Retentive Network

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this work, we propose Retentive Network (RETNET) as a foundation architecture for large language models, simultaneously achieving training parallelism, low-cost inference, and good performance. We theoretically derive the connection between recurrence and attention. Then we propose the retention mechanism for sequence modeling, which supports three computation paradigms, i.e., parallel, recurrent, and chunkwise recurrent. Specifically, the parallel representation allows for training parallelism. The recurrent representation enables low-cost $O(1)$ inference, which improves decoding throughput, latency, and GPU memory without sacrificing performance. The chunkwise recurrent representation facilitates efficient long-sequence modeling with linear complexity, where each chunk is encoded parallelly while recurrently summarizing the chunks. Experimental results on language modeling show that RETNET achieves favorable scaling results, parallel training, low-cost deployment, and efficient inference.

1 Introduction

Transformer [51] has become the de facto architecture for large language models, which was initially proposed to overcome the sequential training issue of recurrent models [25]. However, training parallelism of Transformers is at the cost of inefficient inference, because of the $O(N)$ complexity per step and memory-bound key-value cache [42], which renders Transformers unfriendly to deployment. The growing sequence length increases GPU memory consumption as well as latency and reduces inference speed. Numerous efforts have continued to develop the next-generation architecture, aiming at retaining training parallelism and competitive performance as Transformers while having efficient $O(1)$ inference. It is challenging to achieve the above goals simultaneously.

There have been three main strands of research. First, linearized attention [27, 37] approximates standard attention scores $\exp(\boldsymbol{q} \cdot \boldsymbol{k})$ with kernels $\phi(\boldsymbol{q}) \cdot \phi(\boldsymbol{k})$, so that autoregressive inference can be rewritten in a recurrent form. However, the modeling capability and performance are worse than Transformers, which hinders the method's popularity. The second strand returns to recurrent models for efficient inference while sacrificing training parallelism. As a remedy, element-wise operators [36] are used for acceleration, however, representation capacity and performance are harmed. The third line explores replacing attention with other mechanisms, such as S4 [20], and its variants [11, 38]. None of the previous work can achieve strong performance and efficient inference at the same time compared to Transformers.

In this work, we propose retentive networks (RetNet), achieving low-cost inference, efficient long-sequence modeling, Transformer-comparable performance, and parallel model training simultaneously. Specifically, we introduce a multi-scale retention mechanism to substitute multi-head attention, which has three computation paradigms, i.e., parallel, recurrent, and chunkwise recurrent representations. First, the parallel representation empowers training parallelism to utilize GPU devices fully. Second, the recurrent representation enables efficient $O(1)$ inference in terms of memory and computation. The deployment cost and latency can be significantly reduced. Moreover, the implementation is greatly simplified without key-value cache tricks. Third, the chunkwise recurrent representation can perform efficient long-sequence modeling. We parallelly encode each local block for computation speed while recurrently encoding the global blocks to save GPU memory.

We compare RetNet with Transformer and its variants. Experimental results on language modeling show that RetNet is consistently competitive in terms of both scaling curves and in-context learning. Moreover, the inference cost of RetNet is length-invariant. For a 7B model and 8k sequence length, RetNet decodes 8.4× faster and saves 70% of memory than Transformers with key-value caches. During training, RetNet also achieves 3× acceleration than standard Transformer with highly-optimized FlashAttention-2 [10]. Besides, RetNet's inference latency is insensitive to batch size, allowing enormous throughput. The intriguing properties make RetNet a potential candidate to replace Transformer for large language models.

2 Retentive Network

Retentive network (RetNet) is stacked with L identical blocks, which follows a similar layout (i.e., residual connection, and pre-LayerNorm) as in Transformer [51]. Each RetNet block contains two modules: a multi-scale retention (MSR) module, and a feed-forward network (FFN) module. We introduce the MSR module in the following sections. Given an input sequence $x = x_1 \cdots x_{|x|}$, RetNet encodes the sequence in an autoregressive way. The input vectors $\{\boldsymbol{x}_i\}_{i=1}^{|x|}$ is first packed into $X^0 = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{|x|}] \in \mathbb{R}^{|x| \times d_{\text{model}}}$, where d_{model} is hidden dimension. Then we compute contextualized vector representations $X^l = \text{RetNet}_l(X^{l-1}), l \in [1, L]$.

2.1 Retention

In this section, we introduce the retention mechanism that has a dual form of recurrence and parallelism. So we can train the models in a parallel way while recurrently conducting inference.

Consider a sequence modeling problem that maps $v(n) \mapsto o(n)$ through states $\boldsymbol{s}_n$. Let v_n, o_n denote $v(n), o(n)$ for simplicity. We formulate the mapping in a recurrent manner:

$$\begin{aligned} \boldsymbol{s}_n &= A\boldsymbol{s}_{n-1} + K_n^\intercal v_n, \quad A \in \mathbb{R}^{d \times d}, \quad K_n \in \mathbb{R}^{1 \times d} \\ o_n &= Q_n \boldsymbol{s}_n = \sum_{m=1}^{n} Q_n A^{n-m} K_m^\intercal v_m, \quad Q_n \in \mathbb{R}^{1 \times d} \end{aligned} \tag{1}$$

where we map v_n to the state vector $\boldsymbol{s}_n$, and then implement a linear transform to encode sequence information recurrently. Next, we make the projection Q_n, K_n content-aware:

$$Q = XW_Q, \quad K = XW_K \tag{2}$$

where $W_Q, W_K \in \mathbb{R}^{d \times d}$ are learnable matrices.

We diagonalize the matrix $A = \Lambda(\gamma e^{i\theta})\Lambda^{-1}$, where $\gamma, \theta \in \mathbb{R}^d$. Then we obtain $A^{n-m} = \Lambda(\gamma e^{i\theta})^{n-m}\Lambda^{-1}$. By absorbing Λ into W_Q and W_K, we can rewrite Equation (1) as:

