MixCon: A Hybrid Architecture for Efficient and Adaptive Sequence Modeling

Xin Xu[a] **and Zhouchen Lin**[a,b,c,*]

[a]State Key Lab of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University
[b]Institute for Artificial Intelligence, Peking University
[c]Pazhou Laboratory (Huangpu), Guangzhou, Guangdong, China

Abstract.

Sequence modeling is a critical task in various domains such as natural language processing, speech recognition, and time series analysis. The existing models still face challenges in capturing long-range dependencies and efficiently modeling sequences. This paper proposes a novel hybrid sequence modeling architecture called MixCon to address these challenges. The MixCon (Mixture of Conba) architecture combines a Transformer layer based on attention mechanism, a Conba layer, and a Mixture of Experts (MoE) module. We apply this idea to the design of the attention mechanism, achieving significant improvements in computational efficiency. Additionally, the MixCon architecture integrates feedback and adaptive control mechanism inspired by control theory, providing a new perspective and approach to sequence modeling. The experimental results demonstrate MixCon's superior throughput, outperforming Mixtral by 4.5 times and Jamba by 1.5 times when processing lengthy sequences of up to 128K tokens on a single A800 80GB GPU. Moreover, MixCon achieves top-tier scores on academic benchmarks, exemplified by its outstanding performance with a score of 87.9% on HellaSwag and 83.4% on WinoGrande, showcasing its capability to excel in complex sequence modeling tasks.

1 Introduction

Despite significant advancements in sequence modeling in fields such as natural language processing, speech recognition, and time series analysis, existing models still face challenges in capturing long-range dependencies and efficiently modeling sequences. The Transformer model [32], although dominant in natural language processing tasks, has high computational costs and significant memory requirements. Consequently, researchers have proposed alternative models such as linear attention transformer, Mixture of Experts (MOE) models, and linear RNN models.

The linear attention Transformer is an approach that aims to improve the efficiency of the original transformer model by approximating the attention mechanism with linear complexity. While it reduces computational overhead, it may struggle with capturing long-range dependencies and might experience a decrease in performance compared to the original Transformer. Early research utilized local sensitive hashing schemes to reduce the computational complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$ [19], but it introduced a large constant factor that made it challenging to apply in common cases. Recent studies have explored alternative methods to approximate the Softmax function, which is computationally expensive due to pairwise computations between queries and keys. These methods include using simple activation functions or custom mapping functions [18, 29, 6]. By changing the computation order, the overall complexity can be reduced to $\mathcal{O}(n)$. However, current linear attention methods still suffer from significant performance degradation compared to softmax attention and may introduce additional computational overhead, limiting their practical application.

Linear RNN models, such as Mamba [12], provide an efficient framework for sequence modeling by representing sequences as state spaces and utilizing scan operations, with linear time complexity. However, they may lack adaptability and dynamic characteristics required for complex sequence modeling tasks. Mamba addresses this by introducing selection mechanism and selective information transmission or forgetting through scan operations, but still shares limitations with traditional sequence models, such as the absence of feedback mechanism and adaptive control.

On the other hand, the MoE (Mixture of Experts) models [26, 17, 22] address these challenges by incorporating expert modules, which enable effective handling of long sequences while maintaining computational efficiency. The expert modules can adaptively select the appropriate experts for processing given input data, thus improving the model's computational efficiency and performance. Although the MoE models can efficiently handle long sequences and maintain computational efficiency, its design also brings some challenges. For instance, the sparse activation in the model's expert modules may lead to training stability issues. Additionally, since not all parameters in the model are frequently used, this may reduce parameter efficiency. Moreover, the MoE model may face challenges in computational efficiency and training stability when handling long sequences and may lack adaptability to dynamic changes.

Despite significant progress in the field of sequence modeling, existing models still face limitations and challenges. These include insufficient processing capabilities for long-range dependencies, lack of adaptability to dynamic changes, and limitations in computational resources. To overcome these challenges, we propose the MixCon, an innovative hybrid sequence modeling architecture that skillfully combines Transformer layers based on attention mechanism, Conba layers, and an MoE component.

The main contributions of this paper include:

- Proposal of the Conba architecture: We introduce a new sequence modeling architecture, which integrates feedback and adaptive

* Corresponding Author. Email: zlin@pku.edu.cn

control mechanism inspired by control theory. We provide a detailed mathematical formalization of Conba, including state space equations, adaptive control mechanism, and neural network approximation, offering a new perspective and method for sequence modeling.
- Integration with the MoE model: Utilizing the concept of an MoE model, we combine attention/Conba layer and MoE layer, achieving a flexible balance between low memory usage, high throughput, and high quality. This hybrid architecture not only provides efficient sequence modeling capabilities but also adapts to dynamic changes in sequences, thereby enhancing the model's performance and robustness.
- Demonstration of superior performance: On a single 80GB A800 GPU, MixCon offers a maximum context length that is fourteen times that of Llama-2-70B. In throughput analysis, it achieves a throughput three times higher than Mixtral and double that of Jamba on an 8K context length. Additionally, MixCon outperforms several publicly available models on various natural language processing tasks and demonstrates efficiency with better long-context throughput.

