Yan: A Linear-Time and Stable-Memory Model

A
Rock AI
Shanghai, China
Email:

B
Rock AI
Shanghai, China
Email:

C
Rock AI
Shanghai, China
Email:

Abstract*—Transformers have been broadly adopted in the sphere of Natural Language Processing (NLP) owing to their capacity to process data concurrently and make use of global attention mechanisms. Nevertheless, the employment of matrix cross-multiplication in self-attention gives rise to high computational complexity and slow reasoning speed. To address these constraints, we propose Yan, a novel language model architecture featuring lightweight and fast reasoning capabilities. Unlike the transformer that merely focuses on the information among tokens, the proposed Yan pays more attention to the information among elements and is more elaborate in feature extraction. The core of our proposed Yan lies in the Yan block, which extracts the current and historical information in the input data via two sections, slope and decay, and fuses the information from the element's perspective so as to better express the model. During the inference phase, we optimized the inference process into the form of a recurrent representation, bringing down the time complexity and space complexcity to $O(N)$ and $O(1)$ respectively. Our experiments show that Yan attains higher throughput and lower GPU memory consumption compared to Transformers, while maintaining comparable performance to larger-scale LLMs on benchmark tests. Owing to these fine performances, the proposed Yan present a sound foundation for further development in end-side deployment and embodied intelligence.

1. Introduction

In recent years, the field of Natural Language Processing (NLP) has seen tremendous advancements, with the development of Large Language Models(LLMs) being a significant breakthrough [20]. These models have changed the manner in which we interact with machines, enabling them to understand and produce human-like language. Among them, Transformer dominates in NLP due to its powerful performance. [1]. Benefiting from its self-attention mechanism, Transformer has long-range dependencies that substantially improve the ability to process language. Transformer-based LLMs trained on extensive datasets sourced from the web have achieved remarkable success [6] [7] [8]. Nevertheless, due to the limitations of self-attention, Transformer possesses an $O(N^2)$ time complexity and an $O(N)$ [21] space complexity in inference process. Such limitations impede the progress of transformer-based LLMs.

AFT [5] simplifies the computation of the softmax function by altering the sequence of matrix multiplication in self-attention, thereby lowering the time complexity to $O(N)$ and the space complexity to $O(1)$, which notably enhances the speed of inference. RWKV [2] and RetNet [3] re both based on variants of AFT, having an inference complexity of $O(1)$ [3], which is comparatively fast and capable of handling long sequential. However, they still utilize the QKV ariant as the attention network for operations, and the training time remains considerable. Mamba is based on the evolution of the State Space Model (SSM), completely doing away with the self-attention structure, and expanding the input information to relatively high levels and hidden dimensions. The model has been shown to function efficiently and is notably lightweight. Nevertheless, experimental evidence indicates difficulties in effectively scaling Mamba [9].

Though some results have been attained, there still exist some persistent issues that necessitate further examination. Thus, it is of significance to devise a novel model that can effectively optimize the aforementioned problems from a modeling standpoint. In this undertaking, we propose the Yan model, a novel model that does not entail the intricate computation of the QKV variant as the attention network. The proposed Yan is still possible to boast good performance while significantly simplifying the amount of computation.

This work makes the following contributions:

(1) A novel Yan model has been developed, extracting input context information via a weighted linear summation operation, thereby doing away with the need for arduous matrix multiplication. Specifically, the proposed model extracts both current and historical element information through the slope and decay sections respectively. Compared to Transformer, the information extracted is more comprehensive.

(2) To optimize the conveyance of information, we employ a multi-channel framework that assigns distinct weights to each channel. Subsequently, these weighted channels are seamlessly integrated with contextual cues, thereby bolstering the diversity of information fusion. This approach effectively elevates the overall representational power of the model.

(3) A recurrent representation has been created and specif-

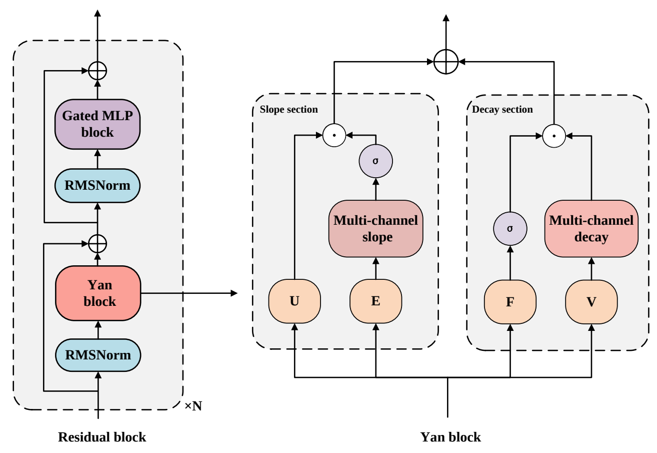

Figure 1. The main backbone of our mode architecture is the residual block, which is stacked N times. (left) The Yan block that two sections are proposed. (right)

ically fine-tuned for the inference process. The generation of the subsequent token output merely requires the computation of the state amount and the current input, bypassing the need to calculate all input tokens. This simplified approach notably hastens the inference speed.

