LongVQ: Long Sequence Modeling with Vector Quantization on Structured Memory

Zicheng Liu[1,2] , **Li Wang**[2] , **Siyuan Li**[1,2] , **Zedong Wang**[2] , **Haitao Lin**[1,2] and **Stan Z. Li**[2†]

[1]Zhejiang University, Hangzhou, China

[2]AI Lab, Research Center for Industries of the Future, Westlake University, Hangzhou, China

{liuzicheng, wangli, lisiyuan, wangzedong, linhaitao, stan.zq.li}@westlake.edu.cn

Abstract

Transformer models have been successful in various sequence processing tasks, but the self-attention mechanism's computational cost limits its practicality for long sequences. Although there are existing attention variants that improve computational efficiency, they have a limited ability to abstract global information effectively based on their hand-crafted mixing strategies. On the other hand, state-space models (SSMs) are tailored for long sequences but cannot capture complicated local information. Therefore, the combination of them as a unified token mixer is a trend in recent long-sequence models. However, the linearized attention degrades performance significantly even when equipped with SSMs. To address the issue, we propose a new method called LongVQ. LongVQ uses the vector quantization (VQ) technique to compress the global abstraction as a length-fixed codebook, enabling the linear-time computation of the attention matrix. This technique effectively maintains dynamic global and local patterns, which helps to complement the lack of long-range dependency issues. Our experiments on the Long Range Arena benchmark, autoregressive language modeling, and image and speech classification demonstrate the effectiveness of LongVQ. Our model achieves significant improvements over other sequence models, including variants of Transformers, Convolutions, and recent State Space Models.

1 Introduction

Researchers and practitioners have been focusing on long-range dependency learning in sequence modeling. To tackle this challenge, popular sequence models like LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Chung *et al.*, 2014] have been introduced. These models can capture the relationships among input tokens (such as words [Ainslie *et al.*, 2020], pixels [Dosovitskiy *et al.*, 2021], and speech signals [Pascual *et al.*, 2019]) during the learning process. By utilizing contextual information, these models can learn intricate patterns of token dependencies in training samples. However, traditional sequential models are inefficient due to their non-parallelizability and limited memory. Consequently, scaling up the model parameters fails to enhance performance and significantly slows down training and inference speed. In response to the limitations of traditional sequence models, the Transformer [Vaswani *et al.*, 2017] architecture was introduced to achieve faster training and SotA performance. A key innovation enabling their parallelizability and superiority is the self-attention mechanism with positional encoding, which explicitly learns dependencies between all combinations of input tokens with order information encoded, aiming to create flexible kernels for sequences.

Transformers have revolutionized natural language processing (NLP) and achieved significant breakthroughs. However, they have a major drawback when dealing with very long input sequences that may contain millions of tokens. That is a quadratic $\mathcal{O}(L^2)$ complexity relative to the input sequence length in the self-attention mechanism, resulting in rapid scaling of compute and memory needs with sequence length L. Thus, many researchers in recent years focused on improving Transformer efficiency. Proposed techniques include approximating the costly self-attention matrix through various means like sparsity [Tay *et al.*, 2020a; Liu *et al.*, 2022] or conditional probability assumptions [Ren *et al.*, 2021]. Other innovations like matrix factorization, optimized memory access patterns, and architectural constraints have also been explored [Wang *et al.*, 2020; Beltagy *et al.*, 2020; Lingle, 2023]. While all of these operations reduce the complexity of the model, at the level of expressiveness, transformers can easily get lost in long sequence data. The *middle* of Fig. 1 demonstrates attention will fail in the long sequence scenarios, *e.g.,* resulting high-entropy attention matrix. In parallel, wholly novel models have been introduced to better handle long sequences. Models like applying linear recurrent units, which integrate state space models (SSMs) and polynomial projections, to reconstruct latent histories and demonstrate promising performance, especially on long-sequence tasks [Gu *et al.*, 2020b; Gu *et al.*, 2021; Gu *et al.*, 2022; Fu *et al.*, 2023a; Fu *et al.*, 2023b; Poli *et al.*, 2023]. As shown in Fig. 1(*left*), it has to be admitted, however, that the SSM model is far less capable of screening and selecting information than models with attention mechanisms [Fu *et al.*,
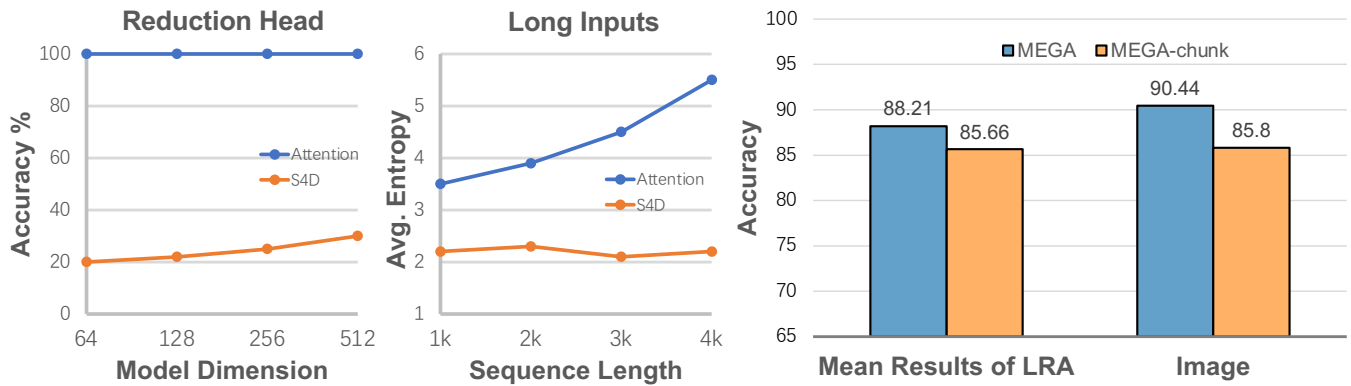
---

†Corrsponding author.