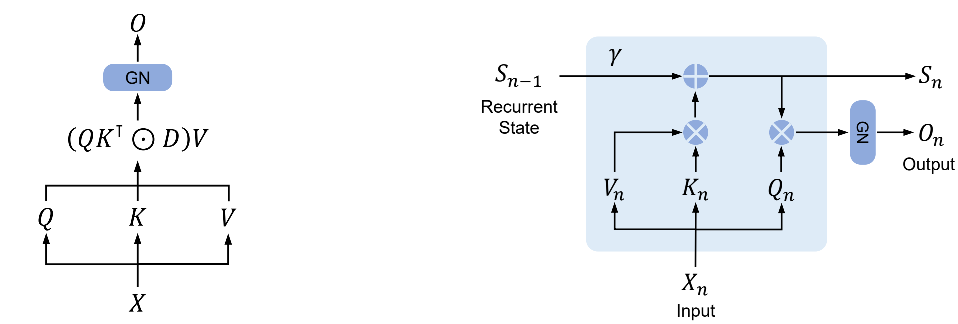
$$\begin{aligned} o_n &= \sum_{m=1}^{n} Q_n (\gamma e^{i\theta})^{n-m} K_m^\intercal v_m \\ &= \sum_{m=1}^{n} (Q_n (\gamma e^{i\theta})^n)(K_m (\gamma e^{i\theta})^{-m})^\intercal v_m \end{aligned} \tag{3}$$

where $Q_n(\gamma e^{i\theta})^n, K_m(\gamma e^{i\theta})^{-m}$ is known as xPos [45], i.e., a relative position embedding proposed for Transformer. We further simplify γ as a scalar, Equation (3) becomes:

$$o_n = \sum_{m=1}^{n} \gamma^{n-m} (Q_n e^{in\theta})(K_m e^{im\theta})^\dagger v_m \tag{4}$$

where † is the conjugate transpose. The formulation is easily parallelizable within training instances.

In summary, we start with recurrent modeling as shown in Equation (1), and then derive its parallel formulation in Equation (4). We consider the original mapping $v(n) \mapsto o(n)$ as vectors and obtain the retention mechanism as follows.



(a) Parallel representation. (b) Recurrent representation.

Figure 1: RetNet has three equivalent computation paradigms, i.e., parallel, recurrent, and chunkwise recurrent representations. Given the same input, three paradigms obtain the same output. "GN" is short for GroupNorm.

The Parallel Representation of Retention As shown in Figure 1a, the retention layer is defined as:

$$
\begin{aligned}
Q = (XW_Q) \odot \Theta, \quad K = (XW_K) \odot \overline{\Theta}, \quad V = XW_V \\
\Theta_n = e^{in\theta}, \quad D_{nm} = \begin{cases} \gamma^{n-m}, & n \geq m \\ 0, & n < m \end{cases} \\
\mathrm{Retention}(X) = (QK^\intercal \odot D)V
\end{aligned}
\tag{5}
$$

where $D \in \mathbb{R}^{|x| \times |x|}$ combines causal masking and exponential decay along relative distance as one matrix, and $\overline{\Theta}$ is the complex conjugate of Θ. In practice, we map $Q, K \in \mathbb{R}^d \rightarrow \mathbb{C}^{d/2}$, add the complex position embedding Θ, then map them back to $\mathbb{R}^d$, following the implementation trick as in LLaMA [48, 44]. Similar to self-attention, the parallel representation enables us to train the models with GPUs efficiently.

The Recurrent Representation of Retention As shown in Figure 1b, the proposed mechanism can also be written as recurrent neural networks (RNNs), which is favorable for inference. For the n-th timestep, we recurrently obtain the output as:

$$
\begin{aligned}
& S_n = \gamma S_{n-1} + K_n^\intercal V_n \\
& \mathrm{Retention}(X_n) = Q_n S_n, \quad n = 1, \cdots, |x|
\end{aligned}
\tag{6}
$$

where Q, K, V, γ are the same as in Equation (5).

The Chunkwise Recurrent Representation of Retention A hybrid form of parallel representation and recurrent representation is available to accelerate training, especially for long sequences. We divide the input sequences into chunks. Within each chunk, we follow the parallel representation (Equation (5)) to conduct computation. In contrast, cross-chunk information is passed following the recurrent representation (Equation (6)). Specifically, let B denote the chunk length. We compute the retention output of the i-th chunk via:

$$
\begin{aligned}
Q_{[i]} = Q_{Bi:B(i+1)}, \quad K_{[i]} = K_{Bi:B(i+1)}, \quad V_{[i]} = V_{Bi:B(i+1)} \\
R_i = K_{[i]}^\intercal (V_{[i]} \odot \zeta) + \gamma^B R_{i-1}, \quad \zeta_{ij} = \gamma^{B-i-1} \\
\mathrm{Retention}(X_{[i]}) = \underbrace{(Q_{[i]} K_{[i]}^\intercal \odot D) V_{[i]}}_{\text{Inner-Chunk}} + \underbrace{(Q_{[i]} R_{i-1}) \odot \xi}_{\text{Cross-Chunk}}, \quad \xi_{ij} = \gamma^{i+1}
\end{aligned}
\tag{7}
$$

where $[i]$ indicates the i-th chunk, i.e., $x_{[i]} = [x_{(i-1)B+1}, \cdots, x_{iB}]$. The proof of the equivalence between recurrent representation and chunkwise recurrent representation is described in Appendix B.

2.2 Gated Multi-Scale Retention

We use $h = d_{\text{model}}/d$ retention heads in each layer, where d is the head dimension. The heads use different parameter matrices $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$. Moreover, **m**ulti-**s**cale **r**etention (MSR) assigns

```python
def ParallelRetention(
    q, k, v, # bsz * num_head * len * qkv_dim
    decay_mask): # num_head * len * len
    retention = q @ k.transpose(-1, -2)
    retention = retention * decay_mask
    output = retention @ v
    output = group_norm(output)
    return output
```

```python
def RecurrentRetention(
    q, k, v, # bsz * num_head * qkv_dim
    past_kv, # bsz * num_head * qk_dim * v_dim
    decay): # num_head * 1 * 1
    current_kv = decay * past_kv + k.unsqueeze(-1) * v.
        unsqueeze(-2)
    output = torch.sum(q.unsqueeze(-1) * current_kv,
        dim=-2)
    output = group_norm(output)
    return output, current_kv
```

```python
def ChunkwiseRetention(
    q, k, v, # bsz * num_head * chunk_size *
        qkv_dim
    past_kv, # bsz * num_head * qk_dim *
        v_dim
    decay_mask, # num_head * chunk_size *
        chunk_size
    chunk_decay, # num_head * 1 * 1
    inner_decay): # num_head * chunk_size
    retention = q @ k.transpose(-1, -2)
    retention = retention * decay_mask
    inner_retention = retention @ v
    cross_retention = (q @ past_kv) *
        inner_decay
    retention = inner_retention +
        cross_retention
    output = group_norm(retention)
    current_kv = chunk_decay * past_kv + k.
        transpose(-1, -2) @ v
    return output, current_kv
```

Figure 2: Pseudocode for the three computation paradigms of retention. Parallel implementation enables training parallelism to fully utilize GPUs. Recurrent paradigm enables low-cost inference. Chunkwise retention combines the above advantages (i.e., parallel within each chunk and recurrent across chunks), which has linear memory complexity for long sequences.

different γ for each head. For simplicity, we set γ identical among different layers and keep them fixed. In addition, we add a swish gate [23, 40] to increase the non-linearity of retention layers. Formally, given input X, we define the layer as:

$$
\begin{aligned}
\gamma &= 1 - 2^{-5-\text{arange}(0,h)} \in \mathbb{R}^h \\
\text{head}_i &= \text{Retention}(X, \gamma_i) \\
Y &= \text{GroupNorm}_h(\text{Concat}(\text{head}_1, \cdots, \text{head}_h)) \\
\text{MSR}(X) &= (\text{swish}(XW_G) \odot Y)W_O
\end{aligned}
\tag{8}
$$

where $W_G, W_O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ are learnable parameters, and GroupNorm [53] normalizes the output of each head, following SubLN proposed in [43]. Notice that the heads use multiple γ scales, which results in different variance statistics. So we normalize the head outputs separately.

