SCALING STICK-BREAKING ATTENTION: AN EFFICIENT IMPLEMENTATION AND IN-DEPTH STUDY

Anonymous authors
Paper under double-blind review

ABSTRACT

The self-attention mechanism traditionally relies on the softmax operator, necessitating positional embeddings like RoPE, or position biases to account for token order. But current methods using still face length generalisation challenges. We investigate an alternative attention mechanism based on the stick-breaking process in larger scale settings. The method works as follows: For each token before the current, we determine a break point, which represents the proportion of the stick to allocate to the current token. This process is repeated on the remaining stick, until all tokens are allocated, resulting in a sequence of attention weights. This process naturally incorporates recency bias, which has linguistic motivations for grammar parsing (Shen et al., 2017). We study the implications of replacing the conventional softmax-based attention mechanism with stick-breaking attention. We then discuss implementation of numerically stable stick-breaking attention and adapt Flash Attention to accommodate this mechanism. When used as a drop-in replacement for current softmax+RoPE attention systems, we find that stick-breaking attention performs competitively with current methods on length generalisation and downstream tasks. Stick-breaking also performs well at length generalisation, allowing a model trained with 2^{11} context window to perform well at 2^{14} with perplexity improvements.

1 INTRODUCTION

The Transformer architecture (Vaswani et al., 2017) uses a self-attention mechanism based on the softmax operator that enables the model to weigh the importance of different tokens in the input data. However, the reliance on softmax requires using positional embeddings to introduce information about the order of tokens, as the attention mechanism itself is permutation-invariant. The sinusoidal position embedding as proposed in Vaswani et al. (2017) has since evolved to relative positional embeddings (Shaw et al., 2018). Learned relative positional biases were used in the T5 model (Raffel et al., 2020), and later fixed relative positional biases in Press et al. (2021). At the time of this writing, a commonly used form of position embedding is RoPE (Su et al., 2021). Allen-Zhu & Li (2023) observe that, in a context-free grammar parsing setting, attention mechanisms attend to the "most adjacent" non-terminal. This suggests an inclination to attend to the most recent entry that matches a given criteria. However, even with relative position information, it is possible to overfit on specific relative positions, resulting in failure to generalise. Kazemnejad et al. (2024) show that decoder-only Transformers with No Positional Embeddings (NoPE) can implicitly recover positional information, experimental results suggest that NoPE Transformers generalise better on length. While this is promising, a higher attention score from an irrelevant token in the sequence can function as a distractor (Kazemnejad et al., 2024; Xiao et al., 2023).

The stick-breaking process may have properties that can alleviate the previously mentioned issues, and possess the 'most recent' bias from Allen-Zhu & Li (2023) that we want. For a token at position j attending to position i, suppose the attention weight is given by:

$$\boldsymbol{A}_{i,j} = \beta_{i,j} \prod_{i<k<j} (1 - \beta_{k,j}) = \sigma(z_{i,j}) \prod_{i<k<j} \left(1 - \sigma\left(z_{k,j}\right)\right),$$

Logits $$z_{i,j} = \frac{\boldsymbol{q}_i^\top \boldsymbol{k}_j}{\sqrt{d_\text{head}}}$$

Softmax $$\boldsymbol{A}_{i,j} = \frac{\exp(z_{i,j})}{\sum_{j' \leq i} \exp(z_{i,j'})}$$

Stick-breaking $$\boldsymbol{A}_{i,j} = \sigma(z_{i,j}) \prod_{i<k<j} (1 - \sigma(z_{k,j}))$$

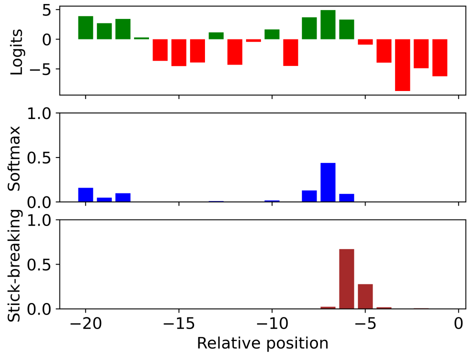


Figure 1: Differences in formulation between stick-breaking and softmax. Stick-breaking assigns high weights to the most recent high logit, while softmax will assign equal weightage to equal logits.

where $z_{k,j}$ are the attention logits. To illustrate via intuition, for $\boldsymbol{A}_{i,j}$ to be high, all $\beta_{k,j}$ for $i < k$ have to be low. Conversely, as long as any $\beta_{k,j}$ for $i < k$ is high, $\boldsymbol{A}_{i,j}$ will be low, as a token between i and j has already been attended to. Shen et al. (2017) makes a similar observation as in Allen-Zhu & Li (2023), and explicitly uses a stick-breaking process to model local structure. Csordás et al. (2021) introduces Geometric attention, named after the Geometric distribution, which only has one parameter p, which gives the probability of success per trial. The geometric distribution then gives the probability for which k trials are needed for the first success: $(1-p)^{k-1}p$. But in stick-breaking and in Geometric attention (Csordás et al., 2021), each p is assigned a different value that corresponds to the attention score between two tokens.

In this paper, we expand upon prior work on this attention mechanism (Csordás et al., 2021; Shen et al., 2023). We focus on the implications of replacing the softmax-based attention mechanism with the stick-breaking process:

1. We compare the different properties of stick-breaking attention against softmax attention,
2. We discuss numerically stable implementations of the stick-breaking attention, and make stick-breaking amenable for large-scale training by implementing a kernel for stick-breaking in Triton,
3. We show the performance of stick-breaking attention on length-generalisation in language modelling, and evaluate 1B and 3B parameter models on various NLP tasks.

2 STICK-BREAKING ATTENTION

For a sequence of L tokens, we have query $\boldsymbol{q}_i \in \mathbb{R}^{d_\text{head}}$, key $\boldsymbol{k}_i \in \mathbb{R}^{d_\text{head}}$, and value $\boldsymbol{v}_i \in \mathbb{R}^{d_\text{head}}$ vectors for $1 \leq i \leq L$. Then the attention weight for token at j attending to position i is computed by:

$$\boldsymbol{o}_j = \sum_{i=1}^{j-1} \boldsymbol{A}_{i,j} \boldsymbol{v}_i, \quad \boldsymbol{A}_{i,j} = \beta_{i,j} \prod_{i<k<j} (1 - \beta_{k,j}), \quad \beta_{i,j} = \sigma(z_{i,j}), \quad z_{i,j} = \frac{\boldsymbol{q}_j^\top \boldsymbol{k}_i}{\sqrt{d_\text{head}}} \tag{1}$$

Equation 1 is the main difference between our proposal and softmax attention. As discussed earlier and in Csordás et al. (2021), this parameterisation biases towards recency. Specifically, for any pair of i and i' such that $i - j < i' - j$ and $z_{i,j} = z_{i',j}$, then $\boldsymbol{A}_{i,j} \geq \boldsymbol{A}_{i',j}$. Consequently, this imposes an ordering on the attention, and we do not use position embeddings with the query and key embeddings.

