SUS backprop: linear backpropagation algorithm for long inputs in transformers

Anonymous ACL submission

Abstract

It is straightforward to design an unbiased gradient estimator that stochastically cuts the backpropagation flow through any part of a computational graph. By cutting the parts that have little effect on the computation, one can potentially save a significant amount of backpropagation computation in exchange for a minimal increase in the stochastic gradient variance, in some situations. Such a situation occurs in the attention mechanism of the transformer architecture. For long sequences, attention becomes the limiting factor, as its compute requirements increase quadratically with sequence length n. At the same time, most attention weights become very small, as most attention heads tend to connect a given token with only a small fraction of other tokens in the sequence. These weights become promising targets for cutting backpropagation. We propose a simple probabilistic rule controlled by a single parameter c that cuts back-propagation through most attention weights, leaving at most c interactions per token per attention head. This brings a factor of c/n reduction in the compute required for the attention backpropagation, turning it from quadratic $O(n^2)$ to linear complexity $O(nc)$. We have empirically verified that, for a typical transformer model, cutting about 99% of the attention gradient flow (i.e. choosing $c \sim 25 - 30$) results in relative gradient variance increase of only about 1% for $n \sim 2000$, and it decreases with n. This approach is amenable to efficient sparse matrix implementation, thus being promising for making the cost of a backward pass negligible relative to the cost of a forward pass when training a transformer model on long sequences.

Contents

1 Introduction **1**

2 SUS backprop: Sparse Unbiased Stochastic backprop **2**

2.1 SUS backprop through attention . 3

2.2 Toy model of sparsity-variance tradeoff 3

3 Experiments and results **4**

3.1 Attention spread in transformer models 4

3.2 Empirical observation of sparsity-variance tradeoff 5

4 Related work **6**

5 Discussion **7**

6 Limitations **7**

A Attention gradients **9**

B k-weight model **9**

C Attention spread in transformers: complete account **10**

1 Introduction

The transformer architecture (Vaswani et al., 2017) plays a central role in modern machine learning, achieving state of the art results in many natural language processing tasks (Patwardhan et al., 2023) and beyond (Islam et al., 2024), finding its applications in computer vision (Dosovitskiy et al., 2020), speech recognition (Radford et al., 2023), and time-series analysis (Zhou et al., 2021), among other fields. The success of transformers is attributed to their ability to process sequential data in parallel, and to the attention mechanism which selects relevant context for every part of the input sequence.

However, a key challenge with the attention mechanism is its computational complexity, which scales quadratically with the sequence length n, making it a significant bottleneck for long sequences. This quadratic complexity arises from

the need to compute interactions between every pair of tokens in the sequence.

There have been numerous attempts to alleviate this issue, often by modifying the attention mechanism to achieve sub-quadratic complexity. To name just a few: sparse attention based on locality sensitive hashing (Kitaev et al., 2020) with $\mathcal{O}(n \log n)$ complexity, attention with kernel functions in place of standard softmax (Katharopoulos et al., 2020; Choromanski et al., 2020), and attention based on low-rank approximation (Wang et al., 2020), both with linear $\mathcal{O}(n)$ complexity. While those implementations demonstrated impressive speedups, the gains were potentially coming with a performance cost, as the originally proposed attention mechanism remains the dominant option for training language models, despite its quadratic cost.

Other approaches, notably Flash Attention (Dao et al., 2022; Dao, 2023; Shah et al., 2024), focused on improving the attention efficiency without altering the underlying algorithm. By moving computations to a smaller but faster part of GPU memory, significant speedups were gained for both forward and backward model passes.

Likewise, the method that we propose does not change the attention mechanism. Our goal is to alleviate the computational bottleneck in attention's backpropagation by changing it from quadratic to linear for long sequences. This is achieved by stochastically introducing sparsity in the gradient flow through attention, without introducing a bias in gradient estimation. Accordingly, it is called: Sparse Unbiased Stochastic backpropagation (SUS backprop). The paper provides empirical evidence that SUS backprop achieves a substantial reduction in computational cost with minimal impact on gradient variance.

The paper is organized as follows. Section 2 introduces the approach and related formalism. Section 3 provides empirical insights in attention's behavior in transformers and demonstrates a favorable sparsity-variance tradeoff of SUS backprop. Related work is discussed in Section 4 with a future work outlook given in Section 5.

2 SUS backprop: Sparse Unbiased Stochastic backprop

Consider a computation graph G (which is necessarily a directed acyclic graph), where each node i is a function $f_i(f_j, \ldots)$ that takes as its arguments the values $f_j, \ldots$ of its parent nodes that have been already computed in accordance with the topological ordering of G. Let ji denote an edge directed from j to i. Let p denote a path in G (traversed in the direction of its edges). Let $\bar{ij}$ and $\bar{p}$ be the reversed edge ji and reversed path p, respectively. Let us assign to every edge ji the partial derivative $\partial f_i / \partial f_j$ of the node i with respect to its parent j, (one can think of $\partial f_i / \partial f_j$ as a Jacobian matrix, in case of vector-valued inputs and outputs of f). Then the full derivative of any node k with respect to any other node l can be computed as a sum of products of partial derivatives along all the possible paths P_{lk} in G connecting l to k:

$$\frac{df_k}{df_l} = \sum_{p \in P_{lk}} \prod_{\bar{ij} \in \bar{p}} \frac{\partial f_i}{\partial f_j}. \tag{1}$$