Figure 1: Illustration of the complementary property of Attenion and State Space Model. Taking Reduction Head [Fu *et al.*, 2023a] and Long Inputs as tasks on Text data: self-attention has a decisive ability to sift through information, but gets lost in long sequences, *i.e.*, high entropy in attention matrix; SSM vice versa (*left* and *middle*). Generally, linear attention causes large performance degradation for the SSM-attention hybrid model (*right*). The first two experiments are conducted with a two-layer model with 128 embedding dimensions.

2023a].

Therefore, in order to combine the respective advantages of the SSMs and Transformers, some hybrid models have been proposed recently and also achieved good results on long sequence tasks [Ma *et al.*, 2022; Zuo *et al.*, 2022]. To accommodate the computational complexity of the SSM model, linear attention is generally used as a collocation. Nevertheless, we can clearly see in the experimental results that the overall performance has a noticeable degradation after switching self-attention to linear attention (see *right* of Fig. 1). The question that needs to be addressed is: **is it possible to combine the advantages of both and match the performance of a hybrid model with self-attention to linear complexity?**

To answer this question, we present ***Long Sequence Modeling with Vector Quantization*** (LongVQ), an attention-SSM hybrid model in linear time with respect to the sequence length. Our method combines vector quantized keys, localized positional biases, and a dynamic fixed-size memory mechanism to store and retrieve complex data structures in long sequences efficiently. It has transformed how we approach data analysis and processing and unlocked new possibilities for various fields, including natural language processing, speech recognition, and image recognition. We benchmarked LongVQ on five popular datasets, including Long Range Arena (LRA) [Tay *et al.*, 2020b], which has diverse and challenging data modalities, the image dataset sCIFAR [Krizhevsky *et al.*, 2009], the natural language datasets WikiText-103 [Merity *et al.*, 2016] and enwik8, and the speech data SpeechCommand [Warden, 2018]. In these datasets our method demonstrates promising performance.

2 Related Work

2.1 Efficient Transformers

Kernel method and clustering Kernelizable attention, a technique introduced in studies by [Katharopoulos *et al.*, 2020], involves computing query and key features and then applying the same nonlinearity to both of them separately. It omits additional nonlinearities when computing attention

weights. Clustering attention, as described by [Vyas *et al.*, 2020], involves using VQ queries and can be kernelized.

Hierarchical Attention Combiner, proposed by [Ren *et al.*, 2021], approximates softmax using a graphical model and enables decoder-only self-attention. H-Transformer-1D [Zhu and Soricut, 2021] reduces the complexity of encoder-only self-attention. Transformer-LS [Zhu *et al.*, 2021] downsamples long-range features in Transformers.

Compressed Attention Compressive Transformers [Rae *et al.*, 2019; Ainslie *et al.*, 2020; Zhu *et al.*, 2021], learn a compression function for long-range features directly. On the other hand, LUNA [Ma *et al.*, 2021] uses cross-attention to compress long-range features into a recurrent state. Our LongVQ differs from compressive/recurrent Transformers in that it equates to quadratic-time attention over vector-quantized keys. This means that if the keys are already vector-quantized, the LongVQ cache reduces the cost to linear time without any loss. In contrast, Perceivers [Jaegle *et al.*, 2021] use cross-attention to attend to long sequences and compute self-attention over only a narrow stack of "latents." However, LongVQ computes dense self-attention in linear time instead of just cross-attention. Therefore, while Perceivers' long-range layers incur a quadratic time complexity during sampling, LongVQ generates sequences in linear.

2.2 State Space Models

Recurrent neural networks and their linear counterparts, such as state-space models, can retain memory of the past. Among them, S4 [Gu *et al.*, 2021] is notable because it can be implemented through convolutions thanks to its linear recurrence. However, the long convolution kernel for this model is as long as the input sequence, and its efficient computation requires sophisticated parameterization and approximation techniques. Although recent advances have found solutions to this issue, initializing these models still requires special effort [Gupta *et al.*, 2022; Gu *et al.*, 2020b]. Many of these models use the HiPPO [Gu *et al.*, 2020a] initialization mechanism, which aims to memorize historical data through projection to orthogonal polynomials. Based on a

structure similar to SSM, an increasing number of works focusing on either linear recurrence or global convolution have been developed recently [Fu *et al.*, 2023b; Fu *et al.*, 2023a; Poli *et al.*, 2023].

2.3 Vector Quantization

Various techniques such as k-means, vector quantization, and codebooks have been used in Transformers for various applications aiming to learn semantic discrete tokens [Roy *et al.*, 2020]. These techniques typically involve codebooks or similar structures within a Transformer architecture. Some models have proposed using a codebook outside the Transformer, such as when Transformers are priors for VQ-VAEs [van den Oord *et al.*, 2018]. LongVQ utilizes one codebook within each layer and performs dense self-attention with SSMs in linear time, which distinguishes it from all of the aforementioned works.