(4) In inference experiments, the presented Yan is placed in comparison with Transformer having an equivalent number of parameters across three metrics: GPU memory, latency, and throughput. The results indicate that the proposed Yan surpasses Transformer under the same conditions in terms of the cost of inference. Provided that the inference cost performs well, our proposed Yan is still of high caliber in terms of model performance. These superiorities increase the potential for widespread applicability of the Yan model on the end-user side.

2. Model Architecture

Here, we introduce the architecture of our proposed model. In 2.1, we present the backbone of the proposed model, which is composed of N residual blocks. In 2.2, we introduce the Yan block, which constitutes the core of the proposed model. In 2.3, we formulate and simplify the inference process, thus allowing the proposed model to possess $O(N)$ time complexity and $O(1)$ space complexity in the inference process.

2.1. Residual block

The backbone of our proposed model is presented in Figure 1, wherein we define the global structure of our model via residual blocks as shown in Figure 1(left), which are inspired by pre-norm Transformers [10]. The core of our proposed model is Yan block. After embedding, we superimpose N Residual blocks, wherein we apply RMSNorm [11] as the normalization function and we utilize Yan block to extract contextual information. And then RMSNorm and gated MLP block [12] are utilized to undertake a depth analysis on the output information of Yan block. During the gated MLP block, we apply GeGeLU [15] as the activation function and combine it with MLP.

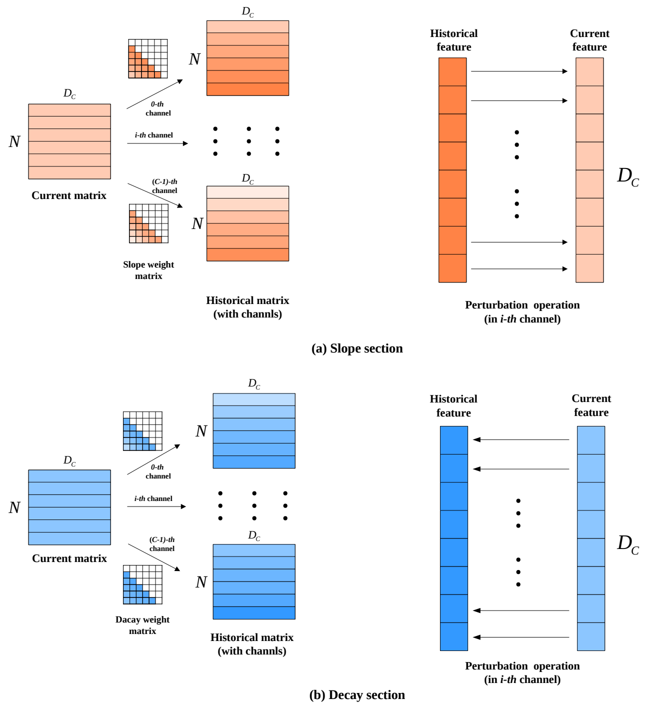

Figure 2. Specific implementations of the slope and decay sections. As can be seen from the figure, in the slope section, the model retains as much information about the current position as possible, while in the decay section, the model retains as much historical information as possible.

2.2. Yan block

The Yan block is as illustrated in Figure 1(right), is divided into two sections: the slope section and the decay section. Each section is further divided into two branches. The specific implementation of the core of these two sections is shown in Figure 2. In the slope section(shown in Figure 2(a)), the current information is perturbed by historical information, resulting in the current information containing a component of historical information. In the decay section(shown in Figure 2(b)), historical information is perturbed by the current information, resulting in the historical information containing a component of the current information. Prior to the specific calculation, a dimensional transformation operation is performed on the input X, $X \in \mathbb{R}^{N\times D} \rightarrow X \in \mathbb{R}^{C\times N\times D_c}$, where N denotes sequence length, D denotes sequence dimension, $D = C \times D_c$, C represents the number of channels, and D_c represents the sequence dimension of each channel.