The pseudocode of retention is summarized in Figure 2.

Retention Score Normalization We utilize the scale-invariant nature of GroupNorm to improve the numerical precision of retention layers. Specifically, multiplying a scalar value within GroupNorm does not affect outputs and backward gradients, i.e., $\text{GroupNorm}(\alpha * \text{head}_i) = \text{GroupNorm}(\text{head}_i)$. We implement three normalization factors in Equation (5). First, we normalize $QK^\intercal$ as $QK^\intercal/\sqrt{d}$. Second, we replace D with $\tilde{D}_{nm} = D_{nm}/\sqrt{\sum_{i=1}^{n} D_{ni}}$. Third, let R denote the retention scores $R = QK^\intercal \odot D$, we normalize it as $\tilde{R}_{nm} = R_{nm}/\max(\sum_{i=1}^{n}|R_{ni}|,1)$. Then the retention output becomes $\text{Retention}(X) = \tilde{R}V$. The above tricks do not affect the final results while stabilizing the numerical flow of both forward and backward passes, because of the scale-invariant property.

2.3 Overall Architecture of Retention Networks

For an L-layer retention network, we stack multi-scale retention (MSR) and feed-forward network (FFN) to build the model. Formally, the input sequence $\{x_i\}_{i=1}^{|x|}$ is transformed into vectors by a word embedding layer. We use the packed embeddings $X^0 = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{|x|}] \in \mathbb{R}^{|x| \times d_{\text{model}}}$ as the input and compute the model output X^L:

$$
\begin{aligned}
Y^l &= \text{MSR}(\text{LN}(X^l)) + X^l \\
X^{l+1} &= \text{FFN}(\text{LN}(Y^l)) + Y^l
\end{aligned}
\tag{9}
$$

where $\text{LN}(\cdot)$ is LayerNorm [3]. The FFN part is computed as $\text{FFN}(X) = \text{gelu}(XW_1)W_2$, where W_1, W_2 are parameter matrices.

Training We use the parallel (Equation (5)) and chunkwise recurrent (Equation (7)) representations during the training process. The parallelization within sequences or chunks efficiently utilizes GPUs to accelerate computation. More favorably, chunkwise recurrence is especially useful for long-sequence training, which is efficient in terms of both FLOPs and memory consumption.

Inference The recurrent representation (Equation (6)) is employed during inference, which nicely fits autoregressive decoding. The $O(1)$ complexity reduces memory and inference latency while achieving equivalent results.

3 Experiments

We perform language modeling experiments to evaluate RetNet. First, we present the scaling curves of Transformer and RetNet. Second, we follow the training settings of StableLM-4E1T [50] to compare with open-source Transformer models in downstream benchmarks. Moreover, for training and inference, we compare speed, memory consumption, and latency. The training corpus is a curated compilation of The Pile [16], C4 [14], and The Stack [29].

3.1 Comparison with Transformer Variants

We compare RetNet with various efficient Transformer variants, including RWKV [36], H3 [11], Hyena [38], and Mamba [19]. We use LLaMA [48] architecture, including RMSNorm [59] and SwiGLU [40, 7] module, as the Transformer backbone, which shows better performance and stability. Consequently, other variants follow these settings. Specifically, Mamba does not have FFN layers so we only implement RMSNorm. For RetNet, the FFN intermediate dimension is $\frac{5}{3}d$ and the value dimensions in W_G, W_V, W_O are also $\frac{5}{3}d$, where the overall parameters are still $12d^2$. All models have 400M parameters with 24 layers and a hidden dimension of 1024. For H3, we set the head dimension to 8. For RWKV, we use the TimeMix module to substitute self-attention layers while keeping FFN layers consistent with other models for fair comparisons. We train the models with 40k steps with a batch size of 0.25M tokens.

Fine-Grained Language Modeling Evaluation As shown in Table 1, we first report the language modeling perplexity of validation sets. Besides the overall validation set, following [2], we divide perplexity into "AR-Hit" and "First Occur". Specifically, AR-Hit contains the predicted tokens that are previously seen bigrams in the previous context, which evaluates the associative recall ability. "First Occur" has the predicted tokens that can not be recalled from the context. Among various Transformer variants, RetNet outperforms previous methods on both "AR-Hit" and "First Occur" splits, which is important for real-world use cases.

Knowledge-Intensive Tasks We also evaluate Massive Multitask Language Understanding (MMLU; [24]) answer perplexity to evaluate models on knowledge-intensive tasks. We report the average perplexity of the correct answers, i.e., given input `[Question, "Answer:", Correct Answer]`, we calculate the perplexity of the "`Correct Answer`" part. RetNet achieves competitive results among the architectures.

	Language Modeling			MMLU				
	Valid. Set	**AR-Hit**	**First-Occur**	**STEMs**	**Humanites**	**Social-Sci.**	**Others**	**Avg**
Transformer [51]	3.320	1.118	3.826	0.584	0.229	0.279	0.402	0.356
Transformer Variants								
Hyena [38]	3.545	1.799	3.947	1.125	0.576	0.654	0.819	0.767
RWKV [36]	3.497	1.706	3.910	1.156	0.609	0.617	0.781	0.768
Mamba [19]	3.379	1.322	3.852	0.668	0.288	0.300	0.425	0.403
H3 [11]	3.563	1.722	3.986	1.169	0.532	0.637	0.792	0.752
RetNet	**3.360**	**1.264**	**3.843**	**0.577**	**0.263**	**0.280**	**0.384**	**0.362**

Table 1: Perplexity results on language modeling and MMLU [24] answers. We use the augmented Transformer architecture proposed in LLaMA [48] for reference. For language modeling, we report perplexity on both the overall validation set and fine-grained diagnosis sets [2], i.e., "AR-Hit" evaluates the associative recall capability, and "First-Occur" indicates the regular language modeling performance. Besides, we evaluate the answer perplexity of MMLU subsets.

3.2 Language Modeling Evaluation with Various Model Sizes

We train language models with various sizes (i.e., 1.3B, 2.7B, and 6.7B) from scratch. The training batch size is 4M tokens with 2048 maximal length. We train the models with 25k steps. The detailed hyper-parameters are described in Appendix E. We train the models with 512 AMD MI200 GPUs.