2 Related Work

2.1 Linear Attention Transformer

Efficient methods for computing attention, such as linear attention mechanism, have been proposed to address the computational challenges of traditional Softmax-based self-attention in deep learning, particularly in Transformer models, when dealing with large-scale data.

Efficient strategies, such as orthogonal random features in Performer [6], separate Softmax functions in Efficient attention [29], and a linear kernel in Castling-ViT [34], have been proposed to approximate Softmax operations and reduce computational complexity in self-attention mechanism.

Nyströmformer [33] and SOFT [23] utilize matrix decomposition techniques to approximate the self-attention matrix, with Nyströmformer employing the Nyström method for low-rank matrix approximation and SOFT employing different operations and optimization strategies.

Alternative softmax attention mechanisms, such as Hydra attention [3], EfficientVit [4], and Focused Linear Attention [14], provide novel computational approaches to achieve self-attention effects, reducing computational complexity, enhancing local feature extraction, and enabling concentrated information processing for specific regions. Infini-Transformer [24] introduces Infini-attention, integrating compressed memory and masked local/long-term linear attention mechanism to handle infinite-length input sequences. However, it introduces increased model complexity and potential challenges in training stability and computational efficiency.

Despite the improvement in computational efficiency, linear attention mechanism may still face challenges when processing long sequences, which may affect their performance and efficiency. Additionally, to maintain computational efficiency, these models may sacrifice some complexity and expressive power in certain cases.

2.2 Linear RNN Model

Linear RNN models have provided new solutions for sequence modeling tasks by optimizing the computational efficiency of traditional RNNs while maintaining model performance.

The RWKV model [25] combines efficient parallel training of Transformers and efficient reasoning of RNNs using linear attention mechanism, achieving comparable performance to similarly scaled Transformers while maintaining constant computational and memory complexity. It is currently the largest dense RNN model trained.

Structured state space models (SSMs), such as S4 [13] and its extension Mamba [12], offer efficient sequence modeling with linear time complexity and selective information transmission. However, they may lack adaptability and struggle to capture the complexity of sequences with changing dynamics or adapt to varying input distributions.

Hawk [10] is a linear architecture that incorporates residual patterns, MLP blocks, and a loop block with RG-LRU for temporal mixing. It outperforms Mamba in downstream tasks despite having fewer training tokens. Griffin [10] combines gated linear recurrence and local attention mechanism, excelling in language modeling tasks and achieving high efficiency in both training and inference stages, comparable to Transformers on TPU-v3.

However, models like Mamba, Hawk, and Griffin still inherit some limitations from traditional sequence models, such as the lack of feedback mechanism and adaptive control. These limitations may hinder their performance in dynamic and complex sequence modeling tasks, where adaptability and feedback are crucial.

2.3 MoE Model

MoE (Mixture of Experts) is a revolutionary technology that allows for a significant increase in the number of model parameters while having a relatively small impact on the computational efficiency of training and inference. The core of MoE models is the sparse activation mechanism, where each processed token only uses a subset of the parameters, greatly improving computational efficiency.

In MoE technology, the feedforward layers in Transformer models have become the standard target for various expert modules because these layers are relatively independent in computation and suitable for the design and deployment of expert modules. Recently, Mixtral 8×7B [17] have begun to be applied in open scenarios. It achieves comparable performance levels with only about 1/6 of the inference computational budget of Llama 2 70B [31].

Jamba [22] is a large language model based on a mixed Transformer-Mamba expert combination. By interleaving Transformer and Mamba layers, it combines the advantages of the two model series and achieves state-of-the-art performance in standard language model benchmarks and long context evaluations, particularly outstanding when the context length reaches 256K tokens.

However, hybrid expert models, while offering advantages in leveraging specialized knowledge, face challenges such as training stability, high computational complexity, and complex parameter management, requiring careful training strategies, efficient computational approaches, and effective parameter management techniques to ensure model performance and efficiency.

Our MixCon model is proposed to address the limitations and drawbacks of existing sequence modeling approaches, such as linear attention Transformers, MoE-Mamba, and linear RNN models, by effectively capturing long-range dependencies and enhancing the efficiency of sequence modeling.

3 Conba Model Architecture

Conba is based on the concept of modeling using state space and incorporates feedback and adaptive control mechanism from control

theory. The mathematical description of Conba includes state space equations, adaptive control mechanism, and neural network approximation. The architecture of Conba is depicted in Figure 1.

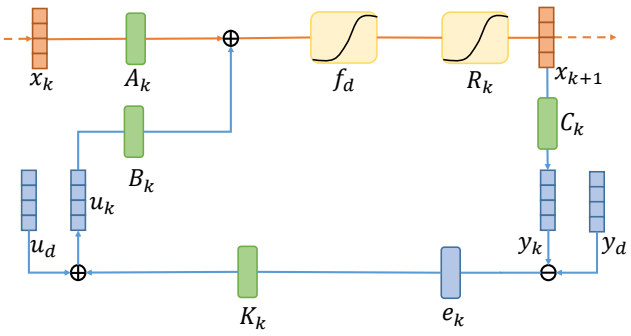

Figure 1. The architecture of Conba. Conba integrates state space, adaptive control, and neural network technologies.