Slope section After the input data has been dimensionally transformed, the slope section yields the matrix U,E as illustrated in Eq. (1) by means of a linear transformation of the input X, where $W_U, W_E \in \mathbb{R}^{C\times D_C\times D_C}$; $U, E \in \mathbb{R}^{C\times N\times D_c}$. One of the branches performs a multi-channel slope operation on the matrix E, setting the weights of the history information according to different channels. This is followed by probabilization of the history perturbation information of the input data using the softmax function, after which the history perturbation is added to the input matrix of the other branch of the slope section. The multi-channel slope operation is shown in Eq. (2-5).

$$U = XW_U, E = XW_E \tag{1}$$

$$\beta_i = \left(2^{-8/C}\right)^{i+1}, i \in \{0, 1, ...C-2, C-1\} \tag{2}$$

$$s_i = \begin{bmatrix} -\infty & \cdots & \cdots & -\infty \\ -\beta_i & -\infty & \ddots & \vdots \\ \vdots & -\beta_i & -\infty & \vdots \\ (-N+1)\,\beta_i & \cdots & -\beta_i & -\infty \end{bmatrix} \tag{3}$$

$$W_s = (s_0, s_1, ..., s_{C-2}, s_{C-1}) \tag{4}$$

$$E' = softmax\,(W_s)\,E \tag{5}$$

In the slope section, different weights β_i, are assigned to the i_{th} different channels. These weights are defined in Eq. (2), where C denotes the number of channels. In addition to applying different weights to each channel, the input history information is extracted. In contrast to Transformer, which employs the transpose of K for the extraction of contextual information, we have devised a more lightweight approach, influenced by [17], which defines the weight matrix s_i as in Eq. (3), where $s_i \in \mathbb{R}^{N\times N}$ represents the weight matrix corresponding to the i_{th} channel, and the upper-triangular portion of the matrix is negatively infinite. Given that our input data has been reshaped into a three-dimensional matrix, divided into C channels, it is necessary to perform an aggregation of the weight matrix s_i into a three-dimensional matrix, as shown in Eq. (4). Given that the magnitude of the matrix $W_s \in \mathbb{R}^{C\times N\times N}$ varies and is not probabilistic, it is necessary to normalize it probabilistically by performing a $softmax$ operation on it. Since $e^{-\infty} \to 0$, the upper triangular portion of matrix W_s tends to 0 after the $softmax$ operation, which is equivalent to the masking of the current and subsequent information for the historical information that does not include the current information. Subsequently, matrix W and matrix E are multiplied by each other. Here, the essence of matrix multiplication lies in the weighting and manipulation of the historical tokens of input X, thereby imbuing the output tokens with information from historical tokens. This process significantly accelerates computation speed. This process greatly speeds up the computation, as it allows the output of the tokens to contain the information of the historical tokens. The specific operation is shown in Eq. (5), where $E' \in \mathbb{R}^{C\times N\times D_c}$ denotes the matrix E output after the multi-channel slope operation.

At the conclusion of the slope section, a perturbation is introduced to the current input information U. The gating function sigmoid is incorporated into the output of E' with the objective of conditionally extracting information from E and controlling the magnitude, thereby facilitating the addition of a perturbation to the current information. The specific formula is shown in Eq. (6). In this equation, σ represents the sigmoid function, and $\Theta_s \in \mathbb{R}^{C\times N\times D_c}$ represents the output of the slope section.

$$\Theta_s = \sigma\,(E') \odot U \tag{6}$$

The matrix s_i enables the slope operation to assign greater weights to more recent historical information and smaller weights to more distant information. Consequently, the current information has minimal impact on information from long ago. However, the use of the slope section alone is not without limitations.

Decay section Similarly, after the input data has been subjected to a dimensional transformation, the matrix F, V is obtained by means of a linear transformation of the input X, as illustrated in Eq. (7), where $W_F, W_V \in \mathbb{R}^{C\times D_C\times D_C}$; $F, V \in \mathbb{R}^{C\times N\times D_c}$. One of the branches performs a multi-channel decay operation on the matrix V, which is analogous to the slope section, with the exception that the weight matrix is distinct. The multi-channel decay operation is illustrated in Eq. (8-11).

$$F = XW_F, V = XW_V \tag{7}$$

$$\alpha_i = 1 - 2^{-5-i}, i \in \{0, 1, ..., C-2, C-1\} \tag{8}$$

$$d_i = \begin{bmatrix} -\infty & -\infty & \cdots & \cdots & -\infty \\ \alpha_i^1 & -\infty & \ddots & \ddots & \vdots \\ \vdots & \ddots & -\infty & \ddots & \vdots \\ \alpha_i^{N-2} & \alpha_i^{N-3} & \ddots & -\infty & -\infty \\ \alpha_i^{N-1} & \alpha_i^{N-2} & \cdots & \alpha_i^1 & -\infty \end{bmatrix} \tag{9}$$