We consider two sets of logits, $z_{i,j}$ and $z'_{i,j}$ and their respective attention weights $\boldsymbol{A}_{i,j}$ and $\boldsymbol{A}'_{i,j}$. If $z_{i,k} = z'_{i,k}$ for $i < k < j$, then $\boldsymbol{A}_{i,j} = \boldsymbol{A}'_{i,j}$. Further, if $\sum_{k=i}^{j-1} \boldsymbol{A}_{k,j} = 1$, then the output is invariant to appending additional context earlier than i. This means that unlike softmax attention, a high attention score further back in the sequence does not 'distract' from a more recent high $z_{i,j}$ score. We note that $\sum_{i=1}^{j-1} \boldsymbol{A}_{i,j} \leq 1$, which allows this attention mechanism to attend to nothing when all $\beta_{i,j} = 0$. We discuss a strategy to deal with the remaining attention weight in Appendix B.

3 RELATED WORK

Stick-breaking Process The stick-breaking process formulation of the Dirichlet process (Sethuraman, 1994) is also known as the GEM distribution, first coined in Ewens (1990) after Griffiths (1989), Engen (1975), and McCloskey (1965). The GEM is a specific case of what was known as a Residual Allocation Model (RAM; Allen & Lambie 1976). There are instances of the distribution being used as a differentiable attention-like mechanism in neural models. Shen et al. (2017) used stick-breaking process for modelling language, and showed that the model can induce grammatical structure to some extent. Csordás et al. (2021) used stick-breaking attention, which they refer to as Geometric attention, in a bidirectional encoder set up. Shen et al. (2023) used stick-breaking attention in a decoding-only setup, but does not explicitly study the properties of stick-breaking.

Softmax attention, Positional embeddings, and Length Generalisation Bondarenko et al. (2024) observe that softmax attention tends to attend to low-information tokens with high scores in order to ‘do nothing’. Xiao et al. (2023) introduces *attention sinks*, a learnable token that the attention can assign attention weights to. Irie et al. (2019) and Haviv et al. (2022) find that in a decoder-only setting, a Transformer with no positional embedding can work fairly well. Kazemnejad et al. (2024) also found similar results, while also showing that NoPE has a tendency to attend to the start of the sequence, while ALiBi (Press et al., 2021) has a tendency to only attend to the most recent tokens. However, Zhou et al. (2024) later found that Transformers without position embeddings do not generalise to out-of-distribution sequence lengths for an addition task. At present, Rotary Positional Embeddings (RoPE; Su et al. 2021) are the most commonly used position embedding. It encodes relative positions via multiplicative interactions with the key and query. RoPE has been found to generalise to out-of-distribution lengths poorly (Press et al., 2021; Zhou et al., 2024; Kazemnejad et al., 2024), but a common trick to extend the context window RoPE-based Transfomers is to use NTK-aware RoPE scaling (bloc97, 2023).

Conditional Computation The use of stick-breaking for conditional computation has also been explored. Tan & Sim (2016) uses a the stick-breaking distribution as a mixture over outputs for each layer in an MLP for an acoustic model. Graves (2016) also suggested a similar formulation for language modelling. Later, Banino et al. (2021) and Tan et al. (2023) also use a stick-breaking formulation for dynamic depth modelling in a Transformer model. These prior works use conditional computation on the *depth* of the model, while in our case, we use stick-breaking as a method of restricting the computation *length*-wise.

Connection to Selective State-space Models Each stick-breaking attention head at every time-step can be viewed as the hidden state of the final step of a selective State-space Model (SSM;Gu & Dao 2023). For a given time-step j, consider the following SSM and its convolutional form (as described in Merrill et al. 2024):

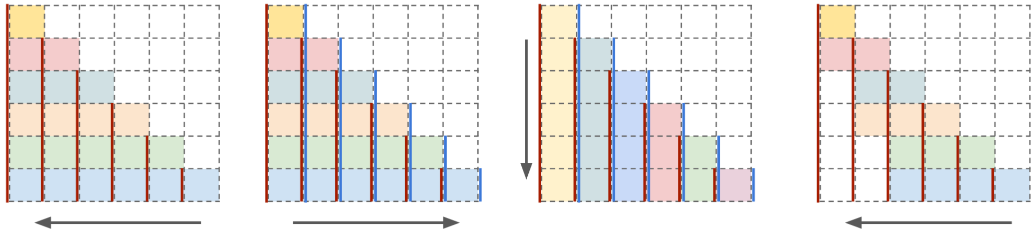
$$\hat{\boldsymbol{o}}_{i,j} = (1-\beta_{i,j})\cdot\hat{\boldsymbol{o}}_{i-1} + \beta_{i,j}\cdot\boldsymbol{v}_j, \qquad \boldsymbol{o}_j = \hat{\boldsymbol{o}}_{j-1,j} = \sum_{1\leq i\leq j}\left(\beta_i \prod_{i<k<j}(1-\beta_k)\right)\cdot\boldsymbol{v}_i, \tag{2}$$

which is equivalent to the first term in Equation 1. Typically, an attention layer for a Transformer with hidden dimension d_{hidden} has h heads such that $d_{\text{hidden}} = h\cdot d_{\text{head}}$. For equivalence with an SSM, we need a constant query vector and each dimension as a separate head, *e.g.* $\boldsymbol{q}_i = \mathbf{1}$ for all i, and $h = d_{\text{hidden}}, d_{\text{head}} = 1$.

Connection to Additive Relative Position Encoding (Additive RPE; Kazemnejad et al. 2024) Generally, Additive RPEs incorporate an added bias function g of the distance of the tokens $i-j$ and the maximum length of the sequence L:

$$\boldsymbol{A}_{ij} \propto \exp\left(\boldsymbol{q}_j^{\top}\boldsymbol{k}_i + b\right)$$

In the case of ALiBi (Press et al., 2021), this is a linear function $b = -m\cdot(j-i)$. This implies that the attention weights will drop off exponentially the further j and i are apart, regardless of the attention scores. In stick-breaking, Equation 12 has a form that accounts for the scores from j to i with the bias $b = -\sum_{k=i+1}^{j}\log\left(1+\exp(z_{k,j})\right)$. Specifically, if $\log\left(1+\exp(z_{k,j})\right)\geq m$, then $b\leq -m\cdot(j-i)$. This suggests a learnable relative position bias that is dependent on the intermediate scores between j and i.