2.4 Data-dependent Gating

Gated attention is a fusion of attention sublayers and GLU-based MLP sublayers, which was introduced in FLASH [Hua *et al.*, 2022b]. Various gating mechanisms have been previously used to stabilize the training of Transformers and other sequence models. Examples of such models include GSS [Mehta *et al.*, 2022], MEGA by [Ma *et al.*, 2022], and RWKV by [Peng *et al.*, 2023].

3 Preliminary

3.1 Self-attention Mechanism

Given the input to the layer is $\boldsymbol{X} \in \mathbb{R}^{L \times d}$, where L is the sequence length and d is the embedding dimension, then the self-attention mechanism outputs pair-wise score $\boldsymbol{M}$:

$$\boldsymbol{M} = \mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \sigma\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}\right)\boldsymbol{V}, \tag{1}$$
$$\text{where } \boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_q, \ \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_k, \ \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_v.$$

Here $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{d \times d}$ are learnable matrices, and σ is the self-attention function *e.g.,* softmax. The self-attention mechanism facilitates the computation of pair-wise relationships of all input tokens as cache, thereby enabling the modeling of long-range dependencies that surpasses the capabilities of recurrent neural networks. For example, denote the attention score $\boldsymbol{M} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d}) \in \mathbb{R}^{L \times L}$. Then, $\boldsymbol{M}_{ij}$ computes the similarity between the i-th and the j-th tokens.

3.2 Linear Recurrent Models

Continuous linear state-space model. 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 $h(t) \in \mathbb{R}^{N \times 1}$. Concretely, LSSL can be formally defined as:

$$\hat{h}(t) = \boldsymbol{A}h(t) + \boldsymbol{B}x(t)$$
$$y(t) = \boldsymbol{C}h(t) + \boldsymbol{D}x(t).$$

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

Many researchers use Eq. 2 to model long sequences. For instance, [Gu *et al.*, 2020a] argue that randomly initialized parameters $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ aren't effective in modeling long-range dependencies. To address this issue, they propose a set of matrices called HiPPO (high-order polynomial projection operators) for initializing $\boldsymbol{A}$. The HiPPO matrices are designed in such a way that the state $x(t)$ at time t can remember the history of the input $u(t)$ up to time t.

Discrete-time form. The continuous-time LSSL in 2 needs to be discretized for real-world data. Using the bilinear method and a step size of Δ:

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

We unroll the above recurrent representation, then we have:

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

Convolution form. This can be written as a convolutional representation $y = \overline{\boldsymbol{K}} * u$, where the convolution kernel

$$\overline{\boldsymbol{K}} \in \mathbb{R}^L = \left(\overline{\boldsymbol{C}\boldsymbol{B}}, \overline{\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}}, \cdots, \overline{\boldsymbol{C}\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}}\right). \tag{3}$$

Here, the SSM transformation can be viewed as embedding mapping from input to output by using a long convolution kernel. "$*$" is the discrete convolution operator, u represents the input, and y represents the corresponding output. In Eq. 3, the output y can be efficiently computed if the convolution kernel $\overline{K}$ is known. However, computing the kernel can be challenging, with most existing algorithms having $O(L^2)$ time and space complexity.

Structured State Space Sequence Model [Gu *et al.*, 2021] develop the Structured State Space Sequence (S4) model to efficiently compute Eq. 3. Specifically, $\boldsymbol{D}$ is the residual connection, $\boldsymbol{C}$ in Eq. 2 is randomly initialized, and $\boldsymbol{A}$ and $\boldsymbol{B}$ are initialized as structured matrices as following:

$$\boldsymbol{A} = \boldsymbol{A}^{(d_s)} - \boldsymbol{P}\boldsymbol{P}^\top, \quad \boldsymbol{B}_i = (2i+1)^{\frac{1}{2}}, \tag{4}$$
$$\text{where } \boldsymbol{P}_i = (i+1/2)^{1/2},$$
$$\boldsymbol{A}_{ij}^{(d_s)} = - \begin{cases} (i+\frac{1}{2})^{\frac{1}{2}}(j+\frac{1}{2})^{\frac{1}{2}}, & i > j, \\ \frac{1}{2}, & i = j, \\ -(i+\frac{1}{2})^{\frac{1}{2}}(j+\frac{1}{2})^{\frac{1}{2}}, & i < j. \end{cases}$$

The convolution can be computed efficiently with Fast Fourier Transforms (FFTs) when the kernel $\overline{K}$ is known. However, for sequence length L, computing the kernel $\overline{K}$ takes $\mathcal{O}(LN^2)$ operations for unstructured $\overline{A}$. The paper proposed an efficient albeit complex algorithm for a class of structured $\overline{A}$ with $\tilde{\mathcal{O}}(N + L)$.

