Know What You Need: Efficient Activation Checkpointing

Danyal Ahmed
Sorbonne University
Paris, France
danyal.ahmed@sorbonne-universite.fr

Edouard Oyallon
CNRS, Sorbonne University
Paris, France

Abstract

Large Language Model training is increasingly constrained by memory capacity. Activation Checkpointing reduces activation memory by avoiding storing selected activations during the forward pass and rematerializing them during backpropagation, thereby trading memory savings for additional computation. Although this technique is essential for scaling model training, rematerialization can be prohibitively expensive, particularly for Transformer linear layers. Approximate computation could reduce this overhead, but conventional approaches either introduce errors that accumulate over the backward pass or fail to exploit information available from the initial forward pass. We introduce a novel approximate rematerialization method that can yield unbiased gradient estimates: using information collected during the initial forward pass, the method selectively recomputes only the most important layer components, either through greedy selection or randomized sketching. Ablation studies show that the resulting gradients closely match the exact gradients. When applied to feed-forward layers, attention layers, or both, our method can recompute only two-thirds of the layer components in a Llama-488M model, while increasing perplexity by just 1.7%, 1.2%, and 2.5%, respectively.

1 Introduction

Large Language Models (LLMs) have progressed largely through scaling, resulting in models with a larger number of parameters that need to process more data in parallel [13, 17]. At the same time, accelerator throughput has improved much faster than memory capacity [48]. This trend has made memory overhead a major bottleneck, particularly during gradient computation: while model states represent a constant fraction of the memory which can be sharded [14, 37, 41, 47], the activation memory also scales with batch size and context length [54]. This makes activations a particularly challenging source of memory pressure during training, since they must be stored during training for computing gradients. Indeed, these gradients are typically computed using reverse-mode automatic differentiation [3, 32] over a computation graph, where operations are represented as nodes of this graph. During the forward pass, intermediate activations are stored and kept in memory until the backward pass, where they are used to compute gradients and are progressively released. Thus, this activation storage can represent a substantial memory overhead and must therefore be carefully managed.

Several strategies have been proposed to reduce this overhead, typically by trading memory savings for additional data movement or computation. One approach is to offload tensors, such as model states [42] or activations [43], from the accelerator to CPU memory. However, this can be inefficient in practice because of the high cost of repeated data transfers between devices. Another approach is to carefully schedule pipeline execution to reduce peak memory usage [25], but such schedules often introduce their own drawbacks, such as communication overhead, known as pipeline bubbles

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

[23], and load imbalance across layers [28], since earlier pipeline stages typically retain activation memory for longer as they are the last to process gradients. A notable example is the V-shaped pipeline schedule used in [12], which improves memory efficiency under specific settings but comes with additional scheduling constraints.

A natural way to avoid storing activations is to recompute them on the fly during the backward pass. For instance, instead of storing activations during the forward pass, invertible architectures [16, 24, 44] use model invertibility to progressively reconstruct them while simultaneously computing gradients during the backward pass, at the cost of additional computation. This idea of reconstructing activations is closely related to Activation Checkpointing (AC) [19], which also recomputes selected parts of the computation graph during the backward pass. Any missing activations required during backpropagation are recomputed on the fly from the latest available checkpoint, enabling gradient computation with lower memory usage at the cost of additional computation, which can cause up to a 33% compute overhead compared to a vanilla backward pass. Consequently, AC trades memory overhead for computation, but its benefits are layer-dependent: selective AC [30] allows one to choose the layers for which this trade-off is favorable. For instance, thanks to its IO-aware implementation, FlashAttention [11] recomputes the attention module during the backward pass, with a negligible overhead: the most important intermediate quantities are rematerialized in the accelerator cache. In contrast, linear layers typically retain matrix-multiplication outputs, since these operations are expensive to rematerialize during backpropagation.