(a) Forward pass (b) 2-phase backward pass (c) Block Skipping

Figure 2: Thread tile assignments for a given attention head and a sequence. Tiles coloured the same are processed by the same thread. For stick-breaking, the forward pass has to be computed from right-to-left, while the backward pass is computed in two-phases: left-to-right for $\nabla \boldsymbol{Q}$, and top-to-bottom for $\nabla \boldsymbol{V}, \nabla \boldsymbol{K}$. Uncoloured tiles are not computed: upper right tiles are not used in causal language modelling, and in the case of block skipping, some blocks can be skipped if all entries have summed to 1. The accumulated values $\boldsymbol{a}, \boldsymbol{b}$ are highlighted at the borders for a visual representation of where they are used.

4 IMPLEMENTATION

Implementing stick-breaking attention naively in PyTorch results in realising the L^2 matrix for the attention logits (where L is the length of the input). FlashAttention (Dao et al., 2022) reduces the memory footprint of attention by side-stepping the $O(L^2)$ memory complexity of realising the attention matrix. In order to achieve this, it only realises tiles of the attention logits and weights at a time, and accumulates the resulting weighted sum of $\boldsymbol{v}_i$. In this section we detail the important differences between computing softmax attention and stick-breaking.

Forward Computing Equation 1 directly will result in underflow issues, especially with lower precision training. We perform the operations in log-space, which results in a cumulative sum instead:

$$\boldsymbol{A}_{i,j} = \exp\left(\log \beta_{i,j} + \sum_{k=i+1}^{j-1} \log\left(1 - \beta_{k,j}\right)\right) = \exp\left(z_{i,j} - \sum_{k=i}^{j-1} \log\left(1 + \exp(z_{k,j})\right)\right) \tag{3}$$

Where $\log\left(1 + \exp(\cdot)\right)$ is commonly known as the $\mathrm{softplus}$ operation. See Appendix A for the full derivation. We further numerically stabilise $\mathrm{softplus}$ with the following computation:

$$\mathrm{softplus}(x) = \begin{cases} \log\left(1 + \exp(x)\right), & \text{if } x \leq 15 \\ x & \text{otherwise} \end{cases} \tag{4}$$

to prevent overflowing of $\exp(x)$.

Backward Let $\tilde{\boldsymbol{A}}_{i,j} = \log \boldsymbol{A}_{i,j}$, then:

$$\frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{A}}_{i,j}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{A}_{i,j}} \cdot \boldsymbol{A}_{i,j}, \qquad \frac{\partial \mathcal{L}}{\partial z_{i,j}} = \underbrace{\frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{A}}_{i,j}}}_{\text{Contribution from } i,j} - \underbrace{\sigma(z_{i,j}) \sum_{i'=1}^{j-1} \frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{A}}_{i',j}}}_{\text{Contribution from before } i,j} \tag{5}$$

The above equations dictate the direction of the order of computation for our implementation. For the forward pass (Eqn. 12), we compute from j to 1, backwards through time and accumulate $\sum_{k=i+1}^{j} \log\left(1 + \exp(z_{k,j})\right)$. For backward pass (Eqn. 5), we compute from 1 to j, accumulating $\sum_{j'=1}^{j-1} \frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{A}}_{i,j'}}$.

4.1 TRITON SPECIFICS

We modify the Triton implementation of Flash Attention for accelerating the stick-breaking attention. In theory, the memory complexity of implementing stick-breaking attention would be similar to

Algorithm 1 FORWARD thread i

```
$\boldsymbol{a} \leftarrow \mathbf{0}, \boldsymbol{O} \leftarrow \mathbf{0}$    // init $\boldsymbol{a} : d_{\text{block}} \times 1$,    $\boldsymbol{O} : d_{\text{block}} \times d_{\text{head}}$
$\boldsymbol{Q} \leftarrow$ load block $i$ of $\boldsymbol{Q}$ // load High Bandwidth Memory (HBM) to Static RAM (SRAM), $\boldsymbol{Q} : d_{\text{block}} \times d_{\text{head}}$
for $k$ in $i \dots 1$ do
    $\boldsymbol{K} \leftarrow$ load block $k$ of $\boldsymbol{K}$    // $\boldsymbol{K} : d_{\text{block}} \times d_{\text{head}}$
    $\boldsymbol{V} \leftarrow$ load block $k$ of $\boldsymbol{V}$    // $\boldsymbol{V} : d_{\text{block}} \times d_{\text{head}}$
    $\boldsymbol{Z} \leftarrow \boldsymbol{Q}\boldsymbol{K}^\top$    // $\boldsymbol{Z} : d_{\text{block}} \times d_{\text{block}}$
    $\boldsymbol{L} \leftarrow -\text{softplus}(\boldsymbol{Z})$    } ①    // $\boldsymbol{L} : d_{\text{block}} \times d_{\text{block}}$
    $\boldsymbol{A} \leftarrow \exp\left(\boldsymbol{Z} + \text{cumsum}_{\leftarrow}\boldsymbol{L} + \boldsymbol{a}\right)$    // cumulative sum right to left, $\boldsymbol{A} : d_{\text{block}} \times d_{\text{block}}$
    $\boldsymbol{O} \leftarrow \boldsymbol{O} + \boldsymbol{A}\boldsymbol{V}$
    $\boldsymbol{a} \leftarrow \boldsymbol{a} + \sum_{\leftarrow} \boldsymbol{L}$
    $\boldsymbol{a} \rightarrow$ store block $(i, k)$ of $\boldsymbol{M}$    // store from SRAM to HBM
end for
$\boldsymbol{O} \rightarrow$ store block $i$ of $\boldsymbol{o}$
```