Since no edge can be traversed twice by a path in G, it follows that the full derivative is a linear function of any partial derivative $\partial f_i / \partial f_j$. Therefore, the expression in Eq.(1) can be stochastized by multiplying every partial derivative $\partial f_i / \partial f_j$ by a random variable $\tilde{m}_{ij}$, which plays the role of an upweighting stochastic mask, defined as

$$\tilde{m} = \begin{cases} \frac{1}{q}, \text{ with probability } q, \\ 0, \text{ with probability } 1 - q, \end{cases} \tag{2}$$

without affecting the expectation value

$$\mathbb{E}\left[\sum_{p \in P_{lk}} \prod_{\bar{ij} \in \bar{p}} \frac{\partial f_i}{\partial f_j} \tilde{m}_{ij} \right] = \frac{df_k}{df_l}. \tag{3}$$

Moreover, the same random variable can be used with different partial derivatives, as long as they cannot be connected by a path, (otherwise, the same random variable will occur more than once on the path, thus entering non-linearly the full derivative and invalidating Eq.(3)). More generally, any coupling of random variables that is linear (in each random variable), with the random variables satisfying Eq.(2), will result in an unbiased stochastic estimator of full derivatives. For example, the coupling could be done internally inside $\partial f_i / \partial f_j$, as long as $\partial f_i / \partial f_j$ remains linear in random variables.

Thanks to the unbiased gradient estimation, we expect our approach to improve computational complexity without fundamentally altering the optimization trajectory during training, in that sense being equivalent to the original attention mechanism. Also, the unbiased gradient generally offers stronger convergence guarantees (Vicol et al., 2021).

2.1 SUS backprop through attention

We now consider a concrete example of the transformer attention mechanism. Let n be the sequence length and d be the attention head dimension. For a single attention head, the attention block takes queries Q, keys K and values V as inputs (all three are $\in \mathbb{R}^{n\times d}$), and outputs a weighted average of values $\bar{V}$:

$$\bar{V} = WV, \tag{4}$$

where the attention weights $W \in \mathbb{R}^{n\times n}$ are

$$W = \text{softmax}\left(QK^{\mathsf{T}}\right). \tag{5}$$

The softmax is applied row-wise, so $\sum_j W_{ij} = 1$. Let the gradient of a with respect to b be $\nabla_b a \equiv da/db$. Let $\mathcal{L}$ be the model loss. For the gradient of the loss with respect to the attention inputs we find (see Appendix A for details):

$$\begin{aligned} \nabla_Q \mathcal{L} &= MK, \\ \nabla_K \mathcal{L} &= M^{\mathsf{T}} Q, \\ \nabla_V \mathcal{L} &= W^{\mathsf{T}} \nabla_{\bar{V}} \mathcal{L}, \end{aligned} \tag{6}$$

where we have defined the matrix M as

$$M_{ij} = W_{ij} \sum_{\nu} \left(V_{j\nu} - \bar{V}_{i\nu}\right) \nabla_{\bar{V}_{i\nu}} \mathcal{L}. \tag{7}$$

We stochastize $\nabla\mathcal{L}$ by stochastizing $W \to \tilde{W} = W \odot \tilde{m}$ where it explicitly enters Eqs.(6,7). (Note that all the forward-computed quantities, e.g. $\bar{V}$, are unchanged by this procedure.) There is no restriction on how the acceptance probabilities q_{ij} are computed. We chose a very simple rule, controlled by a single parameter c:

$$q_{ij} = \min\{cW_{ij}, 1\}. \tag{8}$$

Because $\sum_j q_{ij} \leq c \sum_j W_{ij} = c$, we call c the attention retention parameter, as it plays the role of the upper bound (which is saturated for $c \leq 1$) on the expected number of attention interactions retained by the above rule, per token per attention head. Since only the retained weights are counted in the gradient computation in Eqs.(6,7), and the masking decisions in $\tilde{W}$ can be made in the forward run, then by controlling the sparsity of $\tilde{W}$, c also controls the memory and compute complexity of the back-propagation through attention. It takes $\mathcal{O}(nc)$ memory to store $\tilde{W}$ and $\mathcal{O}(ncd)$ operations to compute the gradient. Therefore, both time and space complexities are reduced by factor c/n (assuming that W is passed from the forward run, rather than recomputed). If W is recomputed in the backward run, then we have the same reduction in compute, while paying $\mathcal{O}(nc)$ in extra memory needed to pass $\tilde{W}$. If c is small relative to d, then this extra memory demand is inconsequential as we already pay $\mathcal{O}(nd)$ for passing (Q, K, V). One way or the other, the complexity of SUS backprop is linear in n, making it superior to any quadratic complexity algorithm for long sequences, provided the benefits of sparsity outweigh the detriments of the gradient variance increase. In this paper, we present compelling evidence in support of that assumption.

2.2 Toy model of sparsity-variance tradeoff

To gain a theoretical insight into the sparsity-variance tradeoff, we consider a toy model of the attention contribution to the gradient variance. We call it a k-weight model, because it features k distinct attention weight values. The simplest case of $k = 2$, despite its crudeness, already provides a qualitative picture of the tradeoff, as shown in Appendix B.