$$W_d = (d_0, d_1, ..., d_{C-2}, d_{C-1}) \tag{10}$$

$$V' = RMSNorm\,(W_d)\,V \tag{11}$$

In the decay section, we define different decay matrices derived from the slope section, as illustrated in Equations (8-9). In accordance with the recommendations set forth by [3], we define distinct values of α_i for each distinct channel. We then incorporate these α values into the matrix d, thereby obtaining different $d_i \in \mathbb{R}^{N\times N}$ matrices. In Eq. (10), the d_i matrices are integrated into $W_d \in \mathbb{R}^{C\times N\times N}$ under different channels in order to ascend the dimension, which facilitates subsequent operations. As in Eq. (11), performs a multi-channel decay operation on the matrix V to extract the history information of the input data and normalize it. However, in contrast to the multi-channel slope, the multi-channel decay operation still assigns a relatively high weight to more distant history information, thus ensuring that matrix $V' \in \mathbb{R}^{C\times N\times D_c}$ encompasses as much global history information as possible.

At the conclusion of the decay section, a perturbation is introduced to the processed history information V', and a gating function sigmoid is incorporated into the current information F at the output. This is done with the intention of performing conditional information extraction and controlling the magnitude of F, in order to facilitate the introduction of a perturbation to the history information. The specific formula is shown in Eq. (12) where σ represents the sigmoid function, and $\Theta_d \in \mathbb{R}^{C\times N\times D_c}$ represents the output of the decay section.

$$\Theta_d = V' \odot \sigma\left(F\right) \tag{12}$$

$$\Theta = Concat\left(\Theta_s, \Theta_d\right) \tag{13}$$

Due to the limitations of utilizing the slope section in isolation, we have extracted the global history information through the decay section. This approach enables the final output to extract the more recent history information without losing the more distant history information, and to enhance the capability of modeling long texts. The outputs of the slope section and the decay section are combined to create a single, comprehensive output from the Yan block. This output contains both current and historical information, enhancing the overall completeness of the information without losing the differentiation. This is demonstrated in equation (13), where $\Theta \in \mathbb{R}^{C\times N\times 2D_c}$ denotes the output of the Yan block.

2.3. Fast inference for the Yan block

The core factor contributing to the slow inference speed of Transformers is the softmax function and the matrix cross-multiplication calculation of the matrix QK, which hampers its tractability when reduced to a recursive form for inference. In order to avoid this disadvantage, we proceeded to eliminate the inclusion of matrix cross-multiplication in the model design and optimize the inference process by derivation into the form of recurrent representation. This transformation resulted in a sapce complexity of $O(1)$ and a time complexity of $O(N)$ in inference process, significantly reducing the inference time. The optimized inference steps in the multi-channel slope are presented in Eq. (14-16), while the optimized inference steps in the multi-channel decay are shown in Eq. (17-19).

$$E'_n = \frac{\sum_{j=1}^{n-1} e^{(-j)\beta_i} E_{n-j}}{\sum_{j=1}^{n-1} e^{(-j)\beta_i}} \tag{14}$$

$$E'_{n+1} = \frac{\sum_{j=1}^{n} e^{(-j)\beta_i} E_{n+1-j}}{\sum_{j=1}^{n} e^{(-j)\beta_i}} \tag{15}$$

$$\begin{aligned} S_{n+1}^{slope} = E'_{n+1} &= e^{(-1)\beta_i} E'_n \frac{\sum_{j=1}^{n-1} e^{(-j)\beta_i}}{\sum_{j=1}^{n} e^{(-j)\beta_i}} + \frac{e^{(-1)\beta_i} E_n}{\sum_{j=1}^{n} e^{(-j)\beta_i}} \\ &= (1 - \frac{1}{\sum_{j=0}^{n-1} e^{(-j)\beta_i}}) S_n^{slope} + \frac{1}{\sum_{j=0}^{n-1} e^{(-j)\beta_i}} E_n \end{aligned} \tag{16}$$

In the context of multi-channel slope and multi-channel decay inputs, $E_j \in \mathbb{R}^{1\times D_c}$ and $V_j \in \mathbb{R}^{1\times D_c}$ represent the j_{th} vector representation in sequence, while n denotes the number of currently inputted vector representations in sequence. $S_{n+1}^{slope} \in \mathbb{R}^{1\times D_c}$ and $S_{n+1}^{decay} \in \mathbb{R}^{1\times D_c}$ represent the next vector representation(with weight 0 at position n) that is inferred under n E_i and V_i inputs, respectively. The i_{th} channel weights, β_i and α_i, are defined by Eq. (2), Eq. (8) , respectively, and represent the channel weights under multi-channel slope and multi-channel decay, respectively.