In this work, we challenge the need to fully recompute expensive linear layers to obtain the memory benefits of Full AC. In contrast to existing approaches that approximate both the forward and backward pass or the gradient computation alone [18, 36, 50], we propose an approximation strategy tailored specifically to AC: we approximate the computations performed during rematerialization. Accordingly, the approximation we use stems from the observation that many monolithic computational blocks can be expressed as linear combinations of elementary components, and can therefore be effectively approximated through sketching [53]. *Consequently, only a fraction of the activations produced by linear layers need to be rematerialized by AC.* Since checkpoints are stored exactly, the approximation is repeatedly reset to reliable intermediate states, preserving unbiased gradient estimates and preventing errors from propagating over long computation chains. We propose KEAP (**K**now What You Need: **E**fficient **A**ctivation Check**p**ointing), an approximation method that leverages activation checkpointing by selectively approximating the computations needed during recomputation, thanks to an initial exact forward pass.

Contributions. We make the following contributions. **(1)** We introduce a novel AC strategy, KEAP, that approximates rematerialization using only information obtained from the full forward pass. **(2)** We show how to maintain high-quality gradients, either through unbiased estimation or through a greedy approximation scheme. **(3)** We explain how to apply the method to both FFN and attention layers. **(4)** We demonstrate numerically, on up to 488M parameter Llama-style models, that KEAP retains the activation-memory benefits of full AC while causing only limited perplexity degradation. **(5)** We provide ablation studies to better understand the trade-off between AC approximation quality and model accuracy.

Structure of the paper. Sec. 3 introduces the necessary background on AC, while Sec. 3.2 shows how KEAP can recover unbiased gradients, followed by a discussion on the computational benefits in Sec. 3.3. Sec. 4 presents our numerical experiments, followed by ablation studies in Sec. 4.3. Our source code can be found at https://github.com/da462/KEAP.

2 Related Work

Activation rematerialization. A natural way to avoid storing activations is to recompute them during the backward pass [19]. Several works study optimal or near-optimal rematerialization strategies, for instance through checkpointing schedules [1, 4, 20, 21]. Closely related to this line of work, Chen et al. [8] improves AC by adaptively deciding, for each activation tensor, whether it should be recomputed during the backward pass or stored in compressed form, thereby reducing memory usage while limiting recomputation overhead. Invertible neural networks provide another way to reduce activation storage by reconstructing activations during the backward pass, in particular through

lifting-based reversible architectures [16, 24]. Similar to standard AC, such reversible blocks can also be combined with checkpointing when only part of the network is reversible [44].

Approximating activations for backward computation. Combinable with AC, approximate-activation methods store compressed or approximate versions of activation states during the forward pass to save memory. Chakrabarti and Moseley [7] showed that using compressed activations has limited impact on training quality. Liu et al. [33] extend this idea to the AC setting, where rematerialization starts from quantized checkpoints, without trying to reduce computational footprint. In contrast to KEAP, the local Jacobians are then evaluated at approximate rather than exact forward activations, allowing approximation errors to propagate through the backward computation. Relatedly, Kong et al. [29] reduce per-sample storage in pipeline-parallel training through communication compression, while Shamshoum et al. [46] compress activations during the forward pass and reuse them during backpropagation. However, these methods typically decode compressed activations during the backward pass and therefore mainly reduce memory, not recomputation. In contrast, KEAP performs only a subset of the original rematerialization computations, for both parameter-gradient and input-gradient computations during the backward pass.

Approximate backpropagation computation. Approximate-gradient methods reduce the cost of the backward pass directly, and could also be combined with AC. Sun et al. [50] exploit the empirical sparsity of per-example gradients by keeping only the top-k entries of the incoming gradient, yielding a biased lower-FLOP backward pass. Chmiel et al. [9] instead enforce unbiasedness for sparse backpropagated gradients and show that the random pruning mask should minimize variance rather than sparsification error [45]. By partitioning the computational graph into independent segments, Belilovsky et al. [5] reduce activation-memory overhead, but at the cost of a degraded accuracy. The local vector-Jacobian product itself has also been approximated with randomized unbiased estimators [2], although this still requires ad hoc computations and also degrades accuracy. In contrast, KEAP approximates the rematerialization introduced by AC: the backward pass is computed through an approximate recomputation path rather than by directly modifying the error signal or the backward rule, while aiming to maintain a high quality gradient approximation.

