maintaining subquadratic complexity. Examples include ETC (Ainslie et al., 2020) and BigBird (Zaheer et al., 2021), with illustration in figs. 3c and 3d. Other methods such as Sparse Transformer (Child et al., 2019) and LongNet (Ding et al., 2023) have sparsity design that allows models to attend to remote tokens sparsely, increasing the model's receptive field, shown in figs. 3e to 3g in Appendix A.

Learnable Pattern These models learn their sparsity patterns at train time and the pattern is fixed at test time. Sukhbaatar et al. (2019) learns a different attention span for each head. Nguyen et al. (2023) proposed to learn a score for each position in the attention matrix, i.e., prior, and prune away the lowest portion at test time.

Adaptive Pattern Methods with adaptive patterns adjust their sparsity pattern based on the input sequence. One direction is to group the most relevant tokens into clusters and apply full attention within clusters. Routing transformer (Roy et al., 2020) performs k-mean clustering on the fly to cluster the tokens. Similarly, Reformer (Kitaev et al., 2020) uses locality-sensitive hashing to assign similar queries and keys into the same buckets. Sinkhorn attention (Tay et al., 2020a) divides the Q, K, V sequences into chunks and uses a neural sorting algorithm to match the most relevant query and key chunks. Another direction is conditional

computation, i.e., performing attention on a subset of tokens. Informer (Zhou et al., 2021) select a subset of queries with a sparsity measurement that measures the 'focusness' of a query vector. COLT5 (Ainslie et al., 2023) divide the computation into 'light' and 'heavy' branches, and apply additional computation in the heavy branch to top-k tokens selected via a simple Multi-Layer Perceptron (MLP).

3.2 Compression methods

We refer to methods that compress arbitrary length input sequences into fixed-sized representations as compression methods. By having a fixed-sized sequence, most methods in the category have a linear complexity $\mathcal{O}(L)$. However, compressing a variable length input to a fixed-size representation is inherently lossy, resulting in decreased performance compared to full attention. We further categorize the methods discussed into recurrent and non-recurrent based on how they process the input. Regarding mixing matrix $\boldsymbol{A}$, methods first downsample then upsample the sequence can be seen as a form of low-rank approximation to the full attention matrix.

3.2.1 Non-recurrent

Percevier (Jaegle et al., 2021) uses a Transformer decoder to process arbitrary length input from any modality, i.e., vision, audio, or point cloud. The model maintains a fix-sized m latent representation and uses decoder blocks to iteratively attend to the input sequence with cross-attention and update the latent with self-attention, allowing the model to extract rich information from the input. Set transformer (Lee et al., 2019) proposed Induced Set Attention Block (ISAB) which uses m learnable queries $I \in \mathbb{R}^{m \times d}$ to reduces the sequence length to m, then projects back to the original length:

$$H = \text{Attention}(I, X, X)$$
$$\text{ISAB}_m(X) = \text{Attention}(X, H, H).$$

Linformer (Wang et al., 2020) compresses the K, V sequence with two additional projection matrices $E, F \in \mathbb{R}^{L \times k}$,

$$K' = E^\top K \quad Q' = F^\top Q$$

where $K', Q' \in \mathbb{R}^{k \times d}$, thus $\text{Attention}(Q, K', V')$ is reduced to linear complexity.

3.2.2 Recurrent

Recurrent methods can be classified as block recurrent or instance recurrent. Block recurrent methods contain recurrence at the chunk level. The most prominent block recurrent approach, Transformer-XL (Dai et al., 2019), applies the transformer models to chunked input sequentially while caching non-differentiable keys and values from the previous chunk. The cached keys and values are similar to the hidden states in RNNs, allowing Transformer models to scale to sequences of arbitrary lengths. Compressive transformer (Rae et al., 2019b) improves over Transformer-XL by adding a secondary memory that compresses and stores earlier memory that is discarded in Transformer-XL. Instance recurrent methods contain recurrence at the token level. Namely, this category contains linearized attention, discussed separately in the next section.

3.2.3 Linearized Attention

We start with a high-level overview of the linearized attention and its properties, then briefly discuss different instantiations, which mainly differ by how they approximate the softmax function to avoid quadratic computation.

Overview of Linearized Attention Linearized attention (Katharopoulos et al., 2020) approximates the attention logits or unnormalized scores:

$$S = \exp\left(\frac{QK^\top}{\sqrt{d}}\right)$$

with linear approximation

$$\hat{S} = \hat{Q}\hat{K}^\top,$$

where each method computes $\hat{Q}, \hat{K} \in \mathbb{R}^{L \times d}$ differently. The attention output is computed as:

$$\hat{\text{Attention}}(Q, K, V) = \hat{D}^{-1}\left(\hat{Q}\left(\hat{K}^\top V\right)\right) \quad (3)$$

where $\hat{\text{Attention}}$ is the approximated attention, $\hat{D}^{-1}$ normalise the score. By rearranging the computing order, the computation complexity is reduced to $\mathcal{O}\left(d^2 L\right)$. We categorize linearized attention as compression methods as $\left(\hat{K}^\top V\right)$ compress the sequence into a fixed-sized representation.