$$V'_n = \sum_{j=1}^{n-1} \alpha_i^j V_{n-j} \tag{17}$$

$$V'_{n+1} = \sum_{j=1}^{n} \alpha_i^j V_{n+1-j} \tag{18}$$

$$\begin{aligned} S_{n+1}^{decay} &= V'_{n+1} = \alpha_i V'_n + \alpha_i V_n \\ &= \alpha_i S_n^{decay} + \alpha_i V_n \end{aligned} \tag{19}$$

In the original definition (e.g., Eq. (14-15), Eq.(17-18)), when reasoning about the data of the next token, it is necessary to compute all the data, which results in a significant increase in computational complexity. In optimized formulations such as Eq. (16), Eq. (19), the next output token is only related to the input of the adjacent previous token, which greatly reduces the computational complexity.

$$\begin{aligned} \theta_{n+1}^i =& U \odot \sigma\left(S_{n+1}^{slope}\right) + \\ & \sigma\left(F\right) \odot RMSNorm\left(S_{n+1}^{decay}\right) \end{aligned} \tag{20}$$

$$\Theta_{n+1} = \left(\theta_{n+1}^0, \theta_{n+1}^1, ..., \theta_{n+1}^{C-1}\right) \tag{21}$$

The final output is presented in Eq. (20) and Eq. (21). In these equations, $\theta_{n+1}^i \in \mathbb{R}^{1\times D_c}$ represents the final output of the $(i+1)_{th}$ channel, while $\Theta_{n+1} \in \mathbb{R}^{C\times 1\times D_c}$ denotes the $(i+1)th$ output of the merged multi-channel sum, where $i \in \{0, 1, ..., C-2, C-1\}$.

3. Experiment

The structure of our model, including the hyperparameters, is presented in TABLE 1. Three models were trained, each with a different number of parameters: 1.6B, 3B and 10B.

TABLE 1. THE HYPERPARAMETERS OF THE PROPOSED YAN

Size	Batch Size	Sequence Length	Hidden Size	Channel Number	Layers	Learning Rate
1.6B	1152	4096	2560	10	12	8e-4
3B	1152	4096	2560	10	20	8e-4
10B	1152	4096	3072	12	60	8e-4

3.1. Scaling Curves

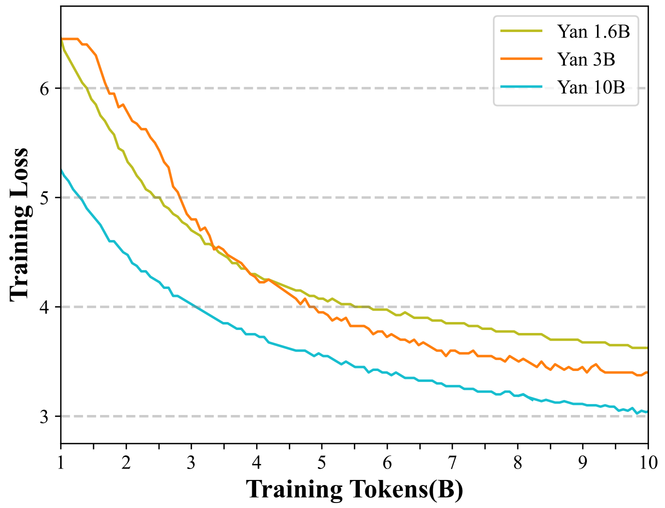

Figure 3. Scaling curves of the proposed Yan. As the quantity of training tokens increases, the loss value decreases in a linear fashion until it reaches a point of convergence. It is evident that the larger the number of model parameters, the smaller the loss value at convergence.

The scaling results, which demonstrate the relationship between the number of training tokens and the training loss, are presented in Figure 3. Three models of Yan were trained, with 1.6B, 3B, and 10B parameters, respectively. A sequence length of 4096 tokens was used. All experiments were performed using the AdamW optimizer [18]. As illustrated in the accompanying figure, in the case of training convergence, the training loss and model size exhibit a proportional relationship, and the outcomes align with Chinchilla's scaling law [19]. All three models demonstrate convergence when the number of training tokens reaches 10B, and the other two models exhibit relatively stable performance throughout the training process, with the exception of the Yan 3B model. This evidence demonstrates that the Yan model can be trained with stability and efficiency, exhibiting good scalability.

3.2. Inference cost

As illustrated in Figure (4-7), a comparison is presented between the GPU memory, latency, and throughput of the Transformer and the proposed Yan during the inference phase. Transformers employ the reuse of the key-value (KV) caches of previously decoded tokens. Yan utilizes a simplified representation, as illustrated in Eq. (20-21). In the conducted inference experiments, the 1.6B model was evaluated on an RTX-3090 24G GPU. The plots demonstrate that Yan outperforms the same parametric number of Transformer models in terms of three inference metrics.