Algorithm 2 BACKWARD-$\boldsymbol{q}$ thread i

```
$\boldsymbol{b} \leftarrow \mathbf{0}, \boldsymbol{O} \leftarrow \mathbf{0}$
$\nabla\boldsymbol{O} \leftarrow$ load block $i$ of $\nabla\boldsymbol{O}$
$\boldsymbol{Q} \leftarrow$ load block $i$ of $\boldsymbol{Q}$
for $k$ in $1 \dots i$ do
    $\boldsymbol{K} \leftarrow$ load block $k$ of $\boldsymbol{K}$
    $\boldsymbol{V} \leftarrow$ load block $k$ of $\boldsymbol{V}$
    $\boldsymbol{a} \leftarrow$ load block $(i, k)$ of $\boldsymbol{M}$
    Do ①
    $\nabla\tilde{\boldsymbol{A}} \leftarrow \boldsymbol{A} \odot (\nabla\boldsymbol{O}\boldsymbol{V}^\top)$
    $\nabla\tilde{\boldsymbol{A}}_{\text{cs}} \leftarrow \text{cumsum}_{\rightarrow}\nabla\tilde{\boldsymbol{A}}$
    $\nabla\boldsymbol{Z} \leftarrow \nabla\tilde{\boldsymbol{A}} - (1 - \exp(\boldsymbol{L})) \odot \left(\nabla\tilde{\boldsymbol{A}}_{\text{cs}} + \boldsymbol{b}\right)$
    $\boldsymbol{b} \leftarrow \boldsymbol{b} + \sum_{\rightarrow} \nabla\tilde{\boldsymbol{A}}$
    $\boldsymbol{b} \rightarrow$ store block $(i, k)$ of $\boldsymbol{N}$
    $\nabla\boldsymbol{Q} \leftarrow \nabla\boldsymbol{Q} + \nabla\boldsymbol{Z}\boldsymbol{K}^\top$
end for
$\nabla\boldsymbol{Q} \rightarrow$ store block $i$ of $\nabla\boldsymbol{Q}$
```

Algorithm 3 BACKWARD-$\boldsymbol{v}$,$\boldsymbol{k}$ thread j

```
$\nabla\boldsymbol{K} \leftarrow \mathbf{0}, \nabla\boldsymbol{V} \leftarrow \mathbf{0}$
$\boldsymbol{K} \leftarrow$ load block $j$ of $\boldsymbol{k}$
$\boldsymbol{V} \leftarrow$ load block $j$ of $\boldsymbol{v}$
for $k$ in $j \dots J$ do
    $\boldsymbol{Q} \leftarrow$ load block $k$ of $\boldsymbol{q}$
    $\nabla\boldsymbol{O} \leftarrow$ load block $k$ of $\nabla\boldsymbol{o}$
    $\boldsymbol{a} \leftarrow$ load block $(k, j)$ of $\boldsymbol{M}$
    $\boldsymbol{b} \leftarrow$ load block $(k, j)$ of $\boldsymbol{N}$
    Do ①
    $\nabla\tilde{\boldsymbol{A}} \leftarrow \boldsymbol{A} \odot (\nabla\boldsymbol{O}\boldsymbol{V}^\top)$
    $\nabla\tilde{\boldsymbol{A}}_{\text{cs}} \leftarrow \text{cumsum}_{\rightarrow}\nabla\tilde{\boldsymbol{A}}$
    $\nabla\boldsymbol{Z} \leftarrow \nabla\tilde{\boldsymbol{A}} - (1 - \exp(\boldsymbol{L})) \odot \left(\nabla\tilde{\boldsymbol{A}}_{\text{cs}} + \boldsymbol{b}\right)$
    $\nabla\boldsymbol{K} \leftarrow \nabla\boldsymbol{K} + \boldsymbol{A}^\top\nabla\boldsymbol{O}$
    $\nabla\boldsymbol{V} \leftarrow \nabla\boldsymbol{V} + \nabla\boldsymbol{Z}^\top\boldsymbol{Q}$
end for
$\nabla\boldsymbol{K} \rightarrow$ store block $j$ of $\nabla\boldsymbol{K}$
$\nabla\boldsymbol{V} \rightarrow$ store block $j$ of $\nabla\boldsymbol{V}$
```

that of Flash Attention, $O(L)$. Due to the limitations of Triton, we borrow from Dao (2023) and use a two-phased backward pass: the first phase accumulating gradients for $\boldsymbol{Q}$, and the second phase accumulates gradients for $\boldsymbol{K}, \boldsymbol{V}$. Due to the intermediate accumulated values $\boldsymbol{a}, \boldsymbol{b}$, our implementation results in a $O(L^2)$ memory complexity. However, the constant factor in the quadratic complexity here is inverse of the predetermined block size d_{block}: larger block sizes reduces the memory allocation. In our implementation, $d_{\text{block}} = 64$. Algorithms 1 details the forward pass (illustrated in Figure 2a), and Algorithms 2 & 3 details the backward pass (illustrated in Figure 2b).

Throughput We measure throughput on Dolomite Engine (Mishra, 2024) with a 1B class model on a node with 8 H100 GPUs. Stick-breaking attains a throughput of 16.27 billion tokens / day. This is a a 19.9% performance drop compared to Flash Attention, which attains throughput of 19.5 billion tokens / day. Despite the slowdown, the Triton implementation allows the method to be used for long sequences as the naive Torch implementation causes out-of-memory issues.

Conditional computation We can incorporate speedups due to the specific nature of the stick-breaking process. Since, we accumulate $\boldsymbol{A}_{k,j} \cdot \boldsymbol{v}_k$ from the diagonal to $k = 1$, when $\sum_{i=k}^{j-1} \boldsymbol{A}_{i,j} = 1$ for some k, we know that all $\boldsymbol{A}_{k',j} = 0$ where $k' < k$. Therefore, once the condition is met, we can skip all subsequent accumulations. We implement block skipping only the forward pass, and measured the time elapsed on a subset of 10,000 instances of The Pile[1]. Evaluating on 16K context lengths using the the LM evaluation harness (Gao et al., 2023) with a batch size of 1, we find that early halting allows for a 9.3% percent speed improvement.

---

[1] https://huggingface.co/datasets/NeelNanda/pile-10k

Table 1: Model hyperparameters and total size

	n_{layer}	d_{hidden}	d_{inter}	n_{head}	L	**Total params.**
350M	24	1024	2730	32	2048	367,526,912
1B	40	1536	4096	24	4096	1,208,083,968
3B	40	2304	9216	36	4096	3,513,473,280

5 EXPERIMENTS

In this section, we compare existing attention methods against stick-breaking. We first look at a modification of a synthetic task from Arora et al. (2023) to understand the inductive biases of stick-breaking. We then compare the stick-breaking against existing length extrapolation methods on a 350M model setting. We then pretrain a 1B parameter model, and evaluate it on various NLP benchmarks, and evaluate it for length extrapolation and retrieval capabilities on long context using the RULER benchmark (Hsieh et al., 2024). We also report benchmark results for a 3B model we have trained. For reference, the size of the models are detailed in Table 1.