3.1 State Space Equations

Conba represents sequence modeling tasks as a state space system. The state space of Conba is defined as follows:

$$x_{k+1} = f(x_k, u_k), \tag{1}$$

$$y_k = h(x_k), \tag{2}$$

where x_k is the state at time step k, u_k is the input at time step k, y_k is the output at time step k, respectively. The functions $f(\cdot)$ and $h(\cdot)$ are nonlinear functions that can be approximated by neural networks.

The state transition function $f(\cdot)$ captures the dynamic changes of the sequence state over time. It can be decomposed into a linear part A_k and a nonlinear part B_k:

$$f(x_k, u_k) = A_k x_k + B_k u_k, \tag{3}$$

where A_k and B_k are matrices with learnable parameters. The observation function $h(\cdot)$ maps the state x_k to the output y_k:

$$h(x_k) = C_k x_k, \tag{4}$$

where C_k is a matrix with learnable parameters.

To address the challenges posed by long sequences, Conba employs a selective state space mechanism. This mechanism enables the model to adaptively pass or forget information based on the current data. The selective state space function is defined as follows:

$$S(x_k, u_k) = Swish(W_k x_k + U_k u_k + v_k), \tag{5}$$

where W_k and U_k are matrices with learnable parameters, v_k is a vector with learnable parameters, and $Swish$ is the Swish activation function. The selective state space function determines the relevant information to be passed to the next state, thereby reducing computational complexity and improving efficiency.

To further enhance the model's ability to capture long-range dependencies, Conba introduces delayed state representation. The delayed state is defined as follows:

$$x^d_{k+1} = f_d(x_k, u_k), \tag{6}$$

where x^d_{k+1} is the delayed state at time step $k+1$, and $f_d(\cdot)$ is a nonlinear function representing delayed dynamics. The delayed state representation captures the influence of past states on the current state, enabling the model to effectively model long-range dependencies.

Conba further incorporates a dynamic state scaling mechanism to adaptively adjust the impact of different state dimensions on the output. The scaling function is defined as follows:

$$R_k(x_k) = Swish(Z_k x_k + g_k), \tag{7}$$

where Z_k is a matrix with learnable parameters, and g_k is a vector with learnable parameters. The scaling function adjusts the influence of each state dimension on the output, allowing the model to adapt to different input distributions and dynamically changing sequences.

To handle nonlinear and complex dependencies in sequences, Conba adopts a hybrid architecture that combines recurrent neural networks (RNNs) and feedforward neural networks (FNNs). RNNs capture temporal dependencies, while FNNs learn complex input-output relationships. The architecture is defined as follows:

$$x_{k+1} = R_k(f_d(x_k, u_k)), \tag{8}$$

$$y_k = h(x_k), \tag{9}$$

where $f(\cdot)$ and $h(\cdot)$ are defined as in (3) and (4). This hybrid architecture endows Conba with the ability to capture long-range dependencies and adapt to dynamic changes in sequences.

3.2 Adaptive Control Mechanism

Conba employs adaptive control mechanism to achieve the desired performance and adaptability. The design objective of the control mechanism is to minimize the tracking error between the actual output y_k and the desired output y_d:

$$u_k = -K_k(y_k - y_d) + u_d, \tag{10}$$

where K_k is the control gain matrix, y_d is the desired output, and u_d is the desired input. The control gain matrix K_k is updated through an adaptive mechanism:

$$K_{k+1} = K_k - \alpha \frac{\partial \|e_k\|_2}{\partial K_k}, \tag{11}$$

where $\|e_k\|_2$ denotes the 2-norm of the tracking error vector $e_k = y_k - y_d$, and α is the learning rate.

In summary, Conba leverages an adaptive control mechanism to achieve flexible adaptation and efficient control in sequence modeling tasks, thereby enhancing the model's performance and robustness to effectively manage various complex and dynamic scenarios.

3.3 Implementation Details

3.3.1 Neural Network Approximation

Conba employs neural networks to approximate the functions $f(\cdot)$, $h(\cdot)$, and the selective state space function $S(\cdot)$. The neural networks are trained end-to-end to determine the optimal parameters for each function. The architecture of these neural networks is crafted to capture the intricate relationships among inputs, states, and outputs, enabling Conba to proficiently model sequences with long-range dependencies.

To capture the dynamic changes of sequence states over time, Conba utilizes a Multi-Layer Perceptron (MLP) in the nonlinear part of the state transition function $f(\cdot)$. This MLP consists of multiple fully connected layers with Swish activation functions. The number of layers and the number of neurons per layer are tunable hyperparameters that can be adjusted to achieve optimal performance.