Sketching strategies for neural networks. As we do, sketching [53] has also been used to approximate neural-network computations. Useful to our work, Daniely et al. [10] propose a theoretical framework for understanding sketches of one-hidden-layer neural networks, while Gazagnadou et al. [15] study related sketching ideas for ridge regression (a zero-hidden layer neural network). These works demonstrate the existence of useful sketches for forward computations. In a similar spirit, Yang et al. [56] combine sketching with natural-gradient methods to reduce the cost of second-order optimization. Our setting differs in that sketching is applied specifically to the rematerialization path induced by AC. We use statistics from the first exact forward pass to remove components during re-materialization. Due to their depth, deep neural networks are generally difficult to combine with sketching, as approximation errors would accumulate and propagate across many layers. Our method avoids this issue by applying sketching only within the recomputation path between exact checkpoints, leading to faithful approximations.

3 Method

3.1 Background on AC

In the following, we represent the forward pass of a neural network (typically a transformer) as a cascade of nonlinear operators ϕ_j (that could be either a block of layers or a single layer), given by

$$x_{j+1} = \phi_j(x_j, \vartheta_j), \qquad \text{(Forward } j\text{)}$$

where ϑ_j denotes the set of parameters of layer j, with final loss $\mathcal{L}$. In standard backpropagation, the forward pass is followed, for each layer j, by the adjoint-state backpropagation

$$\delta_j = \left(J_{x_j}\phi_j(x_j, \vartheta_j)\right)^\top \delta_{j+1}, \qquad \frac{\partial \mathcal{L}}{\partial \vartheta_j} = \left(J_{\vartheta_j}\phi_j(x_j, \vartheta_j)\right)^\top \delta_{j+1}\,. \qquad \text{(Backward } j\text{)}$$

where $\delta_j = \frac{\partial \mathcal{L}}{\partial x_j}$ and $J_x f$ denotes the Jacobian of f with respect to variable x. Sometimes, for the sake of simplicity, we may write $\phi_j(x_j)$ instead of $\phi_j(x_j, \vartheta_j)$.

AC. Instead of storing all intermediate activations produced during the forward pass, only a selected subset is kept in memory, which we denote by $x_{j_1}^{\mathrm{AC}}, \ldots, x_{j_l}^{\mathrm{AC}}, \ldots$. The missing activations are then recomputed on demand during backpropagation. In this way, activation checkpointing trades additional computation for reduced memory usage. More precisely, let $x_{j_l}^{\mathrm{AC}}$ be a stored checkpoint and introduce $\Phi_l = \phi_{j_{l+1}-1} \circ ... \circ \phi_{j_l}$. Then, for any layer index j such that $j_l < j \leq j_{l+1}$, the activation x_j is recovered by applying the corresponding forward operators from the checkpoint $x_{j_l}^{\mathrm{AC}}$ via a second forward pass to recompute $\Phi_l(x_{j_l}^{\mathrm{AC}}, \theta_l)$ with $\theta_l = (\vartheta_{j_l}, ..., \vartheta_{j_{l+1}-1})$, namely

$$x_{j+1}^{\mathrm{AC}} = \phi_j(x_j^{\mathrm{AC}}, \vartheta_j), \qquad j = j_l, \ldots, j_{l+1} - 1. \tag{Recompute forward}$$

Once the missing activations have been recomputed, the backward pass is applied as usual and those activations are progressively discarded, for $l = L, \ldots, 1$,

$$\delta_l = \left(J_{x_{j_l}} \Phi_l(x_{j_l}^{\mathrm{AC}}, \theta_l)\right)^\top \delta_{l+1}, \qquad \frac{\partial \mathcal{L}}{\partial \theta_l} = \left(J_{\theta_l} \Phi_l(x_{j_l}^{\mathrm{AC}}, \theta_l)\right)^\top \delta_{l+1}. \tag{Recompute backward}$$

The indices $j_1, \ldots, j_l, \ldots$ are typically selected via *selective AC*, whereby only a subset of layers retains additional intermediate quantities according to computational or memory considerations, most commonly those produced by linear layers. KEAP challenges this prevailing practice.