5.1 MULTI-QUERY REPEATED ASSOCIATIVE RECALL TASK

To illustrate the inductive bias of stick-breaking attention, we first analyse its behaviour on a simple synthetic task. Arora et al. (2023) demonstrate that there is a correlation between *multi-query associative recall* (MQAR) and the performance of language modelling. They show that while Transformers, which are based on attention can easily handle MQAR, the current linear state-space models cannot. While stick-breaking can also solve the MQAR task, we formulated a different version of this toy task and tested it on both softmax attention and stick-breaking. As before, each instance of the task has an initial assignment of values to variables. However, in the query sequence, the same variable can be queried multiple times, and each query is then followed by a variable assignment, which successive queries must recall. We refer to this task as *multi-query repeated associative recall* (MQRAR).

Input	B	6	P	4	E	3	X	1	Z	2	E	2	B	1	E	5	B	4
Output	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	3	ϕ	6	ϕ	2	ϕ	1	ϕ

We compare Transformer models with Softmax and RoPE against stick-breaking attention, comparing their ability to handle MQRAR with increasing key-value pairs: from 32 to 192, in increments of 32. The full sequence length is 768. Both models are 2-layer Transformers with 256 hidden dimension and one attention head. We sweep through 4 learning rates of $\{10^{-4}, 10^{-\frac{10}{3}}, 10^{-\frac{8}{3}}, 10^{-2}\}$, and report the results of the best performing model. Softmax+RoPE is able to perform perform this task up to 128 key-value pairs, while stick-breaking is able to deal with sequences up to 192 key-value pairs.

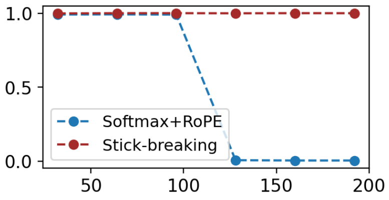


Figure 3: MQRAR performance on increasing key-value pairs.

For a more qualitative analysis, we trained another set of 2-layer Transformers of 100 dimensions on 16 key-value pairs, and visualised the patterns of the attention head. In Figure 4, we visualised the above given example. Note here that the retrieval of the third 'E' is distracted by the earlier assignment of 3 to 'E', while stick-breaking attention correctly retrieves the later assignment of 2. This may be due to the limitations of RoPE with fewer layers and fewer head dimensions (here we use 32).

5.2 350M MODEL LENGTH EXTRAPOLATION

We test the ability of stick-breaking for length generalisation, where we train on a fixed context length ($L = 1024$) and test it on longer context lengths. We started with the LLaMa 2 (Touvron et al., 2023) architecture, and modified the attention module to use the various baselines we compare against:

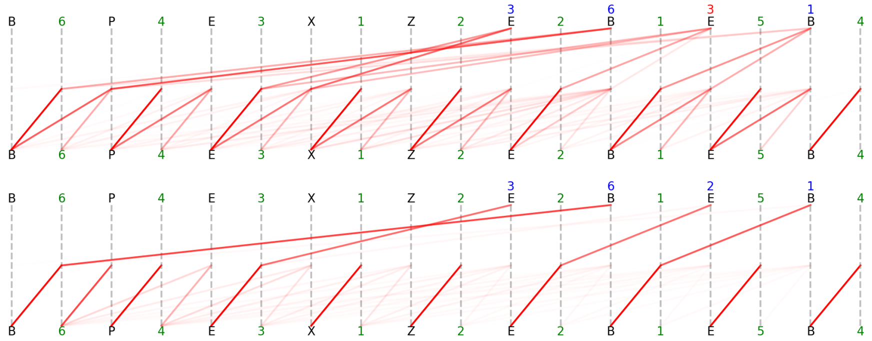
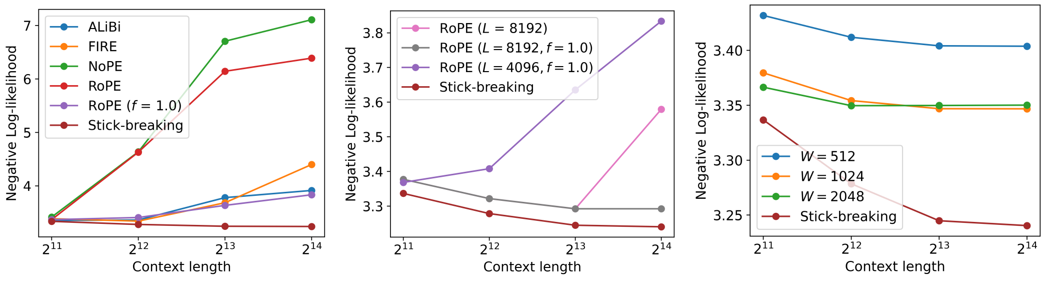

3
6
3
1
B 6 P 4 E 3 X 1 Z 2 E 2 B 1 E 5 B 4
B 6 P 4 E 3 X 1 Z 2 E 2 B 1 E 5 B 4
3
6
2
1
B 6 P 4 E 3 X 1 Z 2 E 2 B 1 E 5 B 4
B 6 P 4 E 3 X 1 Z 2 E 2 B 1 E 5 B 4

Figure 4: Attention visualisation of the models trained on the MQRAR task. The figure shows the attention for each token for the 2-layer 100-dimension Transformer. Note that in the standard Softmax+RoPE setting (above), the attention head is “distracted" at the third retrieval of ‘E’, attending to the first instance of ‘E’ rather than the mroe recent one. In the stick-breaking setting (below), each attention head attends to the prior assignment of the variable.

ALiBi (Press et al., 2021), FIRE (Li et al., 2023), NoPE (Kazemnejad et al., 2024) and the default Llama position embedding RoPE (Su et al., 2021). We trained on the first 15B tokens of SlimPajama (Soboleva et al., 2023), and evaluate it on the Wikitext benchmark in the LM evaluation harness (Gao et al., 2023) , with context lengths of 2048 to 64K. As control, we also trained a model with a context of $L = 8192$, and as expected, the performance on longer sequences was better when RoPE scaling is used (See Figure 5b).

Intuitively, we should expect a model that generalises well on longer sequences to have better likelihood as the context length is increased — longer contexts mean more information for predicting the next word. However, most methods do not generalise well to longer contexts (See Figure 5a).