The observation function $h(\cdot)$ also employs an MLP for approximation, mapping the state x_k to the output y_k. The MLP for $h(\cdot)$ has the same architecture as that for $f(\cdot)$, facilitating its learning of complex input-output relationships.

The selective state space function $S(\cdot)$ is approximated using SwiGLU, a hybrid activation function based on Swish and GLU. SwiGLU captures temporal dependencies in sequences and determines the relevant information to pass to the next state based on the current input and hidden state. SwiGLU consists of two parts and is updated using the following equation:

$$\begin{aligned} S(x_k, u_k) &= SwishGLU(x_k, u_k) \\ &= Swish(W_k^1 x_k + U_k^1 u_k + v_k^1) \odot (W_k^2 x_k + U_k^2 u_k + v_k^2), \end{aligned} \tag{12}$$

where W_k^1, U_k^1, W_k^2, and U_k^2 are matrices with learnable parameters, and v_k^1 and v_k^2 are vectors with learnable parameters. $Swish$ is the Swish activation function, and $\odot$ denotes the element-wise product.

Conba leverages neural network approximations with MLPs and SwiGLU to adeptly capture intricate input-state-output relationships, model long-range dependencies, and dynamically adapt, while fine-tuning architecture and hyperparameters for optimal performance across diverse sequence modeling tasks.

3.3.2 *Further Expansion and Enhanced Robustness*

To further enhance the robustness of the model, we can introduce process noise w_k and observation noise v_k into the state space definition of Conba. The functions $f(\cdot)$ and $h(\cdot)$, which are represented by (3) and (4) respectively, are nonlinear functions that can be approximated using neural networks. The state transition function $f(\cdot)$ captures the dynamic changes of the sequence state over time and can be decomposed into a linear part A_k and a nonlinear part B_k:

$$f\left(x_k, u_k, w_k\right) = A_k x_k + B_k u_k + w_k, \tag{13}$$

where A_k and B_k are matrices with learnable parameters. The observation function $h(\cdot)$ maps the state x_k to the output y_k:

$$h\left(x_k, v_k\right) = C_k x_k + D_k v_k, \tag{14}$$

where C_k and D_k are matrices with learnable parameters.

The incorporation of w_k and v_k enhances Conba's resilience to perturbations by introducing noise, enabling adaptation to data uncertainties, with future research exploring noise types and intensities to optimize robustness across sequence modeling tasks.

4 Model Architecture

MixCon is an innovative hybrid decoder architecture that skillfully combines Transformer layers based on attention mechanism, Conba layers, and MoE components. As shown in Figure 2, this unique combination allows MixCon to achieve a flexible balance between low memory usage, high throughput, and high quality, although these goals may sometimes conflict. In terms of memory usage, it is important to note that the total number of model parameters does not always reflect the actual memory requirements. In MoE models, only the active parameters involved in forward propagation actually occupy memory, and the total number of parameters may be much larger than the number of active parameters. Additionally, the KV cache (used to store attention keys and values in the context) is a key factor. When extending the Transformer model based on attention to handle long contexts, the KV cache may become a performance bottleneck. By carefully balancing between the attention and Conba layers, we can significantly reduce the total size of the KV cache. Our architecture design not only provides very few active parameters but also reduces the KV cache by 32 times compared to Mamba. The comparison results in Table 1 show that even in a 256K token context, MixCon can maintain a smaller KV cache advantage.

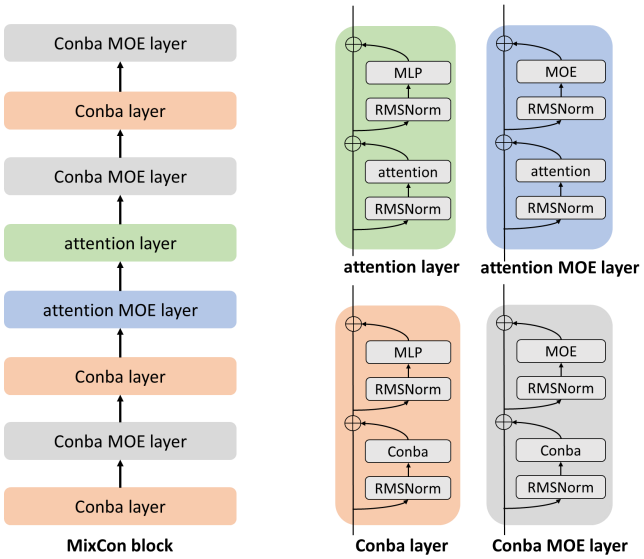

Figure 2. Left: A single MixCon block; Right: Different colors correspond to different layers in the left figure.

In terms of throughput, for short sequences, the proportion of FLOPS (floating-point operations) taken up by attention operations during inference and training is relatively small. However, when processing long sequences, attention operations become the main computational burden. In contrast, the Conba layer is more computationally efficient. Therefore, increasing the proportion of Conba layers, especially when processing long sequences, can significantly improve overall throughput.