4 Sequence Modeling with Vector Quantization on Structured Memory

This section describes our proposed model, *long sequence modeling with vector quantization* (LongVQ). We first introduce the linear-time VQ attention equipped with the SSM model. Then, we present the whole block of LongVQ, including feed-forward and normalization layers. At last, we provide the network blocks and training algorithm of LongVQ.
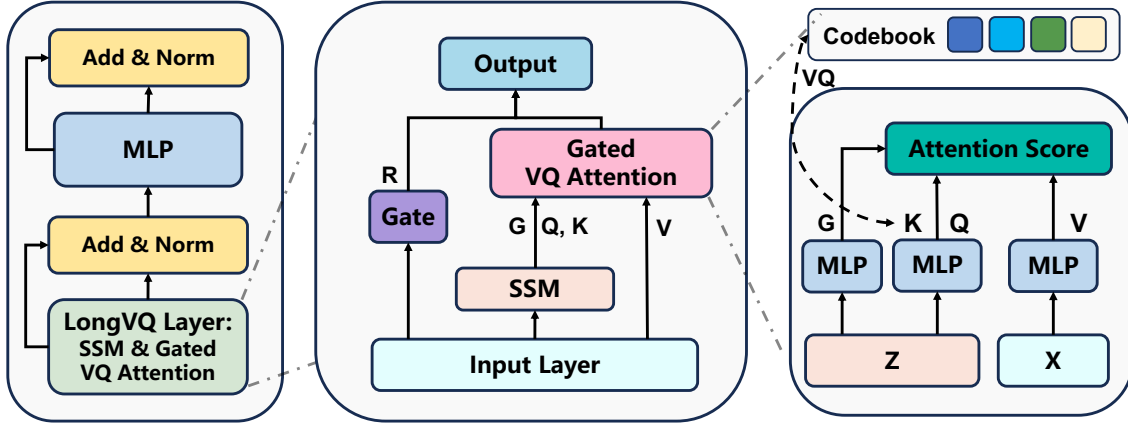
Figure 2: LongVQ – model architecture. Figure *left* shows the overall architecture of each LongVQ block. Figure *middle* illustrates the gated SSM-attention hybrid model with vector quantization method, while Figure *right* displays the details of our proposed VQ-equipped Attention.

4.1 Vector Quantization

Vector quantization (VQ) is a widely used information compression technique [Lingle, 2023]. In our work, VQ will be employed in self-attention layers to ensure linear complexity and make SSM operators more expressive.

What can VQ bring to SSM-attention hybrid? As discussed in Sec. 1, SSM and self-attention mechanisms each have their own limitations despite their wide applications and impressive successes in sequence modeling. The hybrid models enjoy both side benefits by leveraging their properties to complement each other. However, the linear attention, *e.g.,* chunk-wise attention, degrades the performance dramatically [Ma *et al.*, 2022]. The linear mixing methods designed artificially may lose the global view. VQ is a two-way solution that retains linear complexity and dynamically compresses global information to address this issue. The resulting model benefits from strong inductive bias while also being able to efficiently learn complex dependency patterns.

Vector Quantizers and Training A *vector quantizer* is a function $\text{VQ}(\cdot; \mathbf{C})$ compresses a continuous input vector x to a discrete embedding space $\hat{x} \in \mathbb{R}^{S \times D}$ with fixed size:

$$z = \arg\min_s ||\mathbf{x} - \mathbf{C}_s||^2, \tag{5}$$

$$\hat{\mathbf{x}} = \mathbf{C}_z, \tag{6}$$

where $\mathbf{C} \in \mathbb{R}^{S \times D}$ is known as the *codebook*. The row indices $\{0, \ldots, S-1\}$ of $\mathbf{C}$ are called *shortcodes*, and the rows themselves are called *codewords*. Because of the non-differentiable nature of the quantization operation, the straight-through estimator is typically used to update the network and *codebook* in an end-to-end manner:

$$z = \arg\min_s ||\mathbf{x} - \mathbf{C}_s||^2, \tag{7}$$

$$\hat{\mathbf{x}} = \mathbf{x} + \text{SG}(\mathbf{C}_z - \mathbf{x}), \tag{8}$$

where $\text{SG}(\cdot)$ denotes for stop-gradient operator. Since the computation of attention varies with the length of the sequence, linear complexity can be achieved as long as we can compress the $\mathbf{K}$ to a constant length.

4.2 LongVQ Layer

The gated attention method in LongVQ is inspired by MEGA [Ma *et al.*, 2022] and Gated Attention Unit [Hua *et al.*, 2022a] as the basic block in architectures. With an SSM layer embedded to capture long-term patterns, the input of our Gated VQ attention $\boldsymbol{G}, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ can be computed as follows:

$$\boldsymbol{Z} = \phi_{\text{silu}}(\text{SSM}(\boldsymbol{X})), \tag{9}$$

$$\boldsymbol{G}_a = \phi_{\text{silu}}(\text{Linear}(\boldsymbol{Z})), \tag{10}$$

$$\boldsymbol{Q} = \text{Linear}(\boldsymbol{Z}), \tag{11}$$

$$\boldsymbol{K} = \text{Linear}(\boldsymbol{Z}), \tag{12}$$

$$\boldsymbol{V} = \phi_{\text{silu}}(\text{Linear}(\boldsymbol{X})), \tag{13}$$

where $\boldsymbol{Z}$ can be regarded as the structured memory compresses the whole sequence. $\boldsymbol{G}_a$ is the gating variable that empowers the expressiveness of attention. Note that ϕ_{silu} is the self-gated activation function (SiLU) [Ramachandran *et al.*, 2017]. The output of Gated Attention is:

$$\boldsymbol{O}_a = \boldsymbol{G}_a \odot \text{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}), \tag{14}$$

where $\odot$ denotes the element-wise production.