Figure 3 reports perplexity on the validation set for the language models based on Transformer and RetNet. We present the scaling curves with three model sizes, i.e., 1.3B, 2.7B, and 6.7B. RetNet achieves comparable results with Transformers. More importantly, the results indicate that RetNet is favorable in terms of size scaling. In addition to performance, RetNet training is quite stable in our experiments. Experimental results show that RetNet is a strong competitor to Transformer for large language models. Empirically, we find that RetNet starts to outperform Transformer when the model size is larger than 2B.

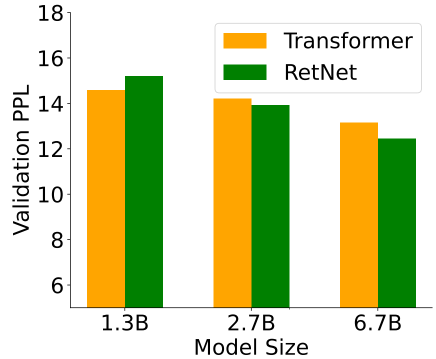


Figure 3: Validation perplexity (PPL) decreases along with scaling up the model size.

3.3 Long-Context Evaluation

We evaluate long-context modeling on the ZeroSCROLLS [41] benchmark. We train a hybrid model of size 2.7B, RetNet+, which stacks the attention and retention layers. Specifically, we insert one attention layer after every 3 retention layers. We follow most configurations of the 2.7B model as in Section 3.2. We scale the number of training tokens to 420B tokens. The batch size is 4M tokens. We first train the model with 4K length and then extend the sequence length to 16K for the last 50B training tokens. The rotation base scaling [55] is used for length extension.

Figure 4 reports the answer perplexity given various lengths of input document. It shows that both Transformer and RetNet+ perform better with longer input documents. The results indicate that the language models successfully utilize the long-distance context. Notice that the 12K and 16K results in Qasper are similar because the lengths of most documents are shorter than 16K. Moreover, RetNet+ obtains competitive results compared with Transformer for long-context modeling. Meanwhile, retention has better training and inference efficiency.

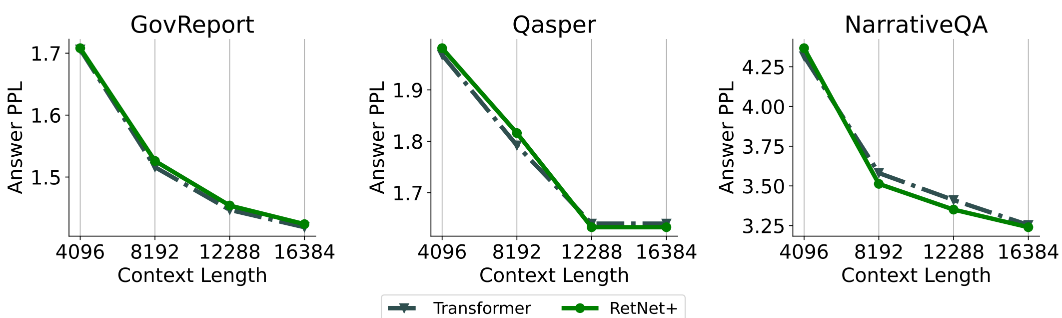
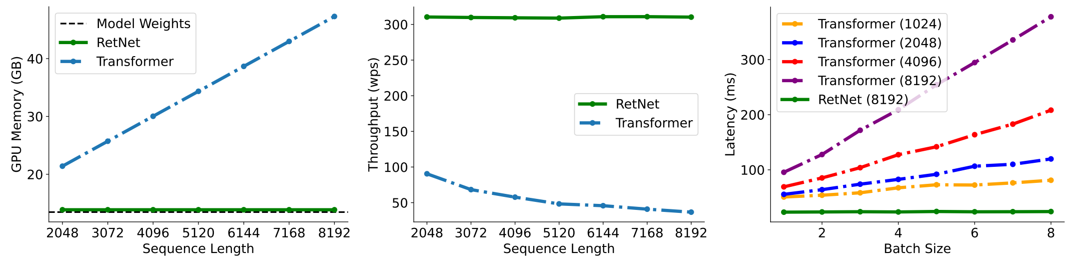


Figure 4: Answer perplexity decreases along with longer input documents. Transformer and RetNet+ obtain comparable performance for long-context modeling on the ZeroSCROLLS [41] benchmark.

3.4 Inference Cost

As shown in Figure 5, we compare memory cost, throughput, and latency of Transformer and RetNet during inference. Transformers reuse KV caches of previously decoded tokens. RetNet uses the recurrent representation as described in Equation (6). We evaluate the 6.7B model on the A100-80GB GPU. Figure 5 shows that RetNet outperforms Transformer in terms of inference cost.

Memory As shown in Figure 5a, the memory cost of Transformer increases linearly due to KV caches. In contrast, the memory consumption of RetNet remains consistent even for long sequences,



(a) GPU memory cost with varying sequence length.

(b) Inference throughput with varying sequence length.

(c) Inference latency with different batch sizes.

Figure 5: Inference cost of Transformer and RetNet with a model size of 6.7B. RetNet outperforms Transformers in terms of memory consumption, throughput, and latency.

requiring much less GPU memory to host RetNet. The additional memory consumption of RetNet is almost negligible (i.e., about 3%) while the model weights occupy 97%.

Throughput As presented in Figure 5b, the throughput of Transformer drops along with the decoding length increases. In comparison, RetNet has higher and length-invariant throughput during decoding, by utilizing the recurrent representation of retention.

Latency Latency is an important metric in deployment that greatly affects the user experience. We report the decoding latency in Figure 5c. Experimental results show that increasing batch size renders the Transformer's latency larger. Moreover, the latency of Transformers grows faster with longer input. In order to make latency acceptable, we have to restrict the batch size, which harms the overall inference throughput of Transformers. By contrast, RetNet's decoding latency outperforms Transformers and stays almost the same across different batch sizes and input lengths.

3.5 Training Throughput

Figure 6 compares the training throughput of Transformer and RetNet, where the training sequence lengths range from 8192 to 65536. The model size is 3.5B, where the hidden dimension is 3072 and the layer size is 28. We use highly optimized FlashAttention-2 [10] for Transformers. In comparison, we implement chunk recurrent representation (Equation (7)) using Triton [46], where the computation is both memory-friendly and computationally efficient. The chunk size is set to 256. We evaluate the results with eight Nvidia H100-80GB GPUs because FlashAttention-2 is highly optimized for H100 cards.