This paper describes a primary configuration aimed at providing improved performance and efficiency. The basic unit of this configuration is a MixCon block, which can be repeatedly used in a sequence. Each MixCon block is composed of a combination of Conba or attention layers. Each such layer contains an attention module or a Conba module, followed by a multi-layer perceptron (MLP) or MoE layer. Figure 2 (right) illustrates examples of different types of layers. A MixCon block consists of 8 layers. In the MixCon, the MLP layer is replaced by an MoE layer, which helps increase the model's capacity while maintaining a relatively small number of active parameters, thus keeping the computational load relatively low.

For the implementation of the Conba layer, we adopt several normalization techniques to help stabilize training at large model scales. Specifically, we apply RMSNorm [36] to the Conba layer. Other architectural details include grouped query attention (GQA) [1], SwiGLU activation function [7], and MoE load balancing [11].

Table 1. Comparison of MixCon with recent open models in terms of total available parameters, active parameters, and KV cache memory in long contexts.

	Available params	Active params	KV cache (256K context, 16bit)
Llama-2 [31]	6.7B	6.7B	128GB
Mamba [12]	3.0B	3.0B	64GB
Hawk [10]	7.0B	7.0B	32GB
Griffin [10]	14.0B	14.0B	32GB
Infini-Transformer [24]	7B	7B	32GB
Mixtral [17]	46.7B	12.9B	32GB
Jamba [22]	52B	12B	4GB
MixCon (Ours)	60B	5B	2GB

The vocabulary size of the model is 256K. The tokenizer uses BPE for training [28], with each digit being a separate token.

5 Experiments and Evaluations

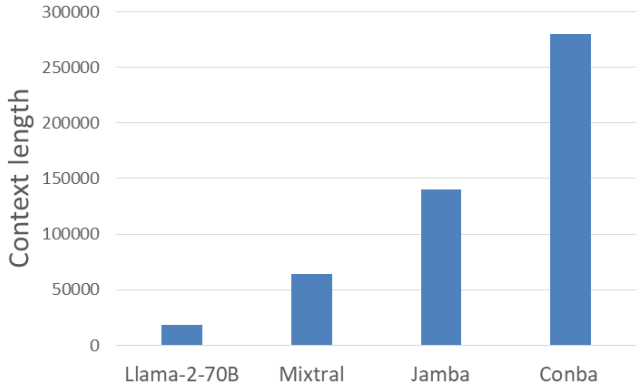

Figure 3. MixCon offers a maximum context length that is twice that of Jamba, four times that of Mixtral, and fourteen times that of Llama-2-70B on a single A800 80GB GPU.

5.1 Implementation Details

In the specific implementation, we carefully select a specific configuration to adapt to the computational capabilities of a single 80GB A800 NVIDIA GPU while achieving optimal performance in terms of quality and throughput. In our implementation, we have a sequence consisting of 4 MixCon blocks. Each MixCon block contains 8 layers ($L = 8$), with a ratio of $2 : 6$ between attention layers and Conba layers ($a : c = 2 : 6$). Every other layer ($e = 2$), we choose to replace the MLP module with MoE. The model has a total of 16 experts ($n = 16$), and each token uses 2 top-level experts ($K = 2$). We selected the ratio of $a : c = 2 : 6$ in our preliminary ablation experiments as it was one of the variants with the highest computational efficiency among the best-performing variants in terms of quality. These settings are designed to improve the performance and efficiency of the model.

5.2 Context Length Analysis

The configuration of the experts is designed to make the model compatible with a single 80GB A800 GPU (with int8 weights) while having enough memory for inputs. Additionally, we balanced n, K, and e to ensure high quality while keeping computational requirements and communication dependencies (memory transfers) under control.

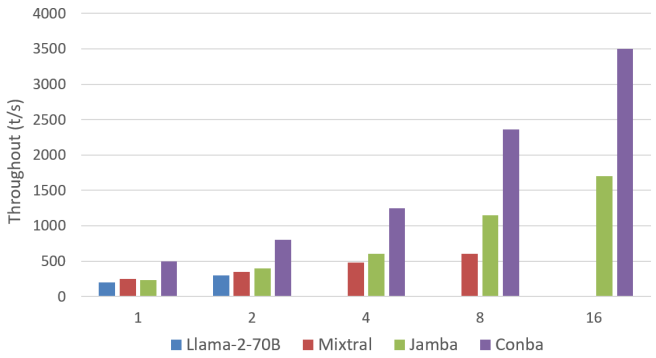

Figure 4. ConMix achieves a throughput three times higher than Mixtral and twice as high as Jamba when considering different batch sizes on a single A800 GPU with an 8K context length.

Hence, we chose to replace the MLP module with MoE every other layer, resulting in a total of 16 experts, with each token utilizing two experts. These choices were inspired by previous work on MoE and validated through preliminary experiments.

Figure 3 illustrates the maximum context length accommodated by our MixCon implementation on a single 80GB A800 GPU compared to Mixtral 8x7B and Llama-2-70B. The context length provided by Conba is twice that of Jamba, four times that of Mixtral, and fourteen times that of Llama-2-70B.