Linear-time VQ Attention Note that the above computation of attention is still in quadratic complexity. Recall the vector quantization process in Sec.3. By following [van den Oord *et al.*, 2018], we could compress the $\boldsymbol{K}$ into a dynamic codebook with a fixed length:

$$\hat{\boldsymbol{K}} = \text{STVQ}(\boldsymbol{K}; \boldsymbol{C}), \tag{15}$$

where $\text{STVQ}(\cdot; \boldsymbol{C})$ represents the row-wise application of vector-quantization with a straight-through gradient estimator. Thus, the attention score can be refactored as:

$$\text{Attn}(\boldsymbol{Q}, \hat{\boldsymbol{K}}, \boldsymbol{V}) = \phi_{\text{element}}(\boldsymbol{Q}\hat{\boldsymbol{K}}^{\top})\boldsymbol{V}, \tag{16}$$

$$= \phi_{\text{element}}(\boldsymbol{Q}\hat{\boldsymbol{C}}^{\top})\Delta\boldsymbol{V}, \tag{17}$$

where Δ could be regarded as *Kronecker delta function* that builds a mapping between each token in $\boldsymbol{K}$ and code in $\boldsymbol{C}$. Here, we take the element-wise activation function as an example, *i.e.,* $\phi_{\text{element}} = \text{relu}^2(\cdot)$. The resulting attention computation achieves linear-time complexity for efficient sequence modeling.

Local window   If $C \ll L$, the code will appear uniformly throughout the sequence and will also cause the model to lose the ability to distinguish between near and far. To address this, a local bias B is added to the VQ attention for enhancing local information:

$$\text{Attn}(Q, \hat{K}, V, B) = \phi_{\text{element}}(Q\hat{K}^\top + B)V. \quad (18)$$

Moreover, we add an output gate G_o after the attention to control the final output O dependent on the input X:

$$G_o = \phi_{\text{sigmoid}}\text{Linear}(X), \quad (19)$$
$$O = G_o \odot O_a + (1 - G_o) \odot X. \quad (20)$$

Fig 2 *middle* plots the block design of LongVQ architecture.

4.3 LongVQ Blocks

The LongVQ layer can be used instead of regular self-attention in the Transformer model. Once added, it is followed by position-wise feed-forward networks (FFNs) and normalization layers to create a LongVQ block. As the gated residual connection is already included in equation 20, the original residual connection is omitted, and a normalization layer is directly applied to Y:

$$Y = \text{Norm}(\text{LongVQ}(X)), \quad (21)$$
$$Y' = \text{Norm}(\text{FFN}(Y)), \quad (22)$$

where Y' is the output of a single LongVQ block. The overall LongVQ block is shown in Fig 2 *left*.

Training loss   Let θ denote the set of non-codebook parameters, and C denote the set of layers' codebooks. For modeling sequences X for classification, our loss can be defined:

$$\mathcal{L}_{\text{LongVQ}} = \mathcal{L}_{\text{CE}}(X; \theta, C) + \gamma\mathcal{L}_{\text{VQ}}(X; \theta, C), \quad (23)$$

where γ is a hyperparameter known as the commit loss coefficient. The $\mathcal{L}_{\text{CE}}$ is the classification loss, and the $\mathcal{L}_{\text{VQ}}$ is employ mean squared error (MSE) between K and C:

$$\mathcal{L}_{\text{VQ}} = \text{MSE}(K, \text{SG}(C)). \quad (24)$$

The main cross-entropy loss is augmented with commitment losses to update the codebook across all layers. As [van den Oord *et al.*, 2018], codebooks are parameterized through smoothed quantizer statistics.

5 Experiments

To evaluate LongVQ, we conduct experiments on five benchmark sequence modeling tasks across various data types, comparing them with current state-of-the-art models on each task. In all cases, we use codebook size 512. All experiments were realized based on NVIDIA A100-80G and Pytorch. We used float32 parameters, with bfloat16 precision for most computations. We adopt AdamW as the optimizer with a gradient clip of 0.1. The codebook commit coefficient was always $\gamma = 0.0001$, and the codebook EMA rate was always $\eta = 0.99$. All models were trained with a global batch size of 128 sequences. In our table, bold indicates the best-performing model and underlines the second-best.

Model	Accuracy (%)
Attention:	
Transformer [Trinh *et al.*, 2018]	62.20
RNN:	
LSTM [Hochreiter and Schmidhuber, 1997]	63.01
r-LSTM [Trinh *et al.*, 2018]	72.20
UR-GRU [Gu *et al.*, 2020b]	74.40
HiPPO-RNN [Gu *et al.*, 2020a]	61.10
LipschitzRNN [Erichson *et al.*, 2020]	64.20
State Space Models:	
S4 [Gu *et al.*, 2022]	91.80
S4D [Gu *et al.*, 2022]	90.69
S5 [Smith *et al.*, 2023]	90.10
Liquid-S4 [Hasani *et al.*, 2022]	<u>92.02</u>
Convolution:	
TrellisNet [Bai *et al.*, 2018]	73.42
CKConv [Li *et al.*, 2022]	63.74
FlexConv [Romero *et al.*, 2021]	80.82
Hybrid:	
LongVQ (Ours)	**92.55**

Table 1: Performance of pixel-level sequential classification on the sCIFAR dataset. Results of compared methods are taken from either the citation or [Hasani *et al.*, 2022].