Let us model the attention contribution from a single token, single head, single layer perspective as

$$\nabla\mathcal{L} = \sum_{a=1}^{a=k} w_a \sum_{b=1}^{b=m_a} v_{ab}, \tag{9}$$

where m_a is the multiplicity of each attention weight value w_a and v_{ab} are independent variables of unit variance. We assume that w_a are ordered in increasing order, for convenience. Note that the form of the above equation is not limited to a single layer case. Due to the multiplicative nature of probabilities, due to the gradient being a sum of products (see Eq.(1)) and due to w and q being probabilities, the same toy model can describe a multi-layer network.

For convenience, we work with reduced quantities, where the attention weights, weight multiplicities, attention retention parameter, attention retention and gradient variance are all scaled with $n = \sum_{a=1}^{a=k} m_a$. Specifically: $\omega_a = w_a n$, $\mu_a = m_a/n$, $\xi = c/n$, and κ and Σ are the reduced attention retention and gradient variance, respectively.

For k-weight model, we derived a simple sparsity-variance tradeoff relationship (see Appendix B), that is conveniently expressed in terms of the slopes of $\kappa(\xi)$ and $\Sigma(\xi)$ and that can be

tested on empirical data:

$$\frac{d\kappa}{d\xi} = -\xi^2 \frac{d\Sigma}{d\xi}. \tag{10}$$

Let Σ_0 be the original stochastic gradient variance, which is controlled by the minibatch sampling choices and coincides with Σ in the limit $\xi \to \infty$. In case of a power-law dependence of $\kappa \propto \xi^\alpha$ and $\Sigma - \Sigma_0 \propto \xi^\beta$, the above equation implies $\alpha - \beta = 2$. How well this relationship is respected by actual language model data is tested in Section 3.2.

3 Experiments and results

Our idea, that backpropagation can be cut through most attention weights without causing a significant increase in the gradient variance, was motivated by an assumption that a typical token interacts via the attention mechanism with a relatively small number of other tokens (Bahdanau et al., 2014; Clark et al., 2019), rendering most attention weights small. To gain understanding, to what extent this assumption is correct (Araabi et al., 2024) or not (Zhai et al., 2023), we will take a closer look at the distribution of the attention weights across the heads, layers, language models and input sequence lengths (Clark et al., 2019), in the first subsection of this section. In the second subsection, we directly measure the dependence of the gradient variance increase on c and n in a language model with a custom implementation of SUS backprop algorithm. We will also compare the results with k-weight model predictions.

3.1 Attention spread in transformer models

Let us define quantities that help us characterize how much the distribution of attention weights is spread or peaked. Let top-p weights be the top weights, whose combined probability mass is $\mathsf{p} \in [0, 1]$, (the threshold parameter is fixed throughout this section to $\mathsf{p} = 0.9$). Let the attention spread s_i for a token at position i be the number of top-p weights among W_{ij}. The spread fraction s_i/i indicates the fraction of tokens that significantly contribute to the attention output $\bar{V}_i$, (contribute on average, assuming statistical independence of V_j). Let us define the aggregate spread fraction ϕ_i as:

$$\phi_i = \frac{\sum_{j=0}^{j=i} s_j}{\sum_{j=0}^{j=i} j}. \tag{11}$$

The spread fraction ϕ_i describes the fraction of top-p reduced weights $\omega_{jk} = (j+1)W_{jk}$ among all the weights with $k \leq j \leq i$. While s_i is suitable for describing the attention of the token at position i, ϕ_i is suitable for describing the attention of all the tokens up to position i. As an aggregate quantity, ϕ_i is much less sensitive to statistical noise, than s_i.

We measure s_i and ϕ_i by averaging over 1000 input sequences. Four models – *opt-125m, opt-350m, opt-1.3b,* (Zhang et al., 2022) and *Mistral-7B-v0.1* (Jiang et al., 2023) – were tested on the sequences from *The Pile* dataset (Gao et al., 2020). The first three smaller models were tested on the full span of permitted input length (2048 tokens), while Mistral could be tested up to maximum 1570 tokens (on A100 GPU).

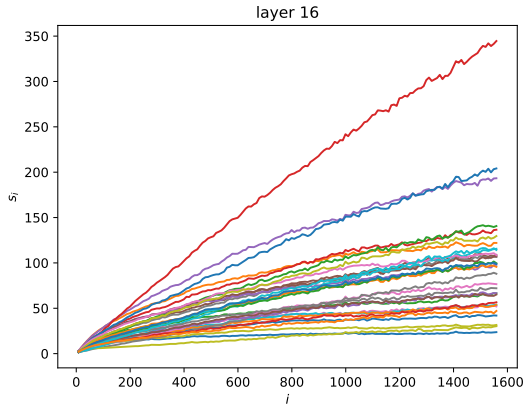

Figure 1: Attention spread s_i vs token position i for 32 attention heads of layer 16, Mistral model.