Note that the above attention mechanism is bidirectional, and causal masking cannot be easily applied, as $\hat{A}$ is never explicitly computed. Thus, for auto-regressive models, the attention can only be computed in a recurrent fashion similar to RNNs

$$\begin{aligned} H_t &= H_{t-1} + k_t v_t^\top \\ y_t &= \hat{D}_t^{-1} q_t H_t \end{aligned} \quad (4)$$

where $H_t = K_{:t}^\top V_{:t}$ is the memory of sequence up to length t, and y_t is the output at step t. Although the recurrent formulation gives constant space complexity, i.e., $\mathcal{O}\left(d^2\right)$, it prevents efficient parallel training. Note that this nicely connects the attention mechanism to the LRMs (see section 4).

Instantiations Katharopoulos et al. (2020) computes $\hat{Q}, \hat{K}$ by passing the Q, K through the following function to approximate softmax

$$\phi(x) = \mathrm{elu}(x) + 1.$$

Performer (Choromanski et al., 2020a) stochastically approximate S matrix with random mapping:

$$\hat{S}_{i,j} = \mathbb{E}\left[\phi^\top(q_i)\phi(k_j)\right],$$

which gives an unbiased estimation of the attention scores. Nyströmformer (Xiong et al., 2021) obtain m landmark queries and keys $\tilde{Q}, \tilde{K}$ by mean pool on Q, K and approximate $\hat{S}$ using Nyström method (Wang and Zhang, 2013). However, linearized attention still lacks in performance compared to full attention, mainly due to the attention dilution problem. The following research works on bridging the gap between vanilla attention and linearized attention. cosFormer (Qin et al., 2022b) reweighs the similarity measure with a cos-reweighting mechanism. TransNormer (Qin et al., 2022a) uses local attention at lower layers to avoid attention dilution problems. They also replace the scaling operation, i.e., $\hat{D}^{-1}$ in eq. (3), with normalization layers, such as LayerNorm (Ba et al., 2016), or RMSNorm (Zhang and Sennrich, 2019), to avoid unbounded gradient. Based (Arora et al., 2023) uses Taylor Expansion to compute the score. More advanced models include a decay mechanism on the hidden states H to forget distant information. RetNet (Sun et al., 2023) learns a diagonal $A \in \mathbb{C}^{d\times d}$

$$H_t = AH_{t-1} + k_t v_t^\top. \tag{5}$$

GateLoop (Katsch, 2024) instead learns a input dependent A_t by following classic RNNs:

$$A_t = \sigma(\mathrm{Linear}_\gamma(x_t)) \cdot \exp\left(i\,\mathrm{Linear}_\theta(x_t)\right),$$

where σ denotes the sigmoid function used to constrain the magnitude of the transition matrix.

3.3 Hierarchical Attention

Hierarchical methods reduce the complexity by hierarchically downsampling the sequence length dimension, similar to the U-Net (Ronneberger et al., 2015). Combiner (Ren et al., 2021) formulates the attention mechanism as conditional expectation, and uses conditional independence assumption to summarize remote chunks into tokens. Importantly, this formulation allows hierarchical attention to be computed without extra parameters. On the other hand, Subramanian et al. (2020); Shi et al. (2023) explores having multiple transformer or convolution layers at different resolutions to process the long-sequence data. They propose three variants, bottom-up, top-down, and retina models, suitable for different data modalities.

4 Linear Recurrent Models

With the popularity of the state space models (SSMs), the general form, Linear Recurrent Models (LRMs), is the new class of efficient models with more efficient performance compared to Transformer variants. We categorize Linear Time Invariant (LTI) Linear Time Variant (LTV) according to the data dependency. We start by briefly reviewing the history of the SSMs and establishing a connection between convolution and linear recurrence. Subsequently, various improvements over the LRMs are introduced by using the gating.

4.1 State Space Models (SSMs)

4.1.1 The HiPPO framework

The history of the family of SSMs dates back to the High-order Polynomial Projection Operators (HiPPO) framework (Gu et al., 2020). For some continuous signal $x(t)$, HiPPO formalizes the memorization problem as approximating the observed signal until time t $x_{\leq t}$ with N basis functions, and the coefficient of the basis functions gives the hidden states $h(t) \in \mathbb{R}^N$ at time t. The approximation is evaluated with some probabilistic measure $u(t)$ specifying the importance of each time. For three special measures, termed HiPPO-LegT, HiPPO-LagT, and HiPPO-LegS, analytical solutions of $h(t)$ were derived in the form of Linear Time-Invariant (LTI) ODEs [1]

$$\dot{h}(t) = \bar{A}h(t) + \bar{B}x(t).$$

[1]Note that we adopt the notation where $x(t), x_t$ denote the input in continuous form and discrete form respectively, $h(t), h_t$ denote the hidden states, and $y(t), y_t$ represent the output. This notation differs from the original formulation but is more aligned with the machine learning literature. We use a bar above the matrix to denote the parameterization in continuous form, i.e., $\bar{A}, \bar{B}$. In addition, A refers to the state transition matrix in the SSMs, not to be confused with the mixing matrix $\boldsymbol{A}$.