5.3 Throughput Analysis

To provide specific insights, we present the throughput results for two particular configurations. In the first configuration, we consider different batch sizes, a single A800 80GB GPU with int8 quantization, an 8K context length, and generate an output of 512 tokens. As shown in Figure 4, MixCon enables efficient processing of large batches, resulting in a throughput (tokens/second) three times higher than Mixtral and double that of Jamba.

In the second configuration, we focus on a single batch (batch size = 1), four A800 GPUs without quantization, varying context lengths, and generate an output of 512 tokens. As depicted in Figure 5, for shorter context lengths, all models exhibit similar throughput. However, MixCon excels in handling long contexts, achieving a throughput 1.5 times higher than Jamba and 4.5 times higher than Mixtral when considering 128K tokens.

5.4 Dataset Evaluation

We report the results of a range of standard academic benchmark tests to comprehensively evaluate the performance of the Conba

Table 2. Comparison of MixCon with other public models, where MixCon achieves similar or better performance.

	HellaSwag	WinoGrande	ARC-E	ARC-Challenge	BoolQ	QuAC	MMLU	BBH
Llama-2 [31] 70B	85.3	80.2	80.2	67.3	85.0	42.4	69.8	51.2
Mamba [12]	85.7	80.5	80.6	67.8	84.6	41.9	70.6	51.6
Hawk [10]	85.9	81.6	**81.9**	70.2	85.1	42.6	**71.6**	51.1
Griffin [10]	86.2	81.9	81.6	**70.3**	85.6	43.4	70.1	**51.9**
Infini-Transformer [24] 7B	87.6	82.4	76.9	69.5	**88.9**	41.6	70.2	50.1
Mixtral [17]	86.7	81.2	77.6	66.2	88.4	40.9	70.6	50.3
Jamba [22]	87.1	82.5	73.5	64.4	88.2	40.9	67.4	45.4
MixCon (Ours)	**87.9**	**83.4**	81.3	69.8	88.6	**44.6**	70.5	51.6

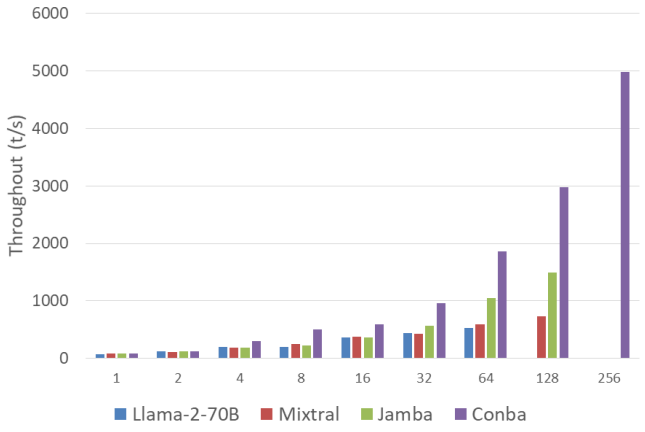

Figure 5. A single batch and four A800 GPUs, ConMix demonstrates a throughput that is 1.5 times higher than Jamba and 4.5 times higher than Mixtral when considering 128K tokens.

Table 3. Ablation Experiment Results on Academic Benchmark Evaluation Dataset.

	HellaSwag	WinoGrande	NQ
Only Attention	57.6	61.9	17.1
Only Conba	55.6	62.5	19.6
MixCon($a:c$ = 1:7, no MOE)	**57.9**	62.5	20.2
MixCon($a:c$ = 2: 6, no MOE)	57.8	**62.6**	**21.2**

model across various natural language processing tasks. These tasks include:

- We assess the performance of Conba on four common-sense reasoning tasks: HellaSwag [35], WinoGrande [27], ARC-E and ARC-Challenge [9]. These tasks aim to evaluate the model's ability to understand and apply common-sense knowledge. We employ 10-shot, 5-shot, 0-shot, and 25-shot learning strategies for these tasks, respectively.
- We examine the performance of Conba on two tasks: BoolQ [8] and QuAC [5]. These tasks assess the model's comprehension and reasoning abilities on reading materials. We use 10-shot and zero-shot learning strategies for BoolQ and QuAC, respectively.
- We also test Conba on two aggregation benchmark tests: MMLU [15] and BBH [30]. These tasks evaluate the model's understanding and application of knowledge across multiple domains. We employ 5-shot and 3-shot learning strategies for MMLU and BBH, respectively.