Contrary to the results in Kazemnejad et al. (2024), NoPE loss increases as context lengths are extended. We also find that RoPE embeddings alone do not generalise, despite being a relative position encoding Using RoPE scaling with the scaling factor $f = 1.0$ does alleviate the issue, but still results in an increase in the loss. Surprisingly, while ALiBi is a fixed linear bias that increases with relative distance, it performs better than FIRE, which learns a function of the bias for a given relative distance. For position embeddings, our experiments suggest that RoPE with scaling works best for length extrapolation. ALiBi is an incremental bias for the logits, effectively down-weighting positions further away, eventually approaching 0. This can be viewed as a ‘soft’ windowed attention, and given the relatively good performance, we also trained 3 models with windowed attention of $W \in \{512, 1024, 2048\}$. The results suggest that windowed attention helps with preventing the



(a) Position bias and embeddings (b) Comparison with $L = 8192$ (c) Sliding window

Figure 5: Comparisons against different methods of sequence length extension. L represents the training context length, f is the RoPE scaling factor used, and W is the window size in sliding window attention. We compare against different position embeddings and biases with $L = 8192$, training with $L = 8192$, and various sliding window sizes with $L = 2048$. Note that the scale on the y-axis are different in all three plots.

Table 2: Results on the various NLP benchmarks for the 1B and 3B pretrained model. 'Softmax' benchmark is the standard Softmax + RoPE setting.

Task	**ARC-c**	**ARC-e**	**Hella.**	**OBQA**	**PIQA**	**RACE**	**SciQ**	**Wino.**	**Avg.**	**Wiki.**
	Accuracy (normalised)				*Accuracy*					*Ppl.*
1B Parameter Models										
Softmax	35.8	65.6	64.8	**38.8**	75.0	36.5	90.5	**63.4**	58.8	13.8
Stick-breaking	**37.7**	**67.6**	**65.4**	36.6	**76.0**	**37.4**	**91.9**	63.1	**59.5**	**13.4**
3B Parameter Models										
Softmax	42.2	73.1	73.2	**40.8**	78.8	37.4	93.5	67.6	63.3	11.3
Stick-breaking	**44.9**	**74.3**	**74.1**	40.4	**79.7**	**37.8**	**93.9**	**68.0**	**64.1**	**10.8**
Gemma2-2B	50.0	80.2	72.9	41.8	79.2	37.3	95.8	68.8	65.8	13.1
Qwen1.5-4B	39.6	61.5	71.4	40.0	77.0	38.2	90.0	68.1	60.7	12.5

loss from spiking when the context length is extended. As expected, $W = 512$ performs the worst. However, $W = 2048$ increases slightly as the context length is increased, while $W = 1024$ decreases till $L = 2^{13}$ and stays constant. The $W = 1024$ model is trained on context lengths of $L = 2048$, which would allow the model to learn to deal with context lengths longer than the window size, while $W = 2048$ is trained similar to the standard non-window attention model. Figure 5c shows the generalisation curves. Note that the y-axis in the plot is on a smaller scale than that of Figure 5a, indicating that the sliding window method is a relatively good method for length extrapolation as well. In all cases, stick-breaking negative log-likelihood still decreases as the context length increases, outperforming the other methods.

5.3 MODEL PRETRAINING

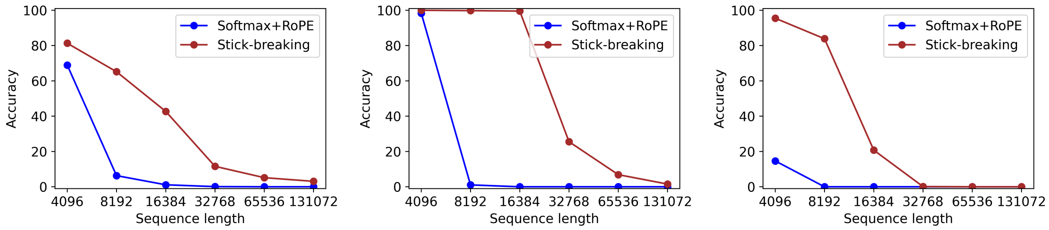
We pretrain the 1B and 3B models in this section using a two-stage training scheme in Hu et al. (2024) and the Power learning rate schedule (Shen et al., 2024). In the first stage, there is a warmup for the learning rate to 0.01, then we apply Power decay. Our training corpus has 1T tokens and mixes large-scale open-source datasets of medium quality with permissive licenses. In the second stage, we exponentially decay the learning rate to zero. The stage 2 training corpus is a mix of stage 1 data and a small amount of high-quality open-source and synthetic corpora with permissive licenses. The training batch size is 1024 and uses padding-free sequence packing for training in Dolomite Engine (Mishra, 2024). We evaluate the pretrained models on language model tasks and multiple-choice tasks from LM evaluation Harness (Gao et al., 2023). The multiple-choice tasks include: grade-school science questions (ARC; Clark et al. 2018), common sense reasoning (Hellaswag; Zellers et al. 2019), open book question answering (OpenBookQA; Mihaylov et al. 2018), physical questions (PIQA; Bisk et al. 2020), reading comprehension (RACE; Lai et al. 2017), and Winograd schema task (Winogrande; Sakaguchi et al. 2021). Table 5 shows the performance.

Table 3: MMLU few-shot results

	MMLU 0-shot	**MMLU** 5-shot
1B Parameter Model		
Softmax	25.7	25.2
Stick-breaking	**28.4**	**29.3**
TinyLlama	25.3	26.0
3B Parameter Model		
Softmax	46.1	49.1
Stick-breaking	**50.8**	**52.9**
Gemma2-2B	49.3	53.1
Qwen1.5-4B	54.2	55.2

Overall, we outperform our own pretrained standard attention models that are trained on the same settings. We perform better on average, and attain better perplexity on Wikitext. We also evaluated the models on MMLU with 0-shot and 5-shot settings. For the 1B models, we have included the results for TinyLlama (Zhang et al., 2024) Due to the inductive bias of stick-breaking, we believed that stick-breaking would perform better on MMLU in a few-shot setting as it would not be distracted by the few-shot examples provided in the context.

Stick-breaking performs surprisingly well even in the 0-shot setting, and improves with few-shot examples provided. Note that this may not always be the case, as in our Softmax + RoPE model, the performance decreases with few-shot examples in context.



(a) Overall (b) Needle in a Haystack (NIAH) (c) Variable Tracking

Figure 6: Results on the RULER benchmark. Without further finetuning for long context, we evaluate the 1B models on the RULER tasks. Subfigures: 6a shows the average over all 13 task, 6b is the average across the various NIAH tasks, 6c is the result for the variable tracking task.