The derived $\bar{A}, \bar{B}$, which they later called the HiPPO matrices, are used extensively as the initialization scheme for subsequent SSMs, and were shown to be crucial for model training.

4.1.2 Linear State-Space Layer

To incorporate the HiPPO framework into deep learning models, Gu et al. (2021b) introduced the Linear State-Space Layer (LSSL). The center of the LSSL layer is a linear continuous-time SSM which maps a 1-dimensional function $x(t)$ to $y(t)$ through the hidden state $h(t) \in \mathbb{R}^{N\times 1}$

$$\begin{aligned} \dot{h}(t) &= \bar{A}h(t) + \bar{B}x(t) \\ y(t) &= \bar{C}h(t) + \bar{D}x(t), \end{aligned} \tag{6}$$

where $\bar{A} \in \mathbb{R}^{N\times N}, \bar{B}, \bar{C} \in \mathbb{R}^{N\times 1}, \bar{D} \in \mathbb{R}^{1\times 1}$. The SSM can also be viewed as a convolution over the input signal [2]

$$\begin{aligned} \bar{K}(t) &= \bar{C}e^{t\bar{A}}\bar{B} \\ y(t) &= (\bar{K} * x)(t), \end{aligned} \tag{7}$$

where the kernel $\bar{K}$ is a linear combination of the basis kernels $\bar{K}_i(t)$

$$\bar{K}(t) = \sum_{i=0}^{N-1} \bar{C}_i \bar{K}_i(t) \text{ where } \bar{K}_i(t) = (e^{t\bar{A}}\bar{B})_i.$$

Discrete-time LSSL The continuous-time LSSL in eq. (6) needs to be discretized for real-world data. Using the bilinear method and a step size of Δ

$$\begin{aligned} h_k &= Ah_{k-1} + Bx_k & A &= (I - \Delta/2 \cdot \bar{A})^{-1} \\ y_k &= Ch_k & & (I + \Delta/2 \cdot \bar{A}) \\ B &= (I - \Delta/2 \cdot \bar{A})^{-1}\Delta\bar{B} & C &= \bar{C}, \end{aligned} \tag{8}$$

giving the recurrent formulation of the LSSL.

LSSL in its recurrent form is inefficient to train in parallel. Unroll the recurrence gives

$$\begin{aligned} y_0 &= CBx_0 \\ y_1 &= CABx_0 + CBx_1 \\ y_2 &= CA^2Bx_0 + CABx_1 + CBx_2 \\ &\ldots\ldots \\ y_k &= CA^kBx_0 + CA^{k-1}Bx_1 + \ldots \\ &\quad + CABx_{k-1} + CBx_k, \end{aligned} \tag{9}$$

and we obtain the convolution kernel K of the layer

$$K = (CB, CAB, ..., CA^{L-1}B) \tag{10}$$

[2] We assume $h(0) = 0$ and $\bar{D} = 0$ for simplicity.

for a sequence of length L.

The two views of the LSSL, i.e., recurrence and convolution, allow LSSL to train in parallel while inference with a constant memory and time cost. An SSM is thus fully parameterized by $(\bar{A}, \bar{B}, \bar{C}, \bar{D}, \Delta)$. Note that the SSM is parameterized in continuous form, giving the additional benefits of stabilized training and the ability to dynamically adjust to signals sampled at different rates at test by changing the Δ. An SSM only takes a one-dimensional input, i.e., Single Input Single Output(SISO). For $X \in \mathbb{R}^{N\times d}$, an LSSL layer simply stacks d SSMs for each dimension.

Computation and Memory Efficiency The convolution can be computed efficiently with Fast Fourier Transforms (FFTs) when the kernel K is known. However, for sequence length L, computing the kernel K takes $\mathcal{O}(LN^2)$ operations for unstructured $\bar{A}$. The paper proposed an efficient albeit complex algorithm for a class of structured $\bar{A}$ with $\tilde{\mathcal{O}}(N + L)$, but is not implemented.

Expressivity II and Lawrence (2007) showed that a state-space model with hidden state size N can represent any rational convolution filter with degree N. Gu et al. (2021b) also showed that an infinitely deep stacked LSSL can approximate any non-linear ODEs. However, as the transition matrix A is limited to a subset of all possible instantiation, the expressivity of the LSSL model is unclear.