In these figures, sequence length represents the number of output tokens. Batch size represents the number of parallel inference processes. The prompt length is 128 tokens.

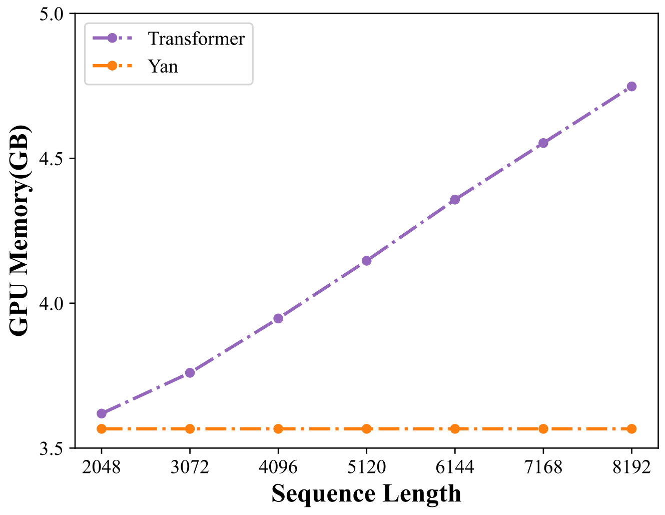

Figure 4. GPU memory versus sequence length curves for Yan and Transformer.

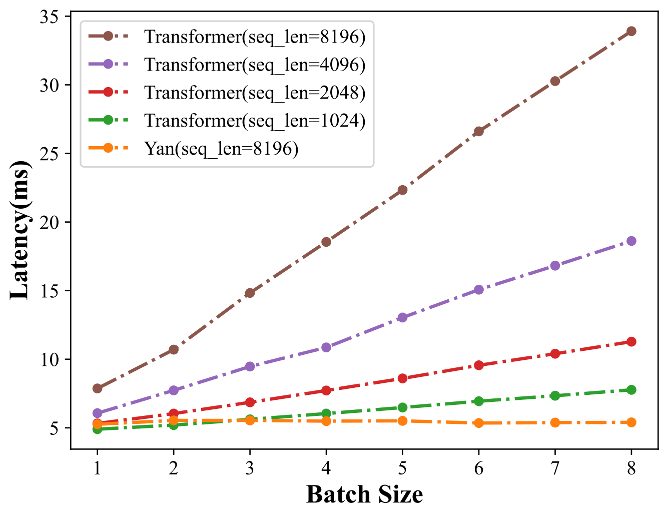

Figure 5. Latency versus batch size curves for Yan and Transformers.

GPU Memory As illustrated in Figure 4, the memory cost of the Transformer increases linearly during inference due to the presence of the KV cache. In contrast, the memory consumption of the proposed Yan remains almost constant

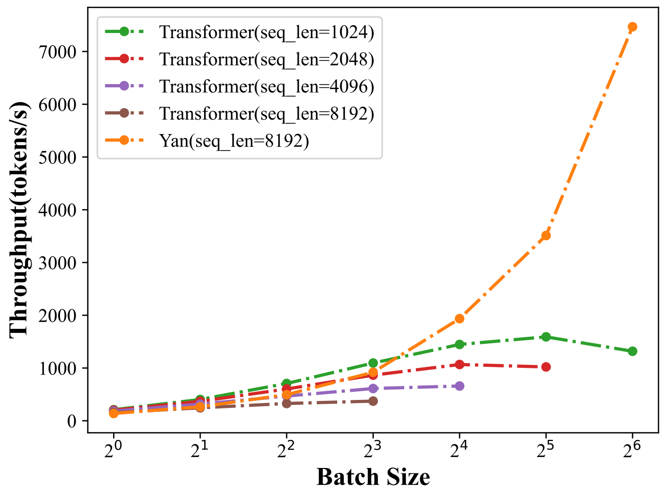

Figure 6. Throughput versus batch size curves for Yan and Transformers.

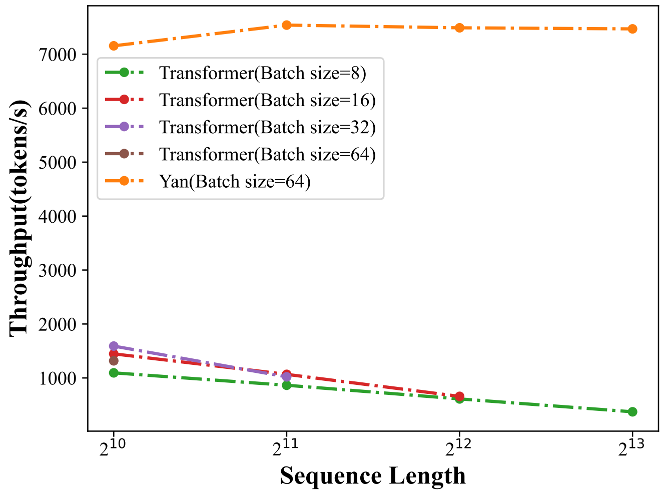

Figure 7. Throughput versus sequence length curves for Yan and Transformers.

even for long sequences, and Yan has a lower memory footprint compared to the Transformer in the sequence length interval equal to 2048 to 8192. This improves the possibility of inference practice on the small memory on the end side.