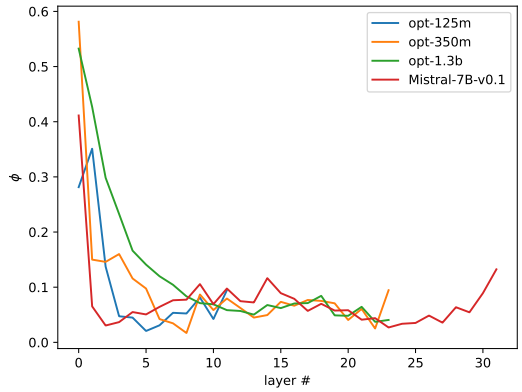

Figure 2: Aggregate spread fraction ϕ_i, averaged over layer's heads, vs layer index for different models. For OPT models, $i = 2000$; for Mistral, $i = 1550$.

Measuring the attention spread s_i across different attention heads of different layers of different models, we observed a great variation in the spread behavior. While it is impossible to reflect all that

diversity in a single illustration, a somewhat common situation in a layer is shown in Fig.(1), (see Appendix C for a complete account of all layers in all models). Typically, the majority of heads show a limited spread s_i, which increases sub-linearly with i. Still, in a few heads the attention is spread much wider, with the spread trending closer to linear. The distribution of heads by their spread is shown in Fig.(3), (note the logarithmic scale of the spread). The heads with wider spread are less numerous, but they disproportionately contribute to the arithmetic mean of the spread, while the more focused heads, which are more numerous, dominate the geometric mean of the spread. These two measures of attention spread are noticeably different, with the geometric mean being significantly lower, see Fig.(4). Notice how the ratio of the arithmetic to geometric means keeps increasing with the sequence length. Encouragingly, the largest and most advanced model demonstrates overall the lowest attention spread.

The presence of heads with a large attention spread may potentially seem problematic for SUS backprop, since, at face value, they violate the assumption of every token talking to only a few counterparts. In practice, cutting backpropagation through the majority of their seemingly equally important weights did not appear to cause much problem, as follows from the results of the next section. We can only speculate on the reason underlying that insensitivity: either information coming from various values V_i is greatly redundant for those heads, or they simply do not extract valuable information (Voita et al., 2019).

The evolution of aggregate attention spread fraction ϕ across model's layers reveals a loosely similar pattern (Clark et al., 2019) in different models. The spread is largest at the lowest layers, then it quickly drops with the layer index increasing, remaining approximately at its lowest values (about ~ 0.1 for the mean spread for $n \sim 2 \times 10^3$) through most of the layers, finally showing a slight increase at the topmost layers, see Fig.(2).

Additionally, we observed various artifacts, such as intricate, non-monotonic behavior of some heads, especially in the OPT series, see Appendix C. In some heads the spread saturates, in some - starts decreasing after an initial increase, yet in some other heads it follows roughly oscillatory patterns of various periodicities. The occurrence of those heads does not follow any obvious pattern, appearing at different layers and model sizes, in the models sharing the same architecture.

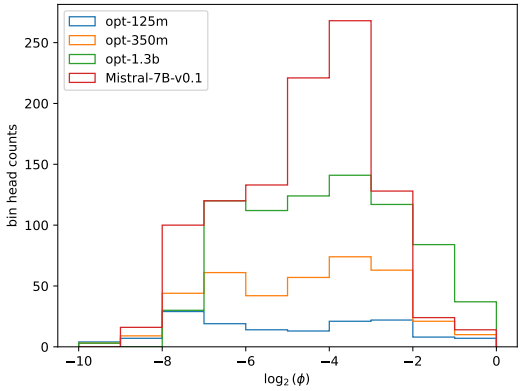

Figure 3: Distribution of attention head log-spreads $\log_2 \phi_i$ in different models. For OPT models, $i = 2000$; for Mistral, $i = 1550$.

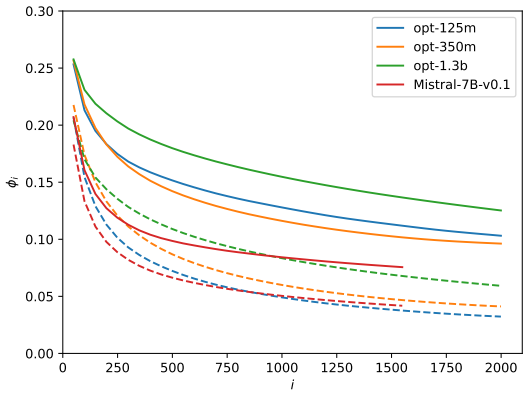

Figure 4: Aggregate spread fraction ϕ_i, averaged over all heads and layers, vs token position i for different models. Solid and dashed lines of the same color represent arithmetic and geometric mean, respectively, for the same model.

3.2 Empirical observation of sparsity-variance tradeoff

We have implemented SUS-backprop algorithm by modifying Hugging Face's attention code for OPT model. The part of the code, that computes the attention weights is replaced by a function with a custom backward() method, that implements the loss gradient with respect to (Q, K, V) as described in Eqs.(6,7). The forward() method implements the weight matrix W stoshastization, saving for the backward pass a sparse matrix $\tilde{W}$, requiring $\mathcal{O}(nc)$ memory (in sparse representation). For testing purposes, the algorithm is implemented in both dense and sparse representations.