Relation to RNNs The LSSL in its recurrent form can be seen as an RNN without non-linearity between time steps. A significant challenge when training RNNs is the vanishing/exploding gradient problem (Pascanu et al., 2013). This is addressed in LSSL with the HiPPO initialization, which gives an inductive bias on learning long-range dependencies. This is corroborated by (Orvieto et al., 2023), which reproduces the performance of SSMs with a linear RNN with initialization that resulted in A with eigenvalues close to 1, demonstrating the importance of HiPPO initialization.

4.1.3 Simplifying the LSSL

S4 Although the algorithm in (Gu et al., 2021b) is theoretically efficient, it is difficult to implement in practice and suffers from numerical instability. Thus, the following research focuses on simplifying the LSSL while retaining its performance. The Structured State Space (S4) (Gu et al., 2021a) parameterize the transition matrices $\bar{A}$ as the sum

of a Normal matrix $A^{(N)}$ Plus a Low-Rank $PP^\top$ matrix (NPLR), which resulted in a different class of structured A and an efficient algorithm with $\tilde{\mathcal{O}}(N+L)$ complexity.

DSS Diagonal State Space (DSS) model (Gupta et al., 2022) further simplifies S4 by parameterizing the transition matrix A as a diagonal matrix and initialize A with diagonalized $A^{(N)}$, denoted $A^{(D)}$. The authors empirically observed that such parameterization gives equivalent performance compared to S4 models. Later, Gu et al. (2022) demonstrates the expressivity of the DSS model with the classical result that almost all SSMs are equivalent to a complex diagonal SSM. It then showed the basis kernels of an SSM model parameterized with $A^{(N)}$ approaches that of an SSM with HiPPO-LegS matrices when the state size $N \to \infty$, explaining the effectiveness of the initialization scheme.

S5 S5 model (Smith et al., 2023) extends LSSL to multi-input multi-output (MIMO) settings with $\bar{A} \in \mathbb{C}^{N\times N}, \bar{B}, \bar{C} \in \mathbb{C}^{N\times d}, \bar{D} \in \mathbb{R}^{d\times 1}$, and $\Delta \in \mathbb{R}^{N\times 1}$. They establish connections between the s4 and s5 models under some constraints and, therefore, utilize the same initialization as the DSS model. Instead of using convolution to speed up the computation, they choose the parallel scan algorithm, which computes the output fully in recurrence mode with the same computation complexity as the S4 model. Note that the parallel scan allows a different set of parameters A, B, C, Δ, to be used at each step, opening the possibility for Linear Time Varying (LTV) models, but is not further explored in the original paper.

4.2 Long Convolution

Convolution is also explored as a long-sequence modeling option (Fu et al., 2023b) where the kernel of an SSM is parameterized directly (explicit kernel). Although it matches the performance of S4 models, the parameter count of the model grows linearly with the sequence length, limiting its applicability to longer sequences. Romero et al. (2022); Poli et al. (2023) parameterize the convolution kernel with an MLP that takes a position as input and outputs the kernel weight (nerual kernel). The implicit parameterization decouples the input length with the parameter, allowing convolutional models to handle longer sequences.

Regarding the mixing matrix $\boldsymbol{A}$, convolution and SSM models learn fixed $\boldsymbol{A}$ that is input-invariant, illustrated in fig. 4. This view demonstrates the inherent deficiency of LTI model.

4.2.1 Convolution as Linear Recurrence

It is clear from eqs. (7) and (10) that SSMs are convolutions. In fact, we can view SSMs as a parameter-efficient way to parameterize a continuous convolution kernel. In another direction, it is also possible to write a convolution in the form of linear recurrence. Given a 1d convolution kernel $K \in \mathbb{R}^{L\times 1}$, we can construct an equivalent discrete SSM with state size L where A is a shift matrix

$$A_{ij} = \begin{cases} 1 & \text{for i-1=j} \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

$B = [1\ 0\ 0 \dots 0]^\top$, $C = K$ and $D = 0$.

4.3 Data-dependent Transformation

Despite the impressive results of LRMs, they lack behind transformers on language modeling. Research (Dao et al., 2022) attributes this discrepancy to the lack of a selective mechanism in these models, i.e., the mixing matrix $\boldsymbol{A}$ is input-independent. This limits the model's ability to filter out irrelevant tokens and focus on the important ones. To address the issue, various forms of *gating* were added for data dependency. The origin of gating can be traced back to LSTM (Hochreiter and Schmidhuber, 1997), which introduces an input gate, output gate, and forget gate to control how information flows in and out of the hidden states, and how the memory is maintained within the hidden states of an RNN. Later, gating was found to be empirically effective in both convolution networks (Hu et al., 2019) and transformers (Shazeer, 2020). Viewing the methods discussed above as linear RNNs, the following methods can be seen as adding input, output, or forget gate to the linear RNN model.