Latency Latency is a crucial metric in deployment, as it can significantly impact user experience. Figure 5 illustrates the decoding latency. The experimental results demonstrate that increasing the batch size leads to a notable increase in the delay of Transformer, which is approximately linear. Furthermore, the latency increases more rapidly when the sequence length increases. This severely constrains the applicability of Transformer to long sequence output. For a given sequence length, the decoding delay of Yan is considerably less than that of the Transformer, and remains relatively consistent across different batch sizes.

Throughput As illustrated in Figures (6-7), we examine the impact of sequence length and batch size on throughput separately. As shown in Figure 6, we fixed the batch size for each curve, and the throughput of Transformer gradually declines with the increase in sequence length, assuming a constant batch size. In the case of a batch size of at least 16, the presence of the KV cache results in an out-of-memory issue. Conversely, the throughput of the proposed Yan reaches a bottleneck after a slight increase in sequence length, remaining almost unchanged in the absence of out-of-memory issues. Furthermore, under the same batch size, Yan exhibits a higher throughput performance than Transformer.

As illustrated in Figure 7, the sequence length is held constant for each curve. Under the assumption of a constant sequence length, the throughput of Transformer reaches a maximum and then declines with an increase in batch size. This trend is comparable to that observed in Figure 6, where the throughput of Transformer is easily constrained and rapidly exhausts memory resources. With a fixed sequence length, the throughput of Yan increases with the increase of batch size, and there is no out-of-memory phenomenon. Furthermore, the throughput of Yan is higher than that of Transformer with the same sequence length. In fact, the throughput of Yan is several times higher than that of Transformer in the case of the same sequence length.

3.3. Downstream Experimental Comparison

A series of evaluations were conducted on the downstream tasks of baseline pre-trained models, and the evaluated accuracies are shown in TABLE 2, where displays the outcomes of the 5-shot evaluation. The external baseline models we compared were Llama3-8B [14], Mamba-3B [4], RWKV5-3B [2], Gemma-2B [16]. Among these models, Mamba-3B and RWKV5-3B are the most robust small recurrent models reported in the literature to date. Gemma-2B represents the most robust transformer-structured small model. Llama3 is a widely used state-of-the-art open transformer model. The proposed Yan is traind with a parametric count of 1.6B.

In TABLE 2, a comparison was conducted between external baseline models and Yan on MMLU, HellaSwang, ARC-E, and ARC-C, as well as WinoGrande datasets. All models were pre-trained model. The Yan-1.6B model is of the smallest number of parameters comparing to the baseline models, yet it demonstrates a high level of performance. The llama3-8B model is of the largest number of parameters among the models, and it outperforms the other models in all the metrics. In summary, Yan-1.6B and Gemma-2B exhibit comparable performance. Yan-1.6B outperforms the larger models RWKV5-3B and Mamba-3B in all three datasets, which are WinoGrande, MMLU, and HellaSwag. In the ARC-C and ARC-E datasets, the accuracy is slightly higher than that of RWKV5-3B, and slightly lower than that of Mamba-3B. In conclusion, the proposed Yan-1.6B demonstrates comparable performance to those with a larger number of parameters, which indicates that Yan has significant potential for further development.

TABLE 2. COMPARISON OF THE ACCURACY OF YAN AND OTHER MODELS IN DOWNSTREAM EXPERIMENTS WITH 5-SHOT

Model type	Training Tokens	Model size	ARC-C	ARC-E	WinoGrande	MMLU	Hellaswag
Llama3	15T	8B	79.6	92.3	67.1	66.3	69.5
Gemma	2T	2B	40.7	73.2	65.4	38.5	69.5
Mamba	0.3T	3B	39.0	69.0	64.7	26.2	68.6
RWKV-5	1.12T	3B	33.1	67.8	59.6	25.6	59.6
Yan	0.6T	1.6B	37.7	68.4	66.1	35.9	69.2