5.1 Pixel-Level Sequential Image Classification

Let's begin with tasks related to image classification, where images are treated as a sequence of pixels in one dimension. In such tasks, models are not allowed to use any two-dimensional bias from the image. Therefore, the model should be capable of identifying patterns at various timescales, including pixels that are close to the original image but far from each other in its sequence representation. We assess our model's performance on the Sequential CIFAR-10 dataset, which is widely used as a standard benchmark for modeling long-term dependencies in RNNs. The CIFAR-10 dataset's standard train and test split is used, and 10% of the training set is withheld as the validation set. To accomplish the classification task, we compute the mean of all tokens in the output sequences and feed the result into a fully connected layer to generate class logits. The test accuracy is reported based on the model with the highest validation accuracy.

Results   The Table 1 displays the results. Our model has achieved state-of-the-art performance and the best test accuracy on the sequence classification task, surpassing numerous strong competitors such as Transformers [Vaswani *et al.*, 2017], RNNs, state space models, and other convolutional models. Specifically, the LongVQ model has outperformed the previous convolution-based models by more than ten percentage points. It is noteworthy to mention that our model has achieved impressive results by surpassing the previously established performance benchmark, despite utilizing a relatively simple architecture. The model primarily uses a hybrid approach that compresses long historical information based on the output of SSM.

Models	ListOps	Text	Retrieval	Image	Pathfinder	PathX	Avg.	Speed	Mem.
Attention:									
Transformer [Vaswani *et al.*, 2017]	36.37	64.27	57.46	42.44	71.40	✗	54.39	1.0×	1.0×
Local Attention [Tay *et al.*, 2020b]	15.82	63.98	52.98	41.46	66.63	✗	46.06	5.3×	0.1×
Linear Trans. [Katharopoulos *et al.*, 2020]	16.13	65.90	53.09	42.34	75.30	✗	50.55	4.7×	0.1×
Linformer [Wang *et al.*, 2020]	35.70	53.94	52.27	38.56	76.34	✗	51.36	5.5×	0.1×
Sparse Transformer [Child *et al.*, 2020]	17.07	63.58	59.59	44.24	71.71	✗	51.24	4.2×	0.2×
Performer [Choromanski *et al.*, 2020]	18.01	65.40	53.82	42.77	77.05	✗	51.41	5.7×	0.1×
Sinkhorn Transformer [Tay *et al.*, 2020a]	33.67	61.20	53.83	41.23	67.45	✗	51.39	3.8×	0.1×
Longformer [Beltagy *et al.*, 2020]	35.63	64.02	59.29	40.83	74.87	✗	55.01	1.1×	0.3×
BigBird [Zaheer *et al.*, 2020]	36.05	64.02	59.29	40.83	74.87	✗	55.01	1.1×	0.2×
Luna-256 [Ma *et al.*, 2021]	37.25	65.78	79.56	47.86	78.55	✗	61.95	4.9×	0.1×
Reformer [Kitaev *et al.*, 2020]	37.27	56.10	53.40	38.07	68.50	✗	50.67	0.8×	0.3×
State Space Models:									
S4 [Gu *et al.*, 2022]	59.60	86.82	76.02	87.09	87.26	86.26	80.50	-	-
DSS [Gupta *et al.*, 2022]	57.60	59.60	86.82	<u>90.90</u>	88.65	94.20	86.09	4.8×	0.1×
S4D [Gu *et al.*, 2022]	<u>60.18</u>	87.34	91.09	87.83	93.78	92.80	85.50	-	-
Hybrid:									
Mega-chunk [Ma *et al.*, 2022]	58.76	**90.19**	<u>91.25</u>	85.80	94.41	93.81	85.66	5.5×	0.1×
SPADE [Zuo *et al.*, 2022]	59.70	87.55	90.13	89.11	<u>96.42</u>	<u>94.22</u>	<u>86.19</u>	5.2×	0.1×
LongVQ (Ours)	**61.02**	<u>89.72</u>	**91.45**	**91.35**	**96.70**	**95.85**	**87.68**	5.3×	0.1×

Table 2: Performance of predicting outcomes of list operations in the LRA [Tay *et al.*, 2020b] benchmark. **Bold** indicates the best-performing model and underlines the second best. Results are taken from either the citation. The training speed and peak memory consumption comparison on the Text task with the input length of 4K.

5.2 Long-Context Sequence Modeling

We have started conducting experiments by evaluating sequence models on the Long Range Arena (LRA) benchmark, which was recently introduced by [Tay *et al.*, 2020b]. The benchmark aims to evaluate the performance of sequence models in long-context scenarios. The benchmark consists of a total of six tasks, each of which is designed to test different aspects of long-sequence modeling. These tasks include ListOps, Text, Retrieval, Image, Pathfinder, and Path-X. To ensure a comprehensive evaluation of our model, the tasks involve input sequences of varying lengths, ranging from 1K to 16K tokens. Additionally, the tasks cover a diverse range of data types and modalities, meaning that models must be capable of handling text, images, and other forms of input in order to perform well on the benchmark.

Results In Table 2, LongVQ is compared to various baselines, such as Transformer and its efficient versions, as well as the top-performing S4 models. Specifically, LongVQ outperforms the second-ranked model by 1.84%, 0.79%, 0.20%, 0.45%, 0.28%, 3.63% on each of the six datasets. To ensure a just comparison, we equalize the number of layers and model dimensions on each task so that LongVQ has a similar number of parameters with S4. The results are an average of five runs with distinct random seeds, and the tuning information and model details are given in the Appendix. The LongVQ has shown remarkable performance over all six tasks, with an average performance of 88.36%, outperforming all the other baselines. Furthermore, we have evaluated the speed and memory efficiency of LongVQ using the byte-level classification task with an input length of 4K. Our VQ mechanism has proven to be highly efficient, being 5.3 times faster and consuming only 10% as much memory as the vanilla Transformer. It is worth noting that the LongVQ layer, with its

VQ-equipped hybrid design, is even more efficient than both Transformers and State Space Models by a large margin.