From practical standpoint, the most important

results are presented in this section, where we test how much the sparsity of $\tilde{W}$ affects the relative increase in gradient variance, which we define as

$$\rho = \frac{\Sigma - \Sigma_0}{\Sigma_0}. \tag{12}$$

For our approach to have a practical significance, we ideally would like to see that for $c \lesssim d$ (implying a moderate demand in additional memory for storing $\tilde{W}$) we have a relatively small $\rho \ll 1$. Indeed, for sufficiently large n, a linear time complexity backprop algorithm should give a factor of $2-3$ speedup, by making the backward pass time negligible in comparison to the forward pass time. But if this speedup is accompanied by a noticeable increase in variance, say, $\rho \sim 1$, it may negate the gains in compute per data sample, as we would need to process significantly more data samples, to counter the effects of the increased gradient variance. Fortunately, we found that for $c \sim 25-30$ and $n = 2000$ we already have $\rho \sim 0.01$, and importantly, ρ keeps decreasing with increasing n. This is shown in Fig.(5).

We measure the gradient variance as a mean variance over all gradient components, averaged over 100 input sequences for every data point in the plots. In the left panel of Fig.(5) the variance increase ρ and the attention retention κ are plotted against the attention retention parameter c for a fixed $n = 2000$. It also shows rather accurate power-law fits of $\rho \propto \xi^{\beta}$ and $\kappa \propto \xi^{\alpha}$, which let us test the relationship $\alpha - \beta = 2$, predicted by k-weight model in Section 2.2. We found $\alpha \approx 0.78$ and $\beta \approx -1.50$, resulting in $\alpha - \beta = 2.28$, which is not too far off from the theoretical prediction. See more in Appendix B on 2-weight model fitting.

In the right panel of Fig.(5) ρ and κ are plotted against the input sequence length n for a fixed $c = 30$. It shows that ρ initially quickly increases with n, reaching the maximum value of about ~ 0.033 at $n \sim 200$, but then the trend reverses and ρ keeps decreasing with n, reaching just under 0.01 at $n \sim 2000$. Assuming this trend continues for larger n and is generic for transformer models, the above results provide strong evidence that SUS backprop can bring significant compute savings for training with long sequences.

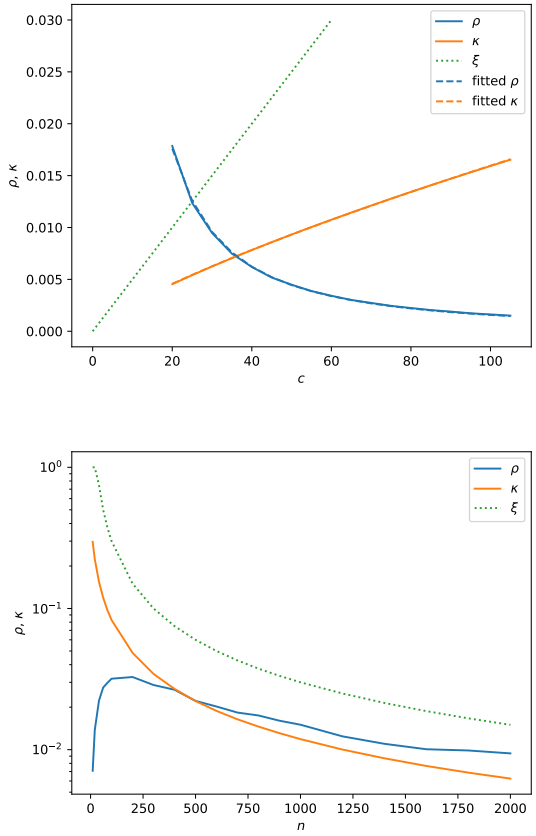

Figure 5: opt-125m: relative gradient variance increase ρ and reduced attention retention κ. The dotted line ξ is the upper bound on κ. Left panel: ρ and κ vs c for sequence length $n = 2000$. The dashed lines are power-law fits: $\rho \propto \xi^{-1.5}$ and $\kappa \propto \xi^{0.78}$. Right panel: ρ and κ vs n for attention retention parameter $c = 30$.

4 Related work

Our formulation of SUS backprop is directly applicable to the original Multi-Head Attention (MHA) mechanism (Vaswani et al., 2017), which treats the heads as a disjoint set of (Q, K, V) triplets. There have been proposed multiple attention mechanism modifications, that fundamentally do not change its quadratic complexity $\mathcal{O}(n^2)$, but can realize a more economical use of memory and compute resources. Some of the most notable variations include Multi-Query Attention (MQA) (Shazeer, 2019), Grouped-Query Attention (GQA) (Ainslie et al., 2023) and Multi-head Latent Attention (MLA) (Liu et al., 2024). MQA, prominently used in PaLM model (Chowdhery et al., 2023), employs multiple queries Q, but shared KV pairs. GQA (used by LLaMA (Touvron et al., 2023) and Mistral (Jiang et al., 2023), among others) interpolates between MHA and MQA, by mapping groups of queries to distinct KV pairs. MLA (used by DeepSeek (Liu et al., 2024)) resorts to low rank compression of KV matrices, that reduces memory footprint. Our method

is in principle compatible with these attention variants, but would require appropriate modification of attention gradient equations Eqs.(6,7).