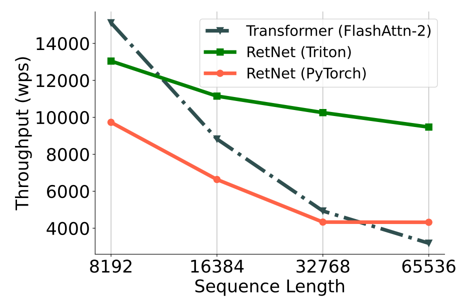


Figure 6: Training throughput (word per second; wps) of Transformer with FlashAttention-2 [10] and RetNet.

Experimental results show that RetNet has higher training throughput than Transformers. The acceleration ratio increases as the sequence length is longer. When the training length is 64k, RetNet's throughput is about 3 times than Transformer's.

3.6 Zero-Shot and Few-Shot Evaluation on Downstream Tasks

We also compare the language models on a wide range of downstream tasks. We evaluate zero-shot and 4-shot learning with the 6.7B models. As shown in Table 2, the datasets include HellaSwag (HS; [57]), BoolQ [8], COPA [52], PIQA [6], Winograd, Winogrande [30], and StoryCloze (SC; [34]). The accuracy numbers are consistent with language modeling perplexity presented in Figure 3. RetNet achieves comparable performance with Transformer on zero-shot and in-context learning settings.

3.7 Ablation Studies

We ablate various design choices of RetNet and report the language modeling results in Table 3. The evaluation settings and metrics are the same as in Section 3.1.

	HS	BoolQ	COPA	PIQA	Winograd	Winogrande	SC	Avg
Zero-Shot Performance								
Transformer	55.9	62.0	69.0	74.6	69.5	56.5	75.0	66.07
RetNet	**60.7**	**62.2**	**77.0**	**75.4**	**77.2**	**58.1**	**76.0**	**69.51**
Few-shot Performance (4-Shot)								
Transformer	55.8	58.7	71.0	75.0	71.9	57.3	75.4	66.44
RetNet	**60.5**	**60.1**	**78.0**	**76.0**	**77.9**	**59.9**	**75.9**	**69.76**

Table 2: Zero-shot and few-shot learning performance. The language model size is 6.7B.

Architecture We ablate the swish gate and GroupNorm as described in Equation (8). Table 3 shows that the above two components improve performance. First, the gating module is essential for enhancing non-linearity and improving model capability. Notice that we use the same parameter allocation as in Transformers after removing the gate. Second, group normalization in retention balances the variances of multi-head outputs, which improves training stability and language modeling results.

Multi-Scale Decay Equation (8) shows that we use different γ as the decay rates for the retention heads. In the ablation studies, we examine removing γ decay (i.e., "$-$ γ decay") and applying the same decay rate across heads (i.e., "$-$ multi-scale decay"). Specifically, ablating γ decay is equivalent to $\gamma = 1$. In the second setting, we set $\gamma = 1 - 2^{-6.5}$ for all heads. Table 3 indicates that both the decay mechanism and using multiple decay rates can improve the language modeling performance.

Head Dimension As indicated by the recurrent perspective of Equation (1), the head dimension implies the memory capacity of hidden states. In ablation, we reduce the default head dimension from 256 to 64, i.e., 64 for queries and keys, and $\lfloor \frac{5}{3} \times 64 \rfloor \approx 108$ for values. We keep the hidden dimension d_{model} the same. Accordingly, we adjust the multi-scale decay as $\gamma = 1 - 2^{-5-arange(0,h)/4}$ to keep the same decay range. Table 3 shows that the larger head dimension achieves better performance.

	Language Modeling			MMLU				
	Valid. Set	AR-Hit	First-Occur	STEMs	Humanites	Social-Sci.	Others	Avg
RetNet	**3.360**	**1.264**	**3.843**	**0.577**	**0.263**	**0.280**	**0.384**	**0.362**
$-$ swish gate	3.509	1.366	4.002	0.599	0.285	0.315	0.421	0.390
$-$ GroupNorm	3.367	1.302	3.843	0.630	0.295	0.327	0.438	0.406
$-$ γ decay	3.920	2.122	4.334	0.958	0.566	0.571	0.694	0.681
$-$ multi-scale decay	3.524	1.768	3.928	0.921	0.433	0.471	0.590	0.582
Reduce head dim.	3.397	1.331	3.872	0.637	0.272	0.294	0.393	0.384

Table 3: Perplexity results on language modeling and MMLU [24] answers. For language modeling, we report perplexity on both the overall validation set and fine-grained diagnosis sets [2], i.e., "AR-Hit" evaluates the associative recall capability, and "First-Occur" indicates the regular language modeling performance. Besides, we evaluate the answer perplexity of the MMLU subsets.

3.8 Results on Vision Tasks

We also compare RetNet with vision Transformers [15, 47] in Table 4, where bidirectional encoders are evaluated. Unlike causal language models, the vision encoders do not require recurrent representations. Specifically, we use retention as follows:

$$Q = (XW_Q) \odot \Theta, \quad K = (XW_K) \odot \overline{\Theta}, \quad V = XW_V$$
$$\text{Retention}(X) = (QK^\intercal)V = Q(K^\intercal V)$$

where multi-scale decay is removed in bidirectional computation. Notice that we can compute retention in different orders. Similar to linear attention [27], the $Q(K^\intercal V)$ paradigm is an efficient operator in bidirectional settings, especially for high-resolution images.

We perform experiments on ImageNet-1K classification [13], COCO object detection [32], and ADE20K semantic segmentation [60]. We compare RetNet with DeiT [47] which is a well-tuned vision Transformer. Besides, we follow [21] and plug in a depth-wise convolution in experiments. We adopt the DeiT-M size, which has about 38M parameters. For ImageNet-1K image classification,

	ImageNet	COCO			ADE20K	
	Acc	AP^b	AP^b_{50}	AP^b_{75}	mIoU	mAcc
DeiT [47]	80.76	0.458	0.678	0.502	43.52	55.08
RetNet	81.57	0.457	0.669	0.488	44.13	56.12

Table 4: Results on vision tasks, i.e., image classification (ImageNet), object detection (COCO), and semantic segmentation (ADE20K). RetNet achieves competitive performance with DeiT, which is a well-tuned vision Transformer.

we use AdamW [33] for 300 epochs, and 20 epochs of linear warm-up. The learning rate is 1×10^{-3}, the batch size is 1024, and the weight decay is 0.05. For COCO object detection, we use Mask R-CNN [22] as the task head, and the above models pre-trained on ImageNet as the backbone with 3x schedules. In ADE20K experiments, we use UperNet [54] as the segmentation head. The detailed configuration can be found in Appendix H.

Table 4 shows the results across various vision tasks. RetNet is competitive compared with DeiT. For classification and segmentation, RetNet is slightly better than DeiT, where RetNet achieves 0.81% accuracy improvement on ImageNet and 0.61% mIoU improvement on ADE20K. For object detection, the results are comparable.