4.3.1 Input and Output Gating

Relation with Linearized Attnetion Recall the linearized attention in its recurrent form eq. (4), we can consider the $k_t v_t^\top$ operation as input gating, controlling how information enters the hidden states, and $q_t H_t$ as output gating, controlling what information in the hidden states is used for output. A minor difference is that linearized attention has a 2d hidden state, $H \in \mathbb{R}^{d\times d}$.

Instantiations The Hungry Hungry Hippo (H3) (Fu et al., 2023a) model takes inspiration from the linearized attention and introduces input-output

gates to the SSM models. The H3 model similarly projects Q, K, V from the input, and performs as:

$$Y = Q \odot \text{SSM}_{\text{diag}}(\text{SSM}_{\text{shift}}(K) \odot V)$$

where SSM_{diag} is a DSS (or S4D) model, and $\text{SSM}_{\text{shift}}$ is an SSM with fixed shift matrix A as in eq. (11). For $B = [1\,0\,0 \ldots 0]^\top$, $\text{SSM}_{\text{shift}}$ degrades to a short convolution with kernel size N, as in section 4.2.1. SSM_{diag} then mix the sequence with an output gate Q. Hyena (Poli et al., 2023) has a similar architecture with H3, whilst using an MLP to parameterize the long convolution kernel. RWKV (Peng et al., 2023) additionally pass Q through a sigmoid function, and K through an exponential function with a decay factor

$$\begin{aligned} H_t &= e^{-w} H_{t-1} + e^{k_{t-1}} v_{t-1}^\top \\ y_t &= \hat{D}_t^{-1} \sigma(q_t) \left(H_t + e^{u+k_t} \right). \end{aligned}$$

This is similar to the decay in RetNet eq. (5).

Limitations Despite the improved performance, Arora et al. (2023) showed that gated convolution models (LTI) failed to learn the Multi-Query Associative Recall (MQAR) task, highlighting the intrinsic limitation of LTI models.

4.3.2 Selective SSMs

The selectivity of the forget gate shows the most important gate in the LSTM model (van der Westhuizen and Lasenby, 2018; Greff et al., 2017; Jozefowicz et al., 2015) [3]. Therefore, many researchers attempt to equip the SSM models with a forget gate, which is achieved by making the transition matrix A input-dependent. An input-dependent A essentially allows past information to be discarded on demand. Note that this modification makes the dynamics of the SSM vary across time steps, making it an LTV system with increased expressivity and better performance on LM tasks.

Liquid S4 Liquid S4 model (Hasani et al., 2022) has the following form

$$\begin{aligned} \dot{h}(t) &= (\bar{A} + \bar{B} x_t) h(t) + \bar{B} x(t) \\ y(t) &= \bar{C} h(t) + \bar{D} x(t), \end{aligned}$$

which can be discretized and given in its convolution form similar to eq. (9)

$$\begin{aligned} y_0 &= CBx_0 \\ y_1 &= CABx_0 + CBx_1 + CB^2x_0x_1 \\ y_2 &= CA^2Bx_0 + CABx_1 + CBx_2 + CAB^2x_0x_1 \\ &\quad + CAB^2x_0x_2 + CB^2x_1x_2 + CB^3x_0x_1x_2 \\ &\ldots\ldots \end{aligned} \tag{12}$$

Note that the output consists of terms in a normal SSM in eq. (9) in black, and terms that take into account all orders of auto-correlation between input in violet. The kernel in eq. (12) can be separated into K, K_{liquid} to better illustrate the difference from a LSSL

$$y = K * x + K_{\text{liquid}} * x_{\text{auto-corr}}$$

where

$$\begin{aligned} K_{\text{liquid}} &= \left(CA^{L-1}B^2, \ldots, CB^2, CA^{L-2}B^3, \right. \\ &\quad \left. \ldots, CB^3, \ldots, CA^{L-P+1}B^P\right) \\ x_{\text{auto-corr}} &= [x_0x_1, \ldots, x_{L-1}x_L, \\ &\quad x_0x_1x_2, \ldots, x_{L-2}x_{L-1}x_L, \\ &\quad x_0x_1x_2 \ldots x_{P-1}, \ldots, x_{L-P+1} \ldots x_L]^\top \end{aligned}$$

In practice, the order of correlation is limited to some constants to avoid high computation costs. However, it is possible to compute the Liquid S4 without approximation utilizing the parallel scan.

Mamba Mamba (Gu and Dao, 2023) is a successful selective SSM approach that introduces the forget gate from a different angle. The state matrix A is made input-dependent by varying the step size $\Delta_t = (\text{softplus}((\text{Linear}_1)(x_t))$. A thus naturally becomes input-dependent during discretization in eq. (8). When $\Delta \to 0$, the hidden states reject new information, the current input is ignored. For larger Δ, the model accepts a larger rate of change. In this sense, Δ is like a tied input and forget gate. Note that B, C is also made data-dependent, and the resulting SSM is computed with parallel scans.