We have included the the Qwen1.5-4B and Gemma2-2B models for comparison, and our 3B model underperforms Gemma2-2B, a smaller model, while it outperforms Qwen1.5-4B. This may be attributable to a variety of factors, including data mixture or training hyperparameters. Finally, we evaluate our 3B model on the GSM8K dataset (Cobbe et al., 2021). Interestingly, the 5-shot setting underperforms standard attention while CoT with 8-shot sees a 5.5% improvement.

Table 4: 3B Model GSM8K Results

	GSM8K	
	5-shot	8-shot, CoT
Softmax	**44.1**	44.2
Stick-breaking	42.3	**49.7**

5.3.1 1B LENGTH EXTRAPOLATION

We test the 1B stick-breaking and softmax model with the RULER benchmark (Hsieh et al., 2024). The benchmark consists of 'needle-in-a-haystack'-like tasks, and is generally used for testing retrieval capabilities of long-context models that are trained specifically for long contexts. In our setting, we use RULER to evaluate both 1B models trained on 4096 contexts. Accordingly, the general capabilities of these models on longer contexts are much worse than purpose-trained models.

On average, the performance of stick-breaking dominates Softmax + RoPE with scaling (Figure 6a). In the breakdown, we find that the stick-breaking model is surprisingly good at extrapolating on NIAH tasks up to 16K context lengths, while the standard model significantly drops in performance. Our results on the variable tracking (VT) task agrees with our experiments on MQRAR. The task involves tracking the variable assignments provided in the context, and we find that even in-distribution ($L = 4096$), the standard model does not perform well at this task.

6 CONCLUSION & FUTURE WORK

We propose a formulation for using the stick-breaking process as a replacement for softmax for attention. Stick-breaking attention allows us to do away with position embeddings, while still retaining model performance. We detail the specifics of implementing the stick-breaking kernel in Triton for large scale training. We then demonstrate that stick-breaking is good at length extrapolation, performing better than other position embedding and position bias methods in our 350M class models. We also show that our pretrained stick-breaking models peform better in a controlled experiment, given the same training data and training regime. On retrieval in the RULER benchmark, stick-breaking outperforms softmax attention.

The drawbacks in efficiency can be improved in future work by making similar optimisations that Flash Attention (Dao et al., 2022; Dao, 2023) made for speedups. These include making full use of features in CUDA, and by cache retrieval optimisations. We believe there is plenty of room for improvement in computation efficiency that can be made in future versions of stick-breaking. Overall, we find stick-breaking attention to be a promising replacement for Softmax + RoPE in Transformer models.

REFERENCES

M Allen and F Lambie. An environmental residual allocation model. In *Proceedings of the Conference on Environmental Modeling and Simulation, April 19-22, 1976, Cincinnati, Ohio*, pp. 236. US Environmental Protection Agency, 1976.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 1, context-free grammar. *arXiv preprint arXiv:2305.13673*, 2023.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927*, 2023.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 7432–7439, 2020.

bloc97. NTK-Aware Scaled RoPE allows LLaMA models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation., 2023. URL https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. *Advances in Neural Information Processing Systems*, 36, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. The neural data router: Adaptive control flow in transformers improves systematic generalization. *arXiv preprint arXiv:2110.07732*, 2021.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Steinar Engen. A note on the geometric series as a species frequency model. *Biometrika*, 62(3): 697–699, 1975.

Warren John Ewens. Population genetics theory-the past and the future. In *Mathematical and statistical developments of evolutionary theory*, pp. 177–227. Springer, 1990.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

RC Griffiths. Genealogical-tree probabilities in the infinitely-many-site model. *Journal of mathematical biology*, 27:667–680, 1989.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. Transformer language models without positional encodings still learn positional information. *arXiv preprint arXiv:2203.16634*, 2022.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Language modeling with deep transformers. *arXiv preprint arXiv:1905.04226*, 2019.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36, 2024.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. Functional interpolation for relative positions improves long context transformers. *arXiv preprint arXiv:2310.04418*, 2023.

John William McCloskey. A model for the distribution of individuals by species in an environment. *PhD Thesis*, 1965.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. *arXiv preprint arXiv:2404.08819*, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Mayank Mishra. Dolomite Engine: A Hyper-Optimized Library for Pretraining and Finetuning, 2024. URL https://github.com/ibm/dolomite-engine.

Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Jayaram Sethuraman. A constructive definition of dirichlet priors. *Statistica sinica*, pp. 639–650, 1994.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. Neural language modeling by jointly learning syntax and lexicon. *arXiv preprint arXiv:1711.02013*, 2017.

Yikang Shen, Zheyu Zhang, Tianyou Cao, Shawn Tan, Zhenfang Chen, and Chuang Gan. Moduleformer: Learning modular large language models from uncurated data. *arXiv preprint arXiv:2306.04640*, 2023.

Yikang Shen, Matthew Stallone, Mayank Mishra, Gaoyuan Zhang, Shawn Tan, Aditya Prasad, Adriana Meza Soria, David D Cox, and Rameswar Panda. Power scheduler: A batch size and token number agnostic learning rate scheduler. *arXiv preprint arXiv:2408.13359*, 2024.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Shawn Tan and Khe Chai Sim. Towards implicit complexity control using variable-depth deep neural networks for automatic speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5965–5969. IEEE, 2016.

Shawn Tan, Yikang Shen, Zhenfang Chen, Aaron Courville, and Chuang Gan. Sparse universal transformer. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 169–179, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. Transformers can achieve length generalization but not robustly. *arXiv preprint arXiv:2402.09371*, 2024.