Through these benchmark tests, we are able to comprehensively evaluate Conba's performance across different natural language processing tasks and validate its adaptability to various task requirements. Table 2 provides a detailed comparison of MixCon's performance against several publicly available models on common academic benchmarks for evaluating language models. Notably, MixCon's performance matches or exceeds that of similarly or larger-scale advanced publicly available models, including Llama-2 70B and Mixtral. Despite having fewer total parameters than Llama-2 (60B vs 70B), MixCon is a sparse model with only 5B active parameters, in contrast to Mixtral's 12.9B active parameters. Furthermore, as a model based on softmax attention, Mixtral has a larger memory footprint when processing long sequences, requiring 32GB of KV cache to handle 256 tokens. In contrast, due to its hybrid MixCon architecture, even in such long contexts, MixCon only requires 2GB of KV cache. In summary, MixCon demonstrates the capability of a hybrid architecture to achieve performance comparable to state-of-the-art Transformer models of the same scale, while also benefiting from SSM advantages.

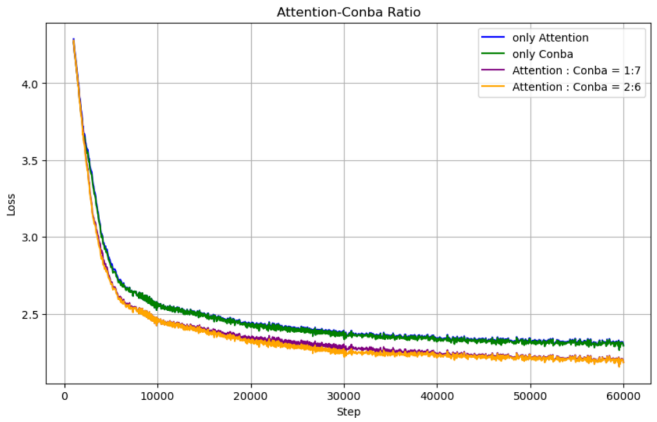

Figure 6. The training loss curves for Attention, Conba, and Attention-Conba hybrid models with different Attention/Conba ratios ($a:c$ = 1:7 and 2:6)

5.5 *Ablation Experiments*

In this section, we delve into the ablation experiments conducted during the implementation of the Conba architecture, focusing on different design choices. We first showcase the benefits of combining Attention and Conba layers, as well as the optimal proportion and interleaving techniques for their combination. Through these experiments, we discovered that the pure Conba model faced difficulties in contextual learning, while the Attention-Conba hybrid demonstrated contextual learning capabilities similar to pure Transformer models. Subsequently, we explore the advantages of adding the MoE layer on top of the Attention-Conba hybrid model.

Table 4. Results (F1) of MixCon's long-context QA benchmark test, using a 3-shot format.

	LongFQA	CUAD	NarrativeQA	NQ	Avg
Llama-2 [31]	0.38	0.41	0.29	0.42	0.38
Mamba [12]	0.32	0.42	**0.39**	0.38	0.38
Infini-Transformer [24]	0.43	0.42	0.36	**0.68**	0.47
Mixtral [17]	0.42	0.46	0.29	0.58	0.44
Jamba [22]	0.44	0.44	0.30	0.60	0.44
MixCon (Ours)	**0.45**	**0.48**	0.38	0.62	**0.48**

Table 5. Comparison of attention, Mamba, and MixCon on academic benchmarks with a 7B model size and trained on 50B tokens.

	HellaSwag	WinoGrande	NQ
attention	60.4	59.7	13.7
Mamba	60.2	55.8	14.0
MixCon ($a : c$ = 2:6, no MOE)	57.9	72.5	20.2
MixCon ($a : c$ = 2:6, MOE)	**67.8**	**72.6**	**21.2**

Table 6. Enhancing MixCon with MOE.

	HellaSwag	WinoGrande	NQ
MixCon (no MoE)	57.9	72.5	20.2
MixCon + MoE	**63.2**	**73.6**	**22.1**

For these ablation experiments, we report the following metrics, which demonstrate good informative performance even with small datasets or smaller model scales: HellaSwag (10-shot) [35], WinoGrande (5-shot) [27], and Natural Questions closed-book (NQ, 5-shot) [21]. The results in Table 3 indicate that even with limited resources, the MixCon exhibits robust performance.

Figure 6 illustrates the training loss of these three architectures. While the pure Transformer and Mamba exhibit similar convergence patterns, the MixCon (without MoE) has lower loss throughout the entire training process.

5.6 Long Context Evaluation

We evaluate the ability of MixCon to handle long contexts using question-answering benchmark tests that contain lengthy inputs. To do so, we reutilize five datasets from the longest-context dataset in L-Eval [2] and structure them in a few-shot format (using three examples in all experiments).

Specifically, we evaluate the model on the following datasets: NarrativeQA (narrative question-answering) [20], LongFQA (finance) [2], Natural Questions (NQ; Wikipedia)[21], CUAD (legal) [16]. The average input lengths in these datasets range from 6K to 62K tokens. These context lengths are further augmented through the few-shot format.

Table 4 summarizes the evaluation results in terms of F1 score. MixCon outperforms Mixtral and Jamba on most datasets and achieves superior performance on average. Additionally, due to the computationally intensive nature of these long-context tasks, MixCon demonstrates efficiency with better long-context throughput.