5 Comparison and Discussion

In Tab.1, 2, and 3, representative models from each category are compared on the LRA (Tay et al., 2020b), WikiText103 (Merity et al., 2016), PG-19 (Rae et al., 2019a), and Pile (Gao et al., 2020) at different scales. Experimental settings are described in Appendix B. Based on the results we reproduced and collected, there are three interesting observations:

[3] Interestingly, (Jozefowicz et al., 2015) found that the forget gate is not so important on language modeling task, contradicting the findings in mamba (Gu and Dao, 2023).

Category	Representative Methods	Time Complexity	LRA Datasets					
			ListOps	Text	Retrieval	Image	Pathfinder	Path-X
-	Vanilla Transformer	$O(L^2)$	36.37	64.27	57.46	42.44	71.40	✗
-	Pretrained Transformer + Flash2	$O(L^2)$	59.15	88.81	90.38	76.00	88.49	88.05
Sparse Attention	Sparse Transformer	$O(L \log L)$	36.29	64.92	55.35	40.15	70.38	✗
	BigBird	$O(L)$	36.05	64.02	59.29	40.83	74.87	✗
	Reformer	$O(L \log L)$	37.27	56.10	53.40	38.07	68.50	✗
Compressed Attention	Performer	$O(L)$	18.01	65.40	53.82	42.77	77.05	✗
	Linformer	$O(L)$	35.70	53.94	52.27	38.56	76.34	✗
	Nyströmformer	$O(L)$	37.15	65.25	79.56	41.58	70.94	✗
Linear Recurrent (SSMs)	S4	$O(L \log L)$	58.35	76.02	87.09	87.26	86.26	88.10
	DSS	$O(L \log L)$	59.60	86.82	90.90	88.65	94.20	96.35
	S5	$O(L \log L)$	62.15	89.31	91.40	88.00	95.33	98.58

Table 1: Comparison of performance and complexity of various sequence models in terms of sequence length (L).

1. *Long-sequence tasks need pretraining:* Causal pre-training on the original data alone and then fine-tuning the model can dramatically improve Transformer's performance on LRA.

2. *Hardware-efficient manner pushes the limit:* Transformer equipped with FlashAttetnion-2 (Flash2) to take full advantage of hardware computational features to complete training in Path-X. It also makes Mamba twice as fast as Flash2 when trained at 512k length.

3. *Time varying is matter:* Mamba substantially improves LRM's performance in Language Modeling with dynamic memory. It demonstrates the importance of combining prior patterns and data dependency simultaneously.

Insight for future design With the observations above, recent research (Zuo et al., 2022; Ma et al., 2022) has shown the advances of hybrid models combining LRMs and Attention. However, there has not been a consensus on the best way to combine the two distinct architectures and move towards hardware-efficient LTV, presenting an area of potential investigation for future architecture.

6 Other Applications

In addition to language modeling, there are very many application scenarios for long-sequence modeling. In **genomics research**, where DNA can interact across 100k+ base pairs or even across chromosomes (Avsec et al., 2021), efficient long sequence models become crucial to model the long-range dependencies. **Weather forecasting**, which was limited to mid-term predictions (Bi et al., 2023; Lam et al., 2023), can also benefit from the longer context to enable accurate long-term prediction.

Models	Datasets	
	WikiText103	PG-19
Transformer-125M	18.67	17.06
Performer-125M	26.83	22.24
Reformer-125M	26.10	21.47
Linear Attention-125M	25.60	19.14
Hybrid H3-125M	18.54	16.22
Mamba-130M	16.32	14.39

Table 2: The test perplexity on WikiText103 (Merity et al., 2016) and PG-19 (Rae et al., 2019a) datasets.

Tokenizer	Models	Perplexity
GPT-2	GPT-Neo-125M	9.42
	Hybrid H3-125M	8.85
	GPT-Neo-1.3B	6.23
	Hybrid H3-1.3B	6.04
	GPT-Neo-2.7B	5.73
	Hybrid H3-2.7B	5.41
NeoX	Pythia-160M	29.64
	Mamba-130M	10.56
	Pythia-1.4B	7.51
	RWKV-1.5B	7.70
	Mamba-1.4B	6.80
	Pythia-2.8B	6.73
	RWKV-3B	7.00
	Mamba-2.8B	6.22

Table 3: The perplexity on the PILE (Gao et al., 2020).

7 Conclusion and limitations

In this survey, we reviewed existing Transformer-based and LRM-based methods for long-sequence modeling. Further, we performed a quantitative comparison in terms of efficiency and offered suggestions regarding future design and applications. However, it can be challenging to fairly evaluate the efficiency of all long sequence models due to multiple factors that need to be considered and balanced. Simply using accuracy and complexity is not sufficient to evaluate effectiveness. It is also crucial to establish what resources, such as money, power consumption, carbon emissions, and others, need to be constrained. It is extremely challenging to unify the deployment of all models and the tools they rely on and to achieve a truly fair comparison.