4 Related Work

Numerous efforts are focused on reducing the quadratic complexity of attention mechanisms. Linear attention [27] uses various kernels $\phi(q_i)\phi(k_j)/\sum_{n=1}^{|x|}\phi(q_i)\phi(k_n)$ to replace the $\mathrm{softmax}$ function. In contrast, we reexamine sequence modeling from scratch, rather than aiming at approximating $\mathrm{softmax}$. AFT [58] simplifies dot-product attention to element-wise and moves $\mathrm{softmax}$ to key vectors. RWKV [36] replaces AFT's position embeddings with exponential decay and runs the models recurrently for training and inference. In comparison, retention preserves high-dimensional states to encode sequence information, which contributes to expressive ability and better performance. S4 [20] unifies convolution and recurrence format and achieves $O(N \log N)$ training complexity leveraging the FFT kernel. Unlike Equation (2), if Q_n and K_n are content-unaware, the formulation can be degenerated to S4 [20]. Hyena [38] generates the convolution kernels, achieving sub-quadratic training efficiency but keeping $O(N)$ complexity in single-step inference. Recently, most related work has focused on modifying γ in Equation (6) as a data-dependent variable, such as Mamba [19], GLA [56], Gateloop [28], and xLSTM [4]. Another strand explores hybrid architectures [31, 12] that interleave the above components with attention layers.

In addition, we discuss the training and inference efficiency of some related methods. Let D denote the hidden dimension, H the head dimension, and N the sequence length. For training, RWKV's token-mixing complexity is $O(DN)$, and Mamba's complexity is $O(DHN)$ with optimized CUDA kernels. Hyena's is $O(DN \log N)$ with Fast Fourier Transform acceleration. In comparison, the chunk-wise recurrent representation is $O(DN(B + H))$, where B is the chunk size, and we usually set $H = 256, B \leq 512$. However, chunk-wise computation is highly parallelized, enabling efficient hardware usage. For large model size (i.e., larger D) or sequence length, the additional $b + h$ has negligible effects. For inference, among the efficient architectures compared, Hyena has the same complexity (i.e., $O(N)$ per step) as Transformer, while the others can perform $O(1)$ decoding.

5 Conclusion

We propose retentive networks (RetNet) for sequence modeling, which enables various representations, i.e., parallel, recurrent, and chunkwise recurrent. RetNet achieves significantly better inference efficiency (in terms of memory, speed, and latency), favorable training parallelization, and competitive performance compared with Transformers. The above advantages make RetNet an ideal successor to Transformers for large language models, especially considering the deployment benefits brought by the $O(1)$ inference complexity. In the future, we are interested in deploying RetNet on various edge devices, such as mobile phones.

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] S. Arora, S. Eyuboglu, A. Timalsina, I. Johnson, M. Poli, J. Zou, A. Rudra, and C. Ré. Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927*, 2023.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. xLSTM: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.

[5] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics.

[6] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[7] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. B. Rao, P. Barnes, Y. Tay, N. M. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. C. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. García, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. O. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Díaz, O. Firat, M. Catasta, J. Wei, K. S. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. PaLM: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311, 2022.

[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. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2924–2936, 2019.

[9] T. Computer. Redpajama-data: An open source recipe to reproduce llama training dataset, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

[10] T. Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[11] T. Dao, D. Y. Fu, K. K. Saab, A. W. Thomas, A. Rudra, and C. Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

[12] S. De, S. L. Smith, A. Fernando, A. Botev, G. Cristian-Muraru, A. Gu, R. Haroun, L. Berrada, Y. Chen, S. Srinivasan, G. Desjardins, A. Doucet, D. Budden, Y. W. Teh, R. Pascanu, N. D. Freitas, and C. Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models. 2024.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] J. Dodge, A. Marasović, G. Ilharco, D. Groeneveld, M. Mitchell, and M. Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[16] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[17] L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

[18] X. Geng and H. Liu. Openllama: An open reproduction of llama, May 2023. URL `https://github.com/openlm-research/open_llama`.

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

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

[21] 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.

[22] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[23] D. Hendrycks and K. Gimpel. Gaussian error linear units (GELUs). *arXiv: Learning*, 2016.

[24] 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.

[25] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, Nov. 1997.

[26] W. Hua, Z. Dai, H. Liu, and Q. Le. Transformer quality in linear time. In *International Conference on Machine Learning*, pages 9099–9117. PMLR, 2022.

[27] 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.

[28] T. Katsch. Gateloop: Fully data-controlled linear recurrence for sequence modeling. *arXiv preprint arXiv:2311.01927*, 2023.

[29] D. Kocetkov, R. Li, L. Ben Allal, J. Li, C. Mou, C. Muñoz Ferrandis, Y. Jernite, M. Mitchell, S. Hughes, T. Wolf, D. Bahdanau, L. von Werra, and H. de Vries. The Stack: 3TB of permissively licensed source code. *Preprint*, 2022.

[30] H. Levesque, E. Davis, and L. Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2012.

[31] 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.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[33] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[34] N. Mostafazadeh, M. Roth, A. Louis, N. Chambers, and J. Allen. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, 2017.

[35] A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De. Resurrecting recurrent neural networks for long sequences. *ArXiv*, abs/2303.06349, 2023.

[36] B. Peng, E. Alcaide, Q. G. Anthony, A. Albalak, S. Arcadinho, H. Cao, X. Cheng, M. Chung, M. Grella, G. Kranthikiran, X. He, H. Hou, et al. RWKV: Reinventing RNNs for the Transformer era. *ArXiv*, abs/2305.13048, 2023.

[37] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. A. Smith, and L. Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.

[38] M. Poli, S. Massaroli, E. Nguyen, D. Y. Fu, T. Dao, S. Baccus, Y. Bengio, S. Ermon, and C. Ré. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*, 2023.

[39] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264.

[40] P. Ramachandran, B. Zoph, and Q. V. Le. Swish: a self-gated activation function. *arXiv: Neural and Evolutionary Computing*, 2017.

[41] U. Shaham, M. Ivgi, A. Efrat, J. Berant, and O. Levy. ZeroSCROLLS: A zero-shot benchmark for long text understanding. *arXiv preprint arXiv:2305.14196*, 2023.

[42] N. M. Shazeer. Fast Transformer decoding: One write-head is all you need. *ArXiv*, abs/1911.02150, 2019.

[43] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[44] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[45] Y. Sun, L. Dong, B. Patra, S. Ma, S. Huang, A. Benhaim, V. Chaudhary, X. Song, and F. Wei. A length-extrapolatable transformer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14590–14604, Toronto, Canada, July 2023. Association for Computational Linguistics.

[46] P. Tillet and D. Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019.

[47] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[48] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[49] 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.