3.4. Ablation experiment

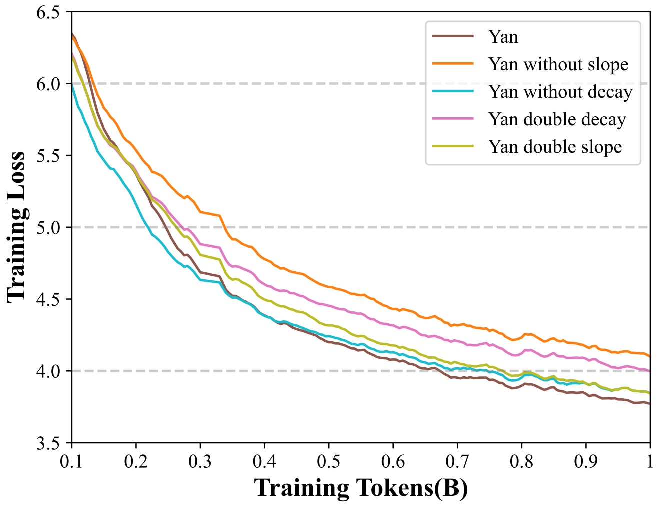

Figure 8. Ablation experiment of the proposed Yan.

The ablation experiment for our proposed Yan is shown in Figure 8. The training loss is plotted as a function of the number of training tokens. Ablation experiments were conducted for both the slope section and the decay section. We established four sets of ablation experiments by either removing the decay section or the slope section, as well as both the sections with decay and slope. The results demonstrate that the loss of any of the four parts boosts the training loss, weakening the model effect to varying degrees. As illustrated in the figure, the loss of the slope section has a more pronounced impact on Yan, while the loss of the decay section has a comparatively smaller effect. Nevertheless, this does not imply that the decay section is superfluous. The rationale is that the majority of the training data employed are short texts, and it is therefore not feasible to assess the impact of the decay section on long texts.

4. Conclusion

This work introduces Yan, a Non-Transformer architecture that extracts the contextual information using Yan block of the slope and decay sections. This approach has been shown to enable very fast inference speed and low inferece cost. In inference experiments, Yan demonstrated better performance than Transformers with the same number of parameters and showed good predictive accuracy. Our experimental results indicate that Yan exhibits excellent language modeling performance at various scales. These advantages enhance the possibilities of Yan for end-side deployment and embodied intelligence.

References

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

[2] 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:2305.13048*, 2023.

[3] Y. Sun, L. Dong, S. Huang, S. Ma, Y. Xia, J. Xue, J. Wang, and F. Wei, “Retentive network: A successor to transformer for large language models,” *arXiv:2307.08621*, 2023.

[4] A. Gu and T. Dao, “Mamba: Linear-time sequence modeling with selective state spaces,” *arXiv:2312.00752*, 2023.

[5] S. Zhai, W. Talbott, N. Srivastava, C. Huang, H. Goh, R. Zhang, and J. Susskind, “An Attention Free Transformer,” *arXiv:2105.14103*, 2021.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[7] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, “GPT-4 technical report,” *arXiv:2303.08774*, 2023.

[8] Gemini Team Google, “Gemini: a family of highly capable multimodal models,” *arXiv:2312.11805*, 2023.

[9] De, Soham, *et al.*, “Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models,” *arXiv:2402.19427*, 2024.

[10] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, “On layer normalization in the transformer architecture,” in *International Conference on Machine Learning*, pp. 10524–10533, PMLR, 2020.

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

[12] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. “Language modeling with gated convolutional networks,” in *International Conference on Machine Learning*, pages 933–941. PMLR, 2017.

[13] D. Hendrycks and K. Gimpel, “Gaussian error linear units (GELUs),” *arXiv:1606.08415*, 2016.

[14] Meta AI. (2024). *Llama 3*. Retrieved from https://llama.meta.com/llama3/ (Accessed: April 19, 2024).

[15] N. Shazeer, *et al.*, “GLU Variants Improve Transformer,” *arXiv:2002.05202*, 2020.

[16] Team G, Mesnard T, Hardin C, *et al.*, “Gemma: Open models based on gemini research and technology,” *arXiv:2403.08295*, 2024.

[17] Ferrada, Hector, *et al.*, “AliBI: An alignment-based index for genomic datasets.” *arXiv:1307.6462*, 2013.

[18] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al., “Training compute-optimal large language models,” *arXiv:2203.15556*, 2022.

[19] I. Loshchilov and F. Hutter, ”Decoupled weight decay regularization,” *arXiv:1711.05101*, 2017

[20] A. Gesmundo and K. Maile, “Composable function-preserving expansions for transformer architectures,” *arXiv:2308.06103*, 2023.

[21] Noam M. Shazeer, “Fast transformer decoding: One write-head is all you need,” *arXiv:1911.02150*, 2019.