References

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. 2023. CoLT5: Faster Long-Range Transformers with Conditional Computation.

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding Long and Structured Inputs in Transformers.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. 2023. Zoology: Measuring and Improving Recall in Efficient Language Models.

Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R Ledsam, Agnieszka Grabska-Barwinska, Kyle R Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R Kelley. 2021. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature methods*, 18(10):1196–1203.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. 2023. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, David Belanger, Lucy Colwell, and Adrian Weller. 2020a. Masked Language Modeling for Proteins via Linearly Scalable Long-Context Transformers.

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

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context.

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

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. LongNet: Scaling Transformers to 1,000,000,000 Tokens.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. 2023a. Hungry Hungry Hippos: Towards Language Modeling with State Space Models.

Daniel Y. Fu, Elliot L. Epstein, Eric Nguyen, Armin W. Thomas, Michael Zhang, Tri Dao, Atri Rudra, and Christopher Ré. 2023b. Simple Hardware-Efficient Long Convolutions for Sequence Modeling.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2017. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

Albert Gu and Tri Dao. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33:1474–1487.

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

Albert Gu, Karan Goel, and Christopher Ré. 2021a. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. 2021b. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585.

Ankit Gupta, Albert Gu, and Jonathan Berant. 2022. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994.

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. 2022. Liquid structural state-space models. *arXiv preprint arXiv:2209.12951*.

Hongyu Hè and Marko Kabic. 2023. A unified view of long-sequence models towards modeling million-scale dependencies. *arXiv preprint arXiv:2302.06218*.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. 2019. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*.

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

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. 2019. Squeeze-and-Excitation Networks.

Robert L. Williams II and Douglas A. Lawrence. 2007. *Linear State-Space Control Systems*. John Wiley & Sons.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. 2021. Perceiver: General Perception with Iterative Attention.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2342–2350. PMLR.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention.

Tobias Katsch. 2024. GateLoop: Fully Data-Controlled Linear Recurrence for Sequence Modeling.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. 2023. Learning skillful medium-range global weather forecasting. *Science*, page eadi2336.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models.

Tan M. Nguyen, Tam Nguyen, Long Bui, Hai Do, Duy Khuong Nguyen, Dung D. Le, Hung Tran-The, Nhat Ho, Stan J. Osher, and Richard G. Baraniuk. 2023. A Probabilistic Framework for Pruning Transformers Via a Finite Admixture of Keys. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. 2023. Resurrecting Recurrent Neural Networks for Long Sequences.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training Recurrent Neural Networks.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Xiangru Tang, Bolun Wang, Johan S. Wind, Stansilaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. RWKV: Reinventing RNNs for the Transformer Era.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena Hierarchy: Towards Larger Convolutional Language Models.

Zhen Qin, XiaoDong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. 2022a. The Devil in Linear Transformer.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. 2022b. cosFormer: Rethinking Softmax in Attention.

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

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2019b. Compressive Transformers for Long-Range Sequence Modelling.

Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. 2021. Combiner: Full Attention Transformer with Sparse Computation Cost.

David W. Romero, Anna Kuzina, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. 2022. CKConv: Continuous Kernel Convolution For Sequential Data.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient Content-Based Sparse Attention with Routing Transformers.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Jiaxin Shi, Ke Alexander Wang, and Emily B. Fox. 2023. Sequence modeling with multiresolution convolutional memory.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. 2023. Simplified State Space Layers for Sequence Modeling.

Sandeep Subramanian, Ronan Collobert, Marc'Aurelio Ranzato, and Y.-Lan Boureau. 2020. Multi-scale Transformer Language Models.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive Attention Span in Transformers.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive Network: A Successor to Transformer for Large Language Models.

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

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020a. Sparse Sinkhorn Attention.

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

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey.

Jos van der Westhuizen and Joan Lasenby. 2018. The unreasonable effectiveness of the forget gate.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Shusen Wang and Zhihua Zhang. 2013. Improving CUR Matrix Decomposition and the Nystr\"{o}m Approximation via Adaptive Sampling.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nystr\"omformer: A Nystr\"om-Based Algorithm for Approximating Self-Attention.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2021. Big Bird: Transformers for Longer Sequences.