[50] J. Tow, M. Bellagente, D. Mahan, and C. Riquelme. StableLM 3B 4E1T. https://aka.ms/StableLM-3B-4E1T, 2023.

[51] 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 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010, 2017.

[52] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*, 2019.

[53] Y. Wu and K. He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[54] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018.

[55] W. Xiong, J. Liu, I. Molybog, H. Zhang, P. Bhargava, R. Hou, L. Martin, R. Rungta, K. A. Sankararaman, B. Oguz, et al. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*, 2023.

[56] S. Yang, B. Wang, Y. Shen, R. Panda, and Y. Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

[57] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[58] S. Zhai, W. Talbott, N. Srivastava, C. Huang, H. Goh, R. Zhang, and J. Susskind. An attention free transformer. *arXiv preprint arXiv:2105.14103*, 2021.

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

[60] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ADE20k dataset. *International Journal of Computer Vision*, 127:302–321, 2019.

A Scaling Up Number of Training Tokens

We scale up the number of training tokens to 350B for the 3B-size models. We compare with strong Transformer checkpoints including OpenLLaMA [18] and StableLM [50]. Moreover, we reproduce a Transformer language model (named $\text{Transformer}_{\text{Repro}}$) for apple-to-apple comparison.

Our model RetNet+ follows the same configuration as in Section 3.3, which is a hybrid model. The model's hidden size is 3072, and the number of layers is 28. Without vocabulary embedding, the total number of parameters is 3.17B, which is between StableLM-3B-4E1T (2.7B) and OpenLLaMA-3B-v1 (3.19B). The batch size is 4M tokens. The training length is 4k. The learning rate is 3.2×10^{-4} with 1000 warm-up steps and linear learning rate decay. The training corpus includes The Pile [16] and RedPajama [9]. $\text{Transformer}_{\text{Repro}}$ follows the exact same setting.

Table 5 reports accuracy numbers on the Harness-Eval benchmark [17]. We directly follow the evaluation protocol. The results show that RetNet+ achieves a performance comparable to $\text{Transformer}_{\text{Repro}}$ on language tasks. Notice that OpenLLaMA-3B-v1 and StableLM-3B use different learning rate schedules. The results of these two models are used for reference purposes.

Model	ARC-C	ARC-C$_{\text{norm}}$	ARC-E	ARC-E$_{\text{norm}}$	Hellaswag	Hellaswag$_{\text{norm}}$
OpenLLaMA-3B-v1	0.303	0.323	0.641	0.599	0.449	0.608
StableLM-3B	—	—	0.649	0.610	—	—
Transformer$_{\text{Repro}}$	**0.322**	**0.354**	0.668	**0.633**	0.476	0.633
RetNet+	0.321	0.347	**0.675**	0.613	**0.478**	**0.639**

Model	OBQA	OBQA$_{\text{norm}}$	PIQA	PIQA$_{\text{norm}}$	Winogrande	Avg
OpenLLaMA-3B-v1	0.222	0.348	0.713	0.724	0.594	0.502
StableLM-3B	—	—	**0.759**	**0.763**	0.608	—
Transformer$_{\text{Repro}}$	**0.258**	0.358	0.746	0.755	0.612	**0.529**
RetNet+	**0.258**	**0.362**	0.750	**0.763**	**0.614**	**0.529**

Table 5: Accuracy on the Harness-Eval benchmark. All models are trained with 350B tokens with a batch size of 4M tokens. The results of OpenLLaMA-3B-v1 are taken from their official repository (`https://bit.ly/openllama-350b-results`), and StableLM-3B from their technical report (`https://bit.ly/StableLM-3B-4E1T`).

B Equivalence Between Chunk-wise Recurrent Representation and Recurrent Representation

We illustrate the equivalence between the recurrent representation and the chunk-wise recurrent representation. Specifically, let B denote the chunk length. For the output O_n, n can be divided as $n = kB + r$ where B is the chunk size. Following Equation 6, we have:

$$
\begin{aligned}
O_n &= \sum_{m=1}^{n} \gamma^{n-m} Q_n K_m^{\intercal} V_m \\
&= (Q_n K_{kB+1:n}^{\intercal} \odot \Gamma) V_{kB+1:n} + (Q_n \gamma^r) \sum_{c=0}^{k-1} \sum_{m=1}^{B} (K_{m+cB}^{\intercal} V_{m+cB} \gamma^{B-m}) \gamma^{(k-1-c)B} \\
&= (Q_n K_{kB+1:n}^{\intercal} \odot \Gamma) V_{kB+1:n} + (Q_n \gamma^r) \sum_{c=1}^{k} (K_{[c]}^{\intercal} (V_{[c]} \odot \zeta)) \gamma^{(k-c)B} \\
&= (Q_n K_{kB+1:n}^{\intercal} \odot \Gamma) V_{kB+1:n} + (Q_n \gamma^r) R_{i-1}
\end{aligned}
\tag{10}
$$

where $\Gamma_i = \gamma^{n-i}$, $\zeta_{ij} = \gamma^{B-m}$, and $[i]$ indicates the i-th chunk, i.e., $x_{[i]} = [x_{(i-1)B+1}, \cdots, x_{iB}]$. Then we write R_n as a recurrent function and compute the retention output of the i-th chunk via:

$$
\begin{aligned}
R_i &= K_{[i]}^{\intercal} (V_{[i]} \odot \zeta) + \gamma^B R_{i-1} \\
\zeta_{ij} &= \gamma^{B-i}, \quad \xi_{ij} = \gamma^i \\
\text{Retention}(X_{[i]}) &= \underbrace{(Q_{[i]} K_{[i]}^{\intercal} \odot D) V_{[i]}}_{\text{Inner-Chunk}} + \underbrace{(Q_{[i]} \odot \xi) R_{i-1}}_{\text{Cross-Chunk}}
\end{aligned}
\tag{11}
$$

Finally, we show that the chunkwise recurrent representation is equivalent to the other representations.

C Results with Different Context Lengths

As shown in Table 6, we report the results of language modeling with different context lengths. In order to make the numbers comparable, we use 2048 text chunks as evaluation data and only compute the perplexity for the last 128 tokens. Experimental results show that RetNet performs comparably with Transformer in different context lengths.

Model	**512**	**1024**	**2048**
Transformer	13.55	12.56	12.35
RetNet	13.09	12.14	11.98

Table 6: Language modeling perplexity of RetNet and Transformer with different context length. The results show that RetNet has a consistent advantage across sequence length.

D Hyperparameters Used in Section 3.1

We use LLaMA [48] architecture, including RMSNorm [59] and SwiGLU [40, 7] module, as the Transformer backbone, which shows better performance and stability. The weights of word embedding and $\mathrm{softmax}$ projection are shared. Consequently, other variants follow these settings. For RetNet, the FFN intermediate dimension is $\frac{5}{3}d$ and the value dimensions in W_G, W_V, W_O are also $\frac{5}{3}d$, where the overall parameters are still $12d^2$.