5.3 Auto-Regressive Language Modeling

By following [Ma *et al.*, 2022; Lingle, 2023], we evaluate LongVQ on two established language modeling benchmarks, *i.e.*, WikiText-103 [Merity *et al.*, 2016] and enwik8, which are next-token prediction tasks. WikiText-103 is a word-level language modeling dataset containing 103M training tokens from Wikipedia articles. Following previous work [Baevski and Auli, 2019], our approach involves the utilization of adaptive softmax and input embeddings, and we work with a vocabulary of 260K tokens. enwik8, on the other hand, is a challenging benchmark for character-level language modeling. It comprises 100M tokens of unprocessed Wikipedia articles and has a vocabulary size of around 200. We divide the test data into segments and sequentially handle each segment during testing.

Results In Table 3, we compare with previous top-performing models that are designed to take advantage of longer context, including Transformers [Baevski and Auli, 2019], Transformer-XL and S4 [Gu *et al.*, 2021]. The LongVQ model has shown impressive performance on both

| Model | WikiText-103 | | |
	#Param.	PPL	Speed
Transformer-adaptive	247M	18.66	5.6 k t/s
Transformer-XL	257M	18.30	-
S4[Gu *et al.*, 2020b]	249M	20.95	-
Mega-chunk[Ma *et al.*, 2022]	252M	<u>18.07</u>	48k t/s
LongVQ (Ours)	254M	**17.85**	50k t/s

Table 3: Performance and training speed on WikiText-103 dataset.

WikiText-103 and enwik8 datasets, surpassing the baseline models by a considerable margin. Additionally, our model is able to achieve a faster (almost $10\times$) inference speed compared to the Pure Transformer model. One of the key benefits of the hybrid design of the SSM layer and VQ technique is that it enables LongVQ to naturally handle length extrapolation at inference time, allowing it to process longer sequences than those seen during training.

Model	enwik8 #Param.	PPL
Transformer-XL	41M	1.06
Mega [Ma *et al.*, 2022]	39M	1.02
Transformer-VQ [Lingle, 2023]	190M	0.99
LongVQ (Ours)	47M	**0.97**

Table 4: Testing bits-per-byte on Enwik8 dataset.

5.4 Raw Speech Classification

We aim to assess LongVQ's ability to model speech signals over a long range by utilizing it to classify unprocessed speech signals that have a length of 16000 instead of relying on conventional preprocessing methods, such as converting them to MFCC features. As per [Gu *et al.*, 2021] approach, we conduct speech classification on the Speech Commands dataset's SC10 subset, which was introduced by [Warden, 2018]. As reported in [Ma *et al.*, 2022], the Mega-chunk uses a chunk size of 1000 to enable processing the data.

Results As shown in Table 5, our LongVQ model with 542K parameters can achieve an accuracy of 97.67, which is the state-of-the-art method in this table. This is mainly because the large number of continuous and low-frequency signals in speech are well-suited to be processed using VQ. Coupled with SSM's ability to extract global information, it is only natural that such results are achieved.

Model	SpeechCommand-Raw #Param.	Accuracy
Transformer	786K	✗
S4 [Gu *et al.*, 2021]	300K	97.50
Mega [Ma *et al.*, 2022]	-	✗
Mega-chunk [Ma *et al.*, 2022]	476K	96.03
LongVQ (Ours)	542K	**97.67**

Table 5: Accuracy on Speech Commands dataset.

5.5 Ablation Study

We focus our analysis on LongVQ's important components to answer two questions: **(1) Is it possible to set parameters for data types in the codebook? (2) What is the best attention function for LongVQ?**

Codebook size We take the subset of LRA to conduct the ablation study. As shown in Table 6, larger codebook sizes may allow more flexible attention patterns and improve the gradients' fidelity, and improve final performance. However,

we can set the size of the codebook according to the resource requirements based on the data characteristics. For example, from images to logical reasoning, images with lower information density can use a lower size and vice versa.

Settings	Codebook Size			Attention Function	
	256	512	1024	softmax	laplace
Image	91.32	91.35	91.58	91.35	90.15
Text	86.26	89.72	89.75	89.72	88.27
ListOps	58.16	61.02	61.10	61.02	58.14

Table 6: Ablation study on Codebook size and Attention functions.

Attention function We tested two attention functions, softmax and laplace, to evaluate performance. Based on LongVQ, our findings differ from Mega's [Ma *et al.*, 2022]. Due to its compressed codebook on K, we found that LongVQ prefers softmax as the attention function for different data types; the results are shown in Table 6. This is mainly due to the existence of VQ technology, which dynamically compresses the data into codes, and softmax, which has the ability to filter information with strong sparsity, is more suitable for the data after compression. Hence, our LongVQ does not need parameter search in the choice of attention function.