A recently proposed Native Sparse Attention (Yuan et al., 2025), offering an order of magnitude speedups over standard full attentions (such as MQA and GQA) thanks to the use of compressed token blocks and hardware-efficient implementation, is still asymptotically a quadratic complexity algorithm, which can potentially benefit from backpropagation sparsification.

There also have been proposed many attention mechanisms with nominal $\mathcal{O}(n)$ complexity. Some mechanisms are based on approximate search (Reformer (Kitaev et al., 2020)), others rely on low-rank approximations (Linformer (Wang et al., 2020)), kernel methods to replace softmax (Performer (Choromanski et al., 2020)) or sparsity (Longformer (Beltagy et al., 2020), BigBird (Zaheer et al., 2020)). These approaches share an idea that explicit computation of negligible (or otherwise unimportant) attention weights can be avoided during the forward run. In which case, the gradient flow in the backward run is unlikely to be sparse, so there would be little room for improvement left for a backpropagation sparsification algorithm, such as SUS backprop. However, the competitiveness of forward $\mathcal{O}(n)$ algorithms remains to be convincingly demonstrated.

A stochastic sparsification of backpropagation was employed in Refs. (Cheng et al., 2022; Fang et al., 2022) in the context of training video models, but unlike in our work, no upweighting was used to ensure unbiased gradient estimation. Also, the gradient flow was cut randomly, relying on redundancy naturally present in video inputs, rather than selecting parts that have little effect on forward computation, as in our work.

5 Discussion

In this paper, we presented a novel backpropagation algorithm – SUS backprop – that sparsifies the flow of gradient through attention in a stochastic and unbiased manner. We presented a compelling evidence, that this sparsification can transform the computational complexity of the backward pass from quadratic to linear, at the cost of only a small increase in gradient variance.

Yet more work remains to be done. To make this approach practically useful and competitive, it needs a more efficient implementation. Our current implementation fully relies on native PyTorch constructs from its sparse matrix module. This was sufficient to prove the concept and to demonstrate a linear compute complexity of the backward pass over quadratic complexity of the forward pass. However, due to serious limitations of PyTorch sparse matrix support, we incurred unnecessarily large overhead, offsetting the gains from backpropagation by the slowdown in the forward run. It will most likely require writing a custom kernel to overcome the current sparse module limitations.

While our analysis shows a very favorable sparsity-variance tradeoff, both in sequence length n and sparsity control parameter c, indicating a great potential of SUS backprop for long sequence training, the ultimate verdict should come from actual training results, be it a supervised fine tuning or full fledged pretraining of a language model. It may also be interesting to observe the evolution of attention spread during pretraining, by analyzing available checkpoints of large language models.

6 Limitations

Modern full-attention implementations rely on optimizations that avoid the materialization of W to minimize the number of slow memory accesses. For example, Flash Attention resorts to an online softmax evaluation. The stochastic sparsification procedure of SUS backprop would need to be suitably modified to be compatible with Flash Attention. In addition, the fine-grained stochastization that we experimented with, where every W_{ij} is treated independently, may be computationally prohibitive for highly optimized GPU-accelerated dense attention computation. In that case, one may need to use a more coarse-grained block-level stochastization of W.

References

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Ali Araabi, Vlad Niculae, and Christof Monz. 2024. Entropy–and distance-regularized attention improves low-resource neural machine translation. In *Proceedings of the 16th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 140–153.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

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

Feng Cheng, Mingze Xu, Yuanjun Xiong, Hao Chen, Xinyu Li, Wei Li, and Wei Xia. 2022. Stochastic backpropagation: A memory efficient strategy for training video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8301–8310.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, and 1 others. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

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

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and 1 others. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Jun Fang, Mingze Xu, Hao Chen, Bing Shuai, Zhuowen Tu, and Joseph Tighe. 2022. An in-depth study of stochastic backpropagation. *Advances in Neural Information Processing Systems*, 35:21819–21830.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Saidul Islam, Hanae Elmekki, Ahmed Elsebai, Jamal Bentahar, Nagat Drawel, Gaith Rjoub, and Witold Pedrycz. 2024. A comprehensive survey on applications of transformers for deep learning tasks. *Expert Systems with Applications*, 241:122666.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

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

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

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, and 1 others. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Narendra Patwardhan, Stefano Marrone, and Carlo Sansone. 2023. Transformers in the real world: A survey on nlp applications. *Information*, 14(4):242.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. 2024. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37:68658–68685.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Paul Vicol, Luke Metz, and Jascha Sohl-Dickstein. 2021. Unbiased gradient estimation in unrolled computation graphs with persistent evolution strategies. In *International Conference on Machine Learning*, pages 10553–10563. PMLR.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

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

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, and 1 others. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and 1 others. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M Susskind. 2023. Stabilizing transformer training by preventing attention entropy collapse. In *International Conference on Machine Learning*, pages 40770–40803. PMLR.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115.