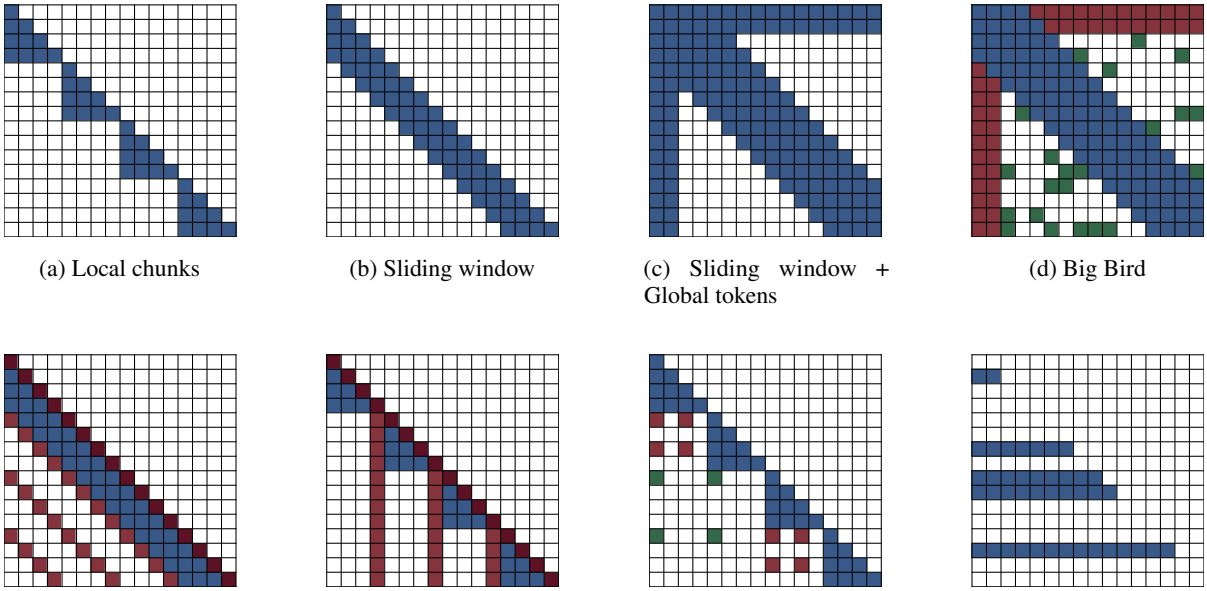
Biao Zhang and Rico Sennrich. 2019. Root Mean Square Layer Normalization.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting.

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.

A Mixing Patterns of Various Efficient Models



(a) Local chunks (b) Sliding window (c) Sliding window + Global tokens (d) Big Bird

(e) Sparse strided (f) Sparse fixed (g) Exponentially dialated attention (h) Q downsampling

Figure 3: Various sparse attention patterns illustrated. Note that different colors are used to distinguish sparsity patterns in one head.

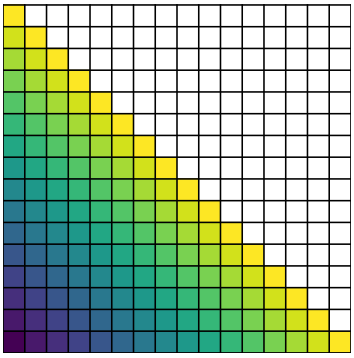


Figure 4: An illustrative mixing matrix $\boldsymbol{A}$ of a convolution model.

B Experimental Details

B.1 Long Range Arena

For all tasks, we closely follow Tay et al. (2020b) for details such as data preprocessing, data split, etc. The hyper-parameters on these tasks are listed in Table 4. The experimental configuration of sCIFAR follows the parameter settings of the image in LRA.

B.2 Language Modeling

The 125M models benchmarked in LM tasks have 12 layers with a model dimension of 1024. All models have an MLP expansion factor of 4, except Mamba. Mamba models have an expansion factor of 2, and double the number of layers. The 1.3B models have 24 layers, and a model dimension of 2048. The 2.7/2.8B models have 32 layers with hidden dimension 2560. The hybrid models have 12, 16, 16, and 20 heads for the 125M, 355M, 1.3B, and 2.7B models, respectively. The 125M hybrid model has attention layers at layers 1 and 7, the 1.3B hybrid models have attention layers at layers 1 and 13, and the 2.7B

Task	Depth	d_{model}	d_{FFN}	Norm	Pre-norm	BSZ	LR	Dropout	WD	Epochs
ListOps	6	80	160	BN	False	64	0.001	0.1	0.01	60
Text	4	128	256	SN	False	50	0.004	0.1	0.01	50
Retrieval	6	128	256	SN	False	64	0.003	0.1	0.04	40
Image	8	160	320	BN	True	50	0.01	0.0	0.02	200
Pathfinder	6	128	256	BN	True	128	0.01	0.0	0.01	200
Path-X	4	64	128	BN	True	128	0.01	0.0	0.01	100

Table 4: Hyper-parameters on LRA and raw speech classification tasks. BSZ is batch size, LR is learning rate and WD is weight decay. BN, LN and SN refer to Batch Normalization, Layer Normalization and Scale Normalization.

hybrid model has attention layers at layers 10 and 21. All H3 models have SSM state size 64, and Mamba models have state size 16. Our hybrid model uses head dimension 1 for H3, while our pure H3 model uses head dimension 8. Models were trained with bf16 precision. The training recipe generally follows that of GPT-3 (Brown et al., 2020).