5.7 Benefits of Combining Attention and Conba

First, we investigate the ratio ($a : c$) between the attention layer and the Conba layer using a model with 1.3 billion parameters trained on 250 billion tokens. As shown in Table 5, the performance of MixCon surpasses that of pure attention or pure Mamba. The ratio between the attention and Conba layers can be 2:6 or 1:7, with little performance difference observed.

Next, we compare the performance of pure Transformer, pure Mamba, and MixCon hybrid models with a model size of 7B parameters and trained on 50B tokens. As shown in Table 5, the pure Mamba layer is competitive but slightly inferior to pure attention. However, MixCon outperforms both pure models while achieving better throughput than the pure Transformer.

5.8 The Impact of Mixed Experts

Recent studies suggest that the use of Mixture of Experts (MoE) technology can enhance the performance of Transformer-based language models while maintaining manageable computational resources. Moreover, preliminary research findings indicate that MoE can improve the performance of the Mamba layer, even when dealing with smaller-scale models and datasets. However, there is a lack of definitive evidence regarding whether MoE can be effectively integrated with large-scale state space models and our hybrid MixCon architecture. In fact, the data presented in Table 6 demonstrates a significant performance improvement when MoE technology is applied in a large-scale context (5B parameters, trained on 50B tokens) of the MixCon architecture. In the MoE variant, there are a total of $n = 16$ experts, with $K = 2$ experts used per token.

6 Conclusion

This paper presents MixCon, an innovative hybrid sequence modeling architecture aimed at tackling the challenges of long-range dependency capture and efficient sequence modeling. By integrating Transformer layers, Conba layers, and a mixed expert component (MoE), MixCon effectively manages complex, dynamic sequences with high computational efficiency. Our experiments show its significant advantages across various tasks, with efficient long sequence processing, low memory usage, and high throughput, making it highly scalable and practical for large, complex datasets. Despite its strengths, MixCon's performance could be enhanced by refining state space representation, adaptive control for long sequences, and domain-specific fine-tuning, along with optimizing training algorithms to minimize computational demands. Overall, MixCon offers a novel solution for sequence modeling, showcasing its prowess in complex sequence handling and paving the way for transformative applications in NLP and beyond.

7 Acknowledgment

Z. Lin was supported by National Key R&D Program of China (2022ZD0160300), the NSF China (No. 62276004), and Qualcomm.

References

[1] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[2] C. An, S. Gong, M. Zhong, M. Li, J. Zhang, L. Kong, and X. Qiu. L-eval: Instituting standardized evaluation for long context language models. *arXiv preprint arXiv:2307.11088*, 2023.

[3] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, and J. Hoffman. Hydra attention: Efficient attention with many heads. In *European Conference on Computer Vision*, pages 35–49. Springer, 2022.

[4] H. Cai, C. Gan, and S. Han. Efficientvit: Enhanced linear attention for high-resolution low-computation visual recognition. *arXiv preprint arXiv:2205.14756*, 2022.

[5] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*, 2018.

[6] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[7] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[8] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[9] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[10] S. De, S. L. Smith, A. Fernando, A. Botev, G. Cristian-Muraru, A. Gu, R. Haroun, L. Berrada, Y. Chen, S. Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

[11] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[12] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

[13] A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[14] D. Han, X. Pan, Y. Han, S. Song, and G. Huang. Flatten transformer: Vision transformer using focused linear attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5961–5971, 2023.

[15] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[16] D. Hendrycks, C. Burns, A. Chen, and S. Ball. Cuad: An expert-annotated nlp dataset for legal contract review. *arXiv preprint arXiv:2103.06268*, 2021.

[17] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[18] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[19] N. Kitaev, Ł. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[20] T. Kočiskỳ, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

[21] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[22] O. Lieber, B. Lenz, H. Bata, G. Cohen, J. Osin, I. Dalmedigos, E. Safahi, S. Meirom, Y. Belinkov, S. Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

[23] J. Lu, J. Yao, J. Zhang, X. Zhu, H. Xu, W. Gao, C. Xu, T. Xiang, and L. Zhang. Soft: Softmax-free transformer with linear complexity. *Advances in Neural Information Processing Systems*, 34:21297–21309, 2021.

[24] T. Munkhdalai, M. Faruqui, and S. Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

[25] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, H. Cao, X. Cheng, M. Chung, M. Grella, K. K. GV, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

[26] M. Pióro, K. Ciebiera, K. Król, J. Ludziejewski, and S. Jaszczur. Moe-mamba: Efficient selective state space models with mixture of experts. *arXiv preprint arXiv:2401.04081*, 2024.

[27] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[28] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[29] Z. Shen, M. Zhang, H. Zhao, S. Yi, and H. Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter Conference on Applications of Computer Vision*, pages 3531–3539, 2021.

[30] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[31] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[33] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148, 2021.

[34] H. You, Y. Xiong, X. Dai, B. Wu, P. Zhang, H. Fan, P. Vajda, and Y. Lin. Castling-vit: Compressing self-attention via switching towards linear-angular attention during vision transformer inference. *arXiv preprint arXiv:2211.10526*, 2022.

[35] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[36] B. Zhang and R. Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.