A Attention gradients

We want to compute the gradient of the loss function $\mathcal{L}$ with respect to the attention inputs Q, K and V. In matrices, we use Latin letters for token position indices and Greek letters for head dimension indices. To simplify notations, we may omit some matrix indices or gradient subscripts, when it does not create a confusion. To compute

$$\nabla\mathcal{L} = \sum_{j\nu} \left(\nabla\bar{V}_{j\nu}\right) \nabla_{\bar{V}_{j\nu}}\mathcal{L}, \tag{13}$$

we need to compute:

$$\nabla\bar{V}_{j\nu} = \sum_k \left(\left(\nabla W_{jk}\right) V_{k\nu} + W_{jk}\nabla V_{k\nu}\right). \tag{14}$$

The attention weights, given by Eq.(5), are:

$$W_{jk} = \frac{e^{S_{jk}}}{\sum_l e^{S_{jl}}}, \quad S_{jk} = Q_j K_k^{\mathsf{T}}. \tag{15}$$

Note that $\nabla_Q V = \nabla_K V = \nabla_V W = 0$. Differentiating W with respect to Q and K we find:

$$\begin{aligned} \nabla_{Q_i} W_{jk} &= \delta_{ij} W_{jk} \left(K_k - \sum_l W_{jl} K_l\right), \\ \nabla_{K_i} W_{jk} &= W_{jk}\left(\delta_{ik} - W_{jl}\right) Q_j. \end{aligned} \tag{16}$$

Also,

$$\nabla_{V_{i\mu}} V_{k\mu} = \delta_{ik}\delta_{\mu\nu}. \tag{17}$$

From Eq(14) and Eqs(16,17) we find:

$$\begin{aligned} \nabla_{Q_{i\mu}} \bar{V}_{j\nu} &= \delta_{ij} \sum_k W_{jk}\left(V_{k\nu} - \bar{V}_{j\nu}\right) K_{k\mu}, \\ \nabla_{K_{i\mu}} \bar{V}_{j\nu} &= W_{ji}\left(V_{i\nu} - \bar{V}_{j\nu}\right) Q_{j\mu}, \\ \nabla_{V_{i\mu}} \bar{V}_{j\nu} &= W_{ji}\delta_{\mu\nu}. \end{aligned} \tag{18}$$

From Eq(13) and Eq(18) we find:

$$\begin{aligned} \nabla_{Q_{i\mu}}\mathcal{L} &= \sum_j W_{ij} \sum_\nu \left(V_{j\nu} - \bar{V}_{i\nu}\right) K_{j\mu} \nabla_{\bar{V}_{i\nu}}\mathcal{L}, \\ \nabla_{K_{i\mu}}\mathcal{L} &= \sum_j W_{ji} \sum_\nu \left(V_{i\nu} - \bar{V}_{j\nu}\right) Q_{j\mu} \nabla_{\bar{V}_{j\nu}}\mathcal{L}, \\ \nabla_{V_{i\mu}}\mathcal{L} &= \sum_j W_{ji} \nabla_{\bar{V}_{j\mu}}\mathcal{L}. \end{aligned} \tag{19}$$

From where the equations Eqs(6,7) in Section 2 follow.

B k-weight model

The weights and their multiplicities in k-weight model are constrained by the following equations, expressing that multiplicities sum up to n and weights sum up to 1, respectively:

$$\begin{aligned} \sum_{a=1}^{a=k} \mu_a &= 1, \\ \sum_{a=1}^{a=k} \mu_a \omega_a &= 1. \end{aligned} \tag{20}$$

Attention retention κ (which is the expectation of the accepted weights) and gradient variance Σ

(which is the variance of the upweighted accepted terms of the gradient in Eq.(9)) are expressed in terms of multiplicities μ_a, weights ω_a and acceptance probabilities q_a as

$$\kappa = \sum_{a=1}^{a=k} \mu_a q_a, \qquad \Sigma = \sum_{a=1}^{a=k} \frac{\mu_a \omega_a^2}{q_a}, \tag{21}$$

where $q_a = \min\{\xi\omega_a, 1\}$. Let ξ be in the interval $1/\omega_l \leq \xi \leq 1/\omega_{l-1}$. Then:

$$\kappa = \xi \sum_{a=1}^{a<l} \mu_a \omega_a + \sum_{a=l}^{a=k} \mu_a, \qquad \Sigma = \frac{1}{\xi} \sum_{a=1}^{a<l} \mu_a \omega_a + \sum_{a=l}^{a=k} \mu_a \omega_a^2. \tag{22}$$

From the above equations, Eq.(10) of Section 2.2 readily follow.

For $k = 2$ everything can be parameterized by two real numbers $\theta_\pm$ via $\omega_\pm = e^{\pm e^{\theta_\pm}}$ (we use $\pm$ as two values of index a in ω_a). For the multiplicity we find $\mu_\pm = (1-\omega_\mp)/(\omega_\pm - \omega_\mp)$. We list below κ and Σ for different ranges of ξ. For $\xi \leq 1/\omega_+$ we have

$$\kappa = \xi, \quad \Sigma = \frac{1}{\xi}, \tag{23}$$

for $1/\omega_+ \leq \xi \leq 1/\omega_-$ we have

$$\kappa = \xi \frac{(\omega_+ - 1)\omega_-}{\omega_+ - \omega_-} + \frac{1-\omega_-}{\omega_+ - \omega_-}, \qquad \Sigma = \frac{1}{\xi} \frac{(\omega_+ - 1)\omega_-}{\omega_+ - \omega_-} + \frac{\omega_+^2 (1-\omega_-)}{\omega_+ - \omega_-}, \tag{24}$$