A DERIVATION OF LOG-SPACE FORMULATION

$\sigma(x)$ is the sigmoid function,

$$\sigma(x) = \frac{1}{1+\exp(-x)} = \frac{\exp(x)}{1+\exp(x)} \tag{6}$$

$$1-\sigma(x) = \frac{1}{1+\exp(x)} \tag{7}$$

Since $\beta_{i,j} = \sigma(z_{i,j})$,

$$\log \beta_{i,j} = z_{i,j} - \log\left(1+\exp(z_{i,j})\right) \tag{8}$$

$$\log\left(1-\beta_{i,j}\right) = \log \frac{1}{1+\exp(z_{i,j})} = -\log\left(1+\exp(z_{i,j})\right) \tag{9}$$

We can substitute these back in,

$$\boldsymbol{A}_{i,j} = \exp\left(\log \beta_{i,j} + \sum_{k=i+1}^{j-1} \log\left(1-\beta_{k,j}\right)\right) \tag{10}$$

$$= \exp\left(z_{i,j} - \log\left(1+\exp(z_{i,j})\right) - \sum_{k=i+1}^{j-1} \log\left(1+\exp(z_{k,j})\right)\right) \tag{11}$$

$$= \exp\left(z_{i,j} - \sum_{k=i}^{j-1} \log\left(1+\exp(z_{k,j})\right)\right) \tag{12}$$

B REMAINDER METHOD

We experimented with using remaining attention weight is assigned to the current timestep $\boldsymbol{v}_i$, by modifying the output of the attention layer as follows:

$$\boldsymbol{o}_j = \sum_{i=1}^{j-1} \boldsymbol{A}_{i,j} \cdot \boldsymbol{v}_i + \left(1 - \sum_{i=1}^{j-1} \boldsymbol{A}_{i,j}\right) \cdot \boldsymbol{v}_j \tag{13}$$

This way when the attention ‘does nothing’, the value vector for the current time-step is the output for that attention head. We refer to this modification as ‘stick-breaking with remainder’ (SB w/ rem.).

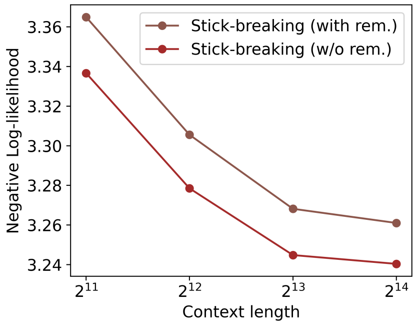


Figure 7: Length generalisation experiment on 350M model.

Table 5: Results on the various NLP benchmarks for the 1B and 3B pretrained model. ‘Softmax’ benchmark is the standard Softmax + RoPE setting.

Task	**ARC-c**	**ARC-e**	**Hella.**	**OBQA**	**PIQA**	**RACE**	**SciQ**	**Wino.**	**Avg.**	**Wiki.**
	Accuracy (normalised)				*Accuracy*					*Ppl.*
1B Parameter Models										
Softmax	35.8	65.6	64.8	**38.8**	75.0	36.5	90.5	**63.4**	58.8	13.8
Stick-breaking	**37.7**	**67.6**	**65.4**	36.6	**76.0**	**37.4**	**91.9**	63.1	**59.5**	**13.4**
SB w/ rem.	35.2	66.1	64.1	38.6	75.1	36.9	90.9	62.7	58.7	13.9

C 1B MODEL RULER RESULTS

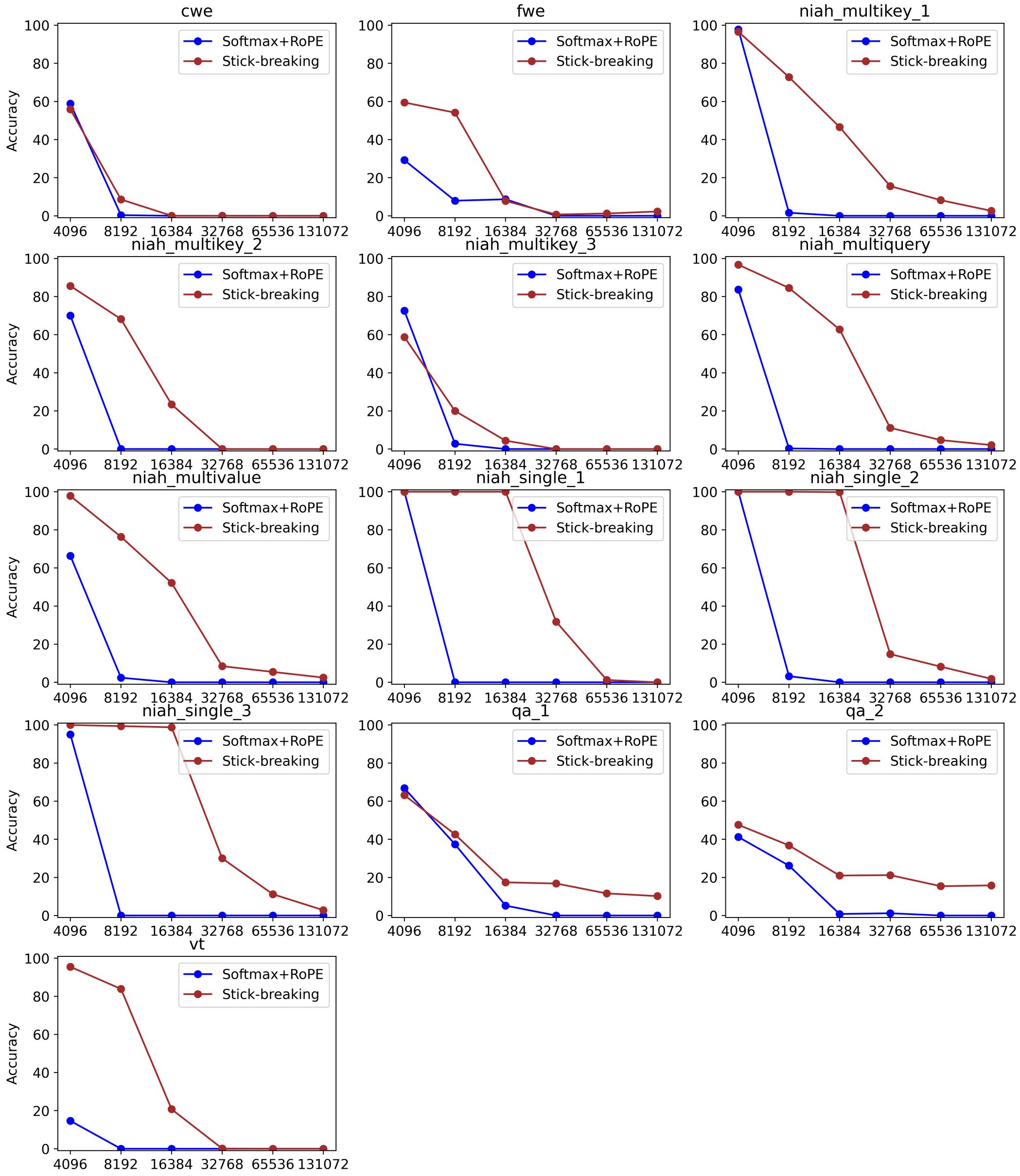


Figure 8: RULER length extrapolation results