For H3, we set the head dimension to 8. For RWKV, we use the TimeMix module to substitute self-attention layers while keeping FFN layers consistent with other models for fair comparisons. For Mamba, we follow all the details in the paper [19], where double-SSM layers are implemented instead of "SSM + SwiGLU". In addition to RetNet and Mamba, the FFN intermediate dimension is all $\frac{8}{3}d$. All models have 400M parameters, 24 layers, and a hidden dimension of 1024. We train the models with 40k steps and a batch size of 0.25M tokens.

Params	**Values**
Layers	24
Hidden size	1024
Vocab size	100,288
Heads	24
Adam β	(0.9, 0.98)
LR	1.5×10^{-4}
Batch size	0.25M
Warmup steps	375
Weight decay	0.05
Dropout	0.0

Table 7: Hyperparamters used for the architecture comparison in Section 3.1.

E Hyperparameters Used in Section 3.2

We re-allocate the parameters in MSR and FFN for fair comparisons. Let d denote d_{model} for simplicity here. In Transformers, there are about $4d^2$ parameters in self-attention where $W_Q, W_K, W_V, W_O \in \mathbb{R}^{d\times d}$, and $8d^2$ parameters in FFN where the intermediate dimension is $4d$. In comparison, RetNet has $8d^2$ parameters in retention, where $W_Q, W_K \in \mathbb{R}^{d\times d}, W_G, W_V \in \mathbb{R}^{d\times 2d}, W_O \in \mathbb{R}^{2d\times d}$. Notice that the head dimension of V is twice Q, K, similar to GAU [26]. The widened dimension is projected back to d by W_O. In order to keep the parameter number the same as Transformer, the FFN intermediate dimension in RetNet is $2d$. Meanwhile, we set the head dimension to 256, i.e., 256 for queries and keys, and 512 for values. For fair comparison, we keep γ identical among different model sizes, where $\gamma = 1 - e^{\text{linspace}(\log 1/32, \log 1/512, h)} \in \mathbb{R}^h$ instead of the default value in Equation (8).

Hyperparameters	1.3B	2.7B	6.7B
Layers	24	32	32
Hidden size	2048	2560	4096
FFN size	4096	5120	8192
Heads	8	10	16
Learning rate	6×10^{-4}	3×10^{-4}	3×10^{-4}
LR scheduler		Linear decay	
Warm-up steps		375	
Tokens per batch		4M	
Adam β		(0.9, 0.98)	
Training steps		25,000	
Gradient clipping		2.0	
Dropout		0.1	
Weight decay		0.05	

Table 8: Hyperparamters used for language modeling in Section 3.2.

F Results on Open-Ended Generation Tasks

Table 9 presents one-shot performance on two open-ended question-answering tasks, including SQUAD [39] and WebQS [5], with 6.7B models as follows. We report the recall metric in the table, i.e., whether the answers are contained in the generated response.

Dataset	SQUAD	WebQS
Transformer	67.7	36.4
RetNet	72.7	40.4

Table 9: Answer recall of RetNet and Transformer on open-ended question answering.

G Inference Cost of Grouped-Query Retention

We compare with grouped-query attention [1] and evaluate the method in the context of RetNet. Grouped-query attention makes a trade-off between performance and efficiency, which has been successfully verified in LLaMA2 34B/70B [49]. The method reduces the overhead of key/value cache during inference. Moreover, the performance of grouped-query attention is better than multi-query attention [42], overcoming the quality degradation brought by using one-head key value.

As shown in Table 10, we compare the inference cost with grouped-query attention and apply the method for RetNet. For the LLaMA2 70B model, the number of key/value heads is reduced by 8$\times$, where the query head number is 64 while the key/value head number is 8. For RetNet-70B, the parameter allocation is identical to LLaMA [48], where the dimension is 8192, and the head number is 32 for RetNet. For RetNet-70B-GQ2, the key-value head number is 16, where grouped-query retention is applied. We run the inference with four A100 GPUs without quantization.

When the batch size is 256, LLaMA2 runs out of memory while RetNet without group query still has a high throughput. When equipped with grouped-query retention, RetNet-70B achieves 38% acceleration and saves 30% memory.

We evaluate LLaMA2 under 2k and 8k lengths separately. The batch size is reduced to 8 so that LLaMA2 can run without out of memory. Table 10 shows that the inference cost of Transformers increases with the sequence length. In contrast, RetNet is length-invariant. Moreover, RetNet-70B-GQ2 achieves better latency, throughput, and GPU memory than LLaMA2-70B-2k/8k equipped with grouped-query attention. Notice that the evaluation metrics are averaged over positions of different sequence lengths for a fair comparison, rather than only considering the inference cost of the maximum length.

Model	Batch Size	Latency (ms)↓	Throughput (wps)↑	Memory (GB)↓
LLaMA2-70B-2k	256	—	—	OOM
LLaMA2-70B-8k	256	—	—	OOM
RetNet-70B	256	639.1	410.19	72.469
RetNet-70B-GQ2	256	461.8	567.66	52.726
LLaMA2-70B-2k	8	184.5	44.42	33.374
LLaMA2-70B-8k	8	277.7	29.50	37.386
RetNet-70B-GQ2	8	106.2	77.02	32.301

Table 10: Inference cost of RetNet and LLaMA2-70B with difference batch size and length. LLaMA2-70B is equipped with grouped-query attention, reducing key/value heads by $8\times$. "-GQ2" means grouped-query retention, which reduces half of key/value heads. "-2k" and "-8k" indicate sequence length for LLaMA2, while RetNet is length-invariant. RetNet is capable of large-batch inference and is favourable in terms of latency, throughput, and GPU memory.

H Hyperparameters Used in Section 3.8

Hyperparameters	DeiT	RetNet
Layers	12	12
Hidden size	512	512
Patch size	16	16
FFN size	2048	1024
Heads	8	2
Learning rate	1×10^{-3}	
LR scheduler	Cosine decay	
Batch size	1024	
Epochs	300	
Warmup epochs	5	
Smoothing	0.1	
Weight decay	0.05	
Drop path	0.3	

Table 11: Hyperparamters used for the ImageNet experiments in Section 3.8.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction is carefully written.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: There is no theoretical result in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The experiment can be easily reproduced based on the model description, hyperparameter, and any well-known pre-training corpus.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code will be released in camera-ready version. All of the data we use is public-available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Hyperparameters are attached in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: For large language models, the variance between different runs is negligible. Moreover, the evaluation pipeline is deterministic.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The corresponding resources are stated in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the NeurIPS Code of Ethics in the research.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We work on fundamental research that has no direct societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not pose safety risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We carefully follow the licenses of open-source code, data, and models.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.