6 Conclusion and Limitations

We presented LongVQ for robust and efficient memorization of long-term patterns in sequential data sources. LongVQ is an SSM-attention hybrid architecture that computes dense self-attention in linear time with respect to sequence length. Its superior performance is enabled by considering the global view of the SSM module and a gating mechanism. The vector quantization method to compress keys is the crucial point to making LongVQ more efficient when handling long sequences. By using the local window design, LongVQ can still capture long-range patterns even with small codebook sizes. Our large-scale and diverse experiments demonstrate that LongVQ is an efficient and flexible long sequence model with excellent performance on image, text, logical reasoning, and speech data. However, LongVQ also has some obvious limitations in empirical implementations. LongVQ has a relatively large training difficulty, and due to the online training codebook, we need to handle the gradient size of different modules carefully. Moreover, the exponential moving average (EMA) technique is essential for stabilizing codebook updates, but it also limits the update speed simultaneously. Whether a better-stabilized training model can be used on codebooks is still a question worth considering.

Acknowledgements

References

[Ainslie *et al.*, 2020] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online, 2020. Association for Computational Linguistics.

[Baevski and Auli, 2019] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling, 2019.

[Bai *et al.*, 2018] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Trellis networks for sequence modeling. *arXiv preprint arXiv:1810.06682*, 2018.

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

[Child *et al.*, 2020] R Child, S Gray, A Radford, and I Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2020.

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

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

[Dosovitskiy *et al.*, 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[Erichson *et al.*, 2020] N Benjamin Erichson, Omri Azencot, Alejandro Queiruga, Liam Hodgkinson, and Michael W Mahoney. Lipschitz recurrent neural networks. *arXiv preprint arXiv:2006.12070*, 2020.

[Fu *et al.*, 2023a] Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models, 2023.

[Fu *et al.*, 2023b] Daniel Y. Fu, Elliot L. Epstein, Eric Nguyen, Armin W. Thomas, Michael Zhang, Tri Dao, Atri Rudra, and Christopher Ré. Simple hardware-efficient long convolutions for sequence modeling, 2023.

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

[Gu *et al.*, 2020b] Albert Gu, Caglar Gulcehre, Thomas Paine, Matt Hoffman, and Razvan Pascanu. Improving the gating mechanism of recurrent neural networks. In *International Conference on Machine Learning*, pages 3800–3809. PMLR, 2020.

[Gu *et al.*, 2021] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *ArXiv preprint*, abs/2111.00396, 2021.

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

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

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

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

[Hua *et al.*, 2022a] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *International Conference on Machine Learning*, pages 9099–9117. PMLR, 2022.

[Hua *et al.*, 2022b] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer quality in linear time, 2022.

[Jaegle *et al.*, 2021] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention, 2021.

[Katharopoulos *et al.*, 2020] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[Kitaev *et al.*, 2020] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 2009.

[Li *et al.*, 2022] Yuhong Li, Tianle Cai, Yi Zhang, Deming Chen, and Debadeepta Dey. What makes convolutional models great on long sequence modeling? *arXiv preprint arXiv:2210.09298*, 2022.

[Lingle, 2023] Lucas D. Lingle. Transformer-vq: Linear-time transformers via vector quantization, 2023.

[Liu *et al.*, 2022] Zicheng Liu, Siyuan Li, Di Wu, Zihan Liu, Zhiyuan Chen, Lirong Wu, and Stan Z. Li. Automix: Unveiling the power of mixup for stronger classifiers, 2022.

[Ma *et al.*, 2021] Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. *Advances*

in Neural Information Processing Systems, 34:2441–2453, 2021.

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

[Mehta *et al.*, 2022] Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces, 2022.

[Merity *et al.*, 2016] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[Pascual *et al.*, 2019] Santiago Pascual, Mirco Ravanelli, Joan Serrà, Antonio Bonafonte, and Yoshua Bengio. Learning problem-agnostic speech representations from multiple self-supervised tasks, 2019.

[Peng *et al.*, 2023] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Bolun Wang, Johan S. Wind, Stanislaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. Rwkv: Reinventing rnns for the transformer era, 2023.

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

[Rae *et al.*, 2019] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019.

[Ramachandran *et al.*, 2017] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

[Ren *et al.*, 2021] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost, 2021.

[Romero *et al.*, 2021] David W Romero, Robert-Jan Bruintjes, Jakub M Tomczak, Erik J Bekkers, Mark Hoogendoorn, and Jan C van Gemert. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. *arXiv preprint arXiv:2110.08059*, 2021.

[Roy *et al.*, 2020] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers, 2020.

[Smith *et al.*, 2023] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *International Conference on Learning Representations*, 2023.

[Tay *et al.*, 2020a] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*. PMLR, 2020.

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

[Trinh *et al.*, 2018] Trieu Trinh, Andrew Dai, Thang Luong, and Quoc Le. Learning longer-term dependencies in rnns with auxiliary losses. In *International Conference on Machine Learning*, pages 4965–4974. PMLR, 2018.

[van den Oord *et al.*, 2018] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018.

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

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

[Wang *et al.*, 2020] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[Warden, 2018] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition, 2018.

[Zaheer *et al.*, 2020] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

[Zhu and Soricut, 2021] Zhenhai Zhu and Radu Soricut. H-transformer-1d: Fast one-dimensional hierarchical attention for sequences. *arXiv preprint arXiv:2107.11906*, 2021.

[Zhu *et al.*, 2021] Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. *Advances in Neural Information Processing Systems*, 34, 2021.

[Zuo *et al.*, 2022] Simiao Zuo, Xiaodong Liu, Jian Jiao, Denis Charles, Eren Manavoglu, Tuo Zhao, and Jianfeng Gao. Efficient long sequence modeling via state space augmented transformer, 2022.