and for $1/\omega_- \leq \xi$ we have

$$\kappa = 1, \quad \Sigma = \omega_+(1-\omega_-) + \omega_-. \tag{25}$$

The last expression represents the original gradient variance Σ_0. Fitting 2-weight model to empirical data from Section 3.2, we find $\theta_- = 0.086$ and $\theta_+ = 2.08$. The predicted $\rho(\xi)$ and $\kappa(\xi)$ are described by Eq.(24), as the range of c in Fig.(5) falls within the middle range $1/\omega_+ < \xi < 1/\omega_-$, hence $\alpha = 1$ and $\beta = -1$. These exponents describe the sparsity-variance tradeoff qualitatively correctly, but they lack the quantitative accuracy of power-law fits of Section 3.2 with $\alpha = 0.78$ and $\beta = -1.50$.

C Attention spread in transformers: complete account

This appendix contains the figures showing the attention spread s_i vs token position i for each attention head, grouped by layers, for all four model that we have considered: *opt-125m* in Fig.(7), *opt-350m* in Fig.(8), *opt-1.3b* in Fig.(9), and *Mistral-7B-v0.1* in Fig.(6). In all cases we used $\mathsf{p} = 0.9$. The spread s_i data points are plotted in 10 token position increments, averaged over 10 tokens, correspondingly.

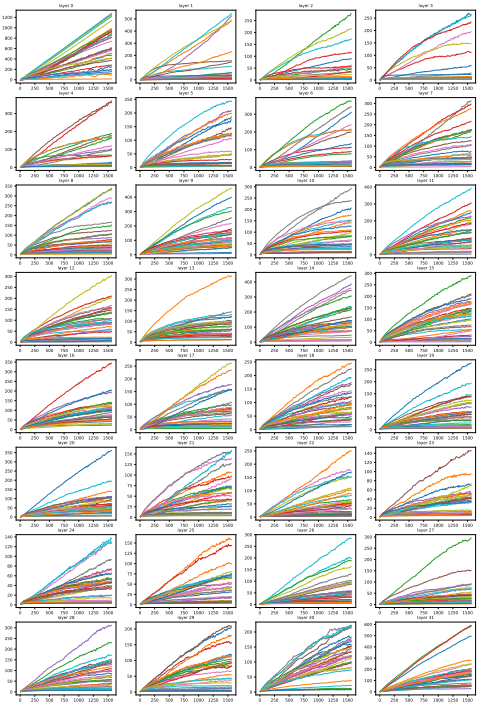

Figure 6: Mistral-7B-v0.1: attention spread s_i vs token position i for every layer and head; 32 layers, 32 attention heads in each layer.

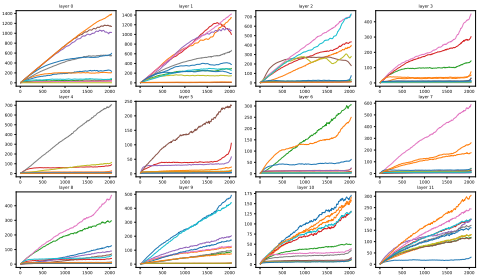

Figure 7: opt-125m: attention spread s_i vs token position i for every layer and head; 12 layers, 12 heads in each layer.

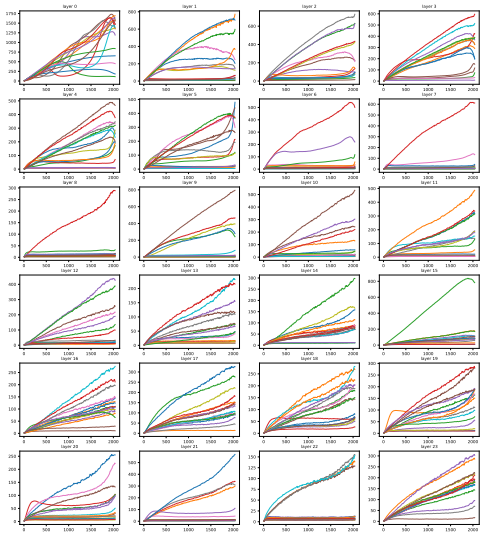

Figure 8: opt-350m: attention spread s_i vs token position i for every layer and head; 24 layers, 16 heads in each layer.

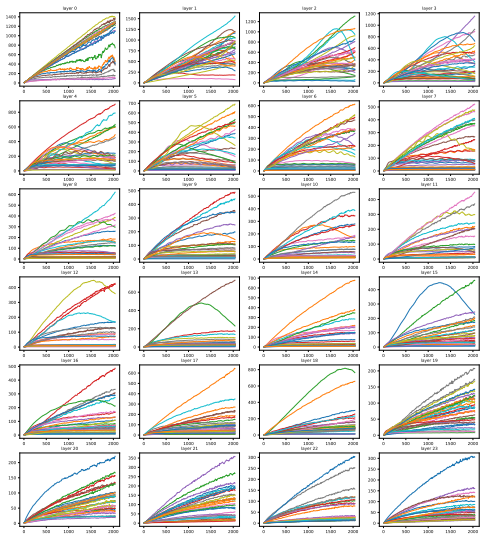

Figure 9: opt-1.3b: attention spread s_i vs token position i for every layer and head; 24 layers, 16 heads in each layer.