3.2 Toward KEAP

We now introduce a modification of AC for separable layers. We consider layer blocks $\Phi = \Phi_l$ (potentially, a single layer), that decompose, for token sequences $x = (x_t)_t$ so that $x_t \in \mathbb{R}^D$, as

$$\Phi(x, \theta) = \sum_{i=1}^{n} \varphi^i(x, \theta^i). \tag{Sep}$$

For instance, feed-forward layers (FFN) with hidden dimension n expand componentwise as

$$\mathrm{FFN}(x)_t = \sum_{i=1}^{n} W^2_{:i} \left(\mathrm{SiLU}((W^1 x_t)_i)\,(W^3 x_t)_i\right), \tag{1}$$

and likewise, a multi-head attention layer [52] with n heads can be written as a sum over heads

$$\mathrm{Attn}(x)_t = \sum_{i=1}^{n} W_i^o\, \mathbf{Att}(W_i^q x_t,\, W_i^k x,\, W_i^v x)\,. \tag{2}$$

Note that any layer satisfies this property when $n = 1$. Motivated by this separability property, we introduce sketching methods based on a masking vector, which may be either deterministic or stochastic. We define the corresponding sketched operator as

$$\Phi^\omega(x, \theta) = \sum_{i=1}^{n} \omega^i\, \varphi^i(x, \theta^i), \tag{Approx Sep}$$

where $\omega = (\omega^1, \ldots, \omega^n)$ is a potentially k-sparse vector, meaning that it has only k non-zero entries in expectation. A simple choice is to sample independently $z_i \sim \mathcal{B}(k/n)$ and set $\omega^i = \frac{n}{k} z_i$, so that $\mathbb{E}[\omega^i] = 1$ for $i = 1, \ldots, n$. In this case, the estimator is unbiased, since $\mathbb{E}[\Phi^\omega] = \Phi$. Its Jacobian estimator is also unbiased, as $\mathbb{E}[J\Phi^\omega] = J\Phi$ and, in expectation, only k elements are computed. However, unbiasedness is generally not preserved under composition. Indeed,

$$\mathbb{E}[J(\Phi_2^{\omega_2} \circ \Phi_1^{\omega_1})] \neq J(\Phi_2 \circ \Phi_1),$$

due to the dependence introduced by the forward pass, as highlighted by the blue term in

$$\mathbb{E}[J\Phi_2^{\omega_2}(\Phi_1^{\omega_1}(x), \theta_2) \circ J\Phi_1^{\omega_1}(x, \theta_1)] \neq J\Phi_2(\Phi_1(x), \theta_2) \circ J\Phi_1(x, \theta_1)\,.$$

This issue is resolved by combining this approximation with AC. Indeed, the first forward pass has already constructed $x_2^{\mathrm{AC}} = \Phi_1(x_1^{\mathrm{AC}})$ with $x_1^{\mathrm{AC}} = x$, so that the Jacobians $J\Phi_1^{\omega_1}, J\Phi_2^{\omega_2}$ are recomputed via a second forward pass, so that

$$\mathbb{E}[J\Phi_2^{\omega_2}(x_2^{\mathrm{AC}}, \theta_2) \circ J\Phi_1^{\omega_1}(x_1^{\mathrm{AC}}, \theta_1)] = J\Phi_2(\Phi_1(x), \theta_2) \circ J\Phi_1(x, \theta_1)\,.$$

This leads to the following generalization, whose proof can be found in App. A:

Proposition 1 (Unbiased gradient in the separable setting)**.** *Let $\mathcal{L} \circ \Phi_L \circ \cdots \circ \Phi_1$ be a model and assume that each block Φ_l admits a separable decomposition as in Eq. (Sep). For each block, let ω_l be a random mask such that $\mathbb{E}[\omega_l] = \mathbf{1}$, with masks sampled independently across blocks. Let $\{x_l^{\mathrm{AC}}\}_{l \le L}$ be the checkpointed activations obtained from the exact forward pass, as in Eq. (Recompute forward). Then, the sketched AC backward pass gives an unbiased estimator*

$$\mathbb{E}_\omega \left[\left(\widehat{\nabla}_{\theta_1}^{\omega,\mathrm{AC}} \mathcal{L}, \ldots, \widehat{\nabla}_{\theta_L}^{\omega,\mathrm{AC}} \mathcal{L} \right) \right] = \nabla_{(\theta_1,\ldots,\theta_L)} \mathcal{L}\,,$$

where we have for $1 \le l \le L$,

$$\widehat{\nabla}_{\theta_l}^{\omega,\mathrm{AC}} \mathcal{L} := \left(J_{\theta_l} \Phi_l^{\omega_l}(x_l^{\mathrm{AC}}, \theta_l) \right)^\top \left(J_x \Phi_{l+1}^{\omega_{l+1}}(x_{l+1}^{\mathrm{AC}}, \theta_{l+1}) \right)^\top \cdots \left(J_x \Phi_L^{\omega_L}(x_L^{\mathrm{AC}}, \theta_L) \right)^\top \nabla_{x_{L+1}} \mathcal{L}.$$

Building on this observation, we introduce three strategies within the KEAP framework:

Rand-$\frac{k}{n}$. A simple and naïve baseline is to sample $z \in \{0,1\}^n$ uniformly with exactly k ones, and set $\omega^i = \frac{n}{k} z_i$. In this case, we have $\mathbb{E}[J\Phi^\omega] = J\Phi$.

Opt-$\frac{k}{n}$. We optimize the sampling probabilities $\{p_i\}_{i \le n}$ under an expected budget of k selected components. Specifically, we sample independent Bernoulli variables $z_i \sim \mathcal{B}(p_i)$ with $\omega^i = \frac{z_i}{p_i}$, while enforcing a lower bound ε to avoid numerical instabilities. For samples $\{x^m\}_m$, it leads to

$$\begin{aligned} \min_{\{p_i\}_{i=1}^n} \quad & \sum_m \mathbb{E}_\omega \left\| \Phi(x^m) - \Phi^\omega(x^m) \right\|^2 \\ \text{s.t.} \quad & \sum_{i=1}^n p_i = k, \qquad \varepsilon \le p_i \le 1, \quad i = 1, \ldots, n. \end{aligned} \tag{*}$$

Top-$\frac{k}{n}$. For a collection of inputs $\{x^m\}_m$, define $\varphi^i := \{\varphi^i(x^m, \theta^i)\}_m$, so that $\|\varphi^i\|^2 = \sum_m \|\varphi^i(x^m, \theta^i)\|^2$. In this deterministic strategy, we keep only the k components with largest norm $\|\varphi^i\|$, and set the remaining ones to zero. Equivalently, $\omega^i = 1$ for the k indices corresponding to the largest values of $\|\varphi^i\|$, and $\omega^i = 0$ otherwise. This yields a biased but better approximation by preserving only the dominant contributions to $\Phi(x, \theta)$.

The following proposition solves the Opt-$\frac{k}{n}$ problem in Eq. (*); the proof is deferred to App. A.

Proposition 2 (Optimal Bernoulli sampling probabilities)**.** *Let $\omega^i = \frac{z_i}{p_i}$ and $z_i \sim \mathrm{Bernoulli}(p_i)$ independent, and let Φ and Φ^ω be defined as in Eq. (Sep) and Eq. (Approx Sep), respectively. Then the probabilities p_i minimizing Eq. (*) are given by*

$$p_i^\star = \min\left(1, \max\left(\varepsilon, \frac{\|\varphi^i\|}{\lambda}\right)\right),$$

where $\lambda > 0$ is chosen so that $\sum_{i=1}^n \min\left(1, \max\left(\varepsilon, \frac{\|\varphi^i\|}{\lambda}\right)\right) = k$. In particular, if the box constraints are inactive, i.e., $\varepsilon \le \frac{k\|\varphi^i\|}{\sum_{j=1}^n \|\varphi^j\|} \le 1$ for all i, then $p_i^\star = \frac{k\|\varphi^i\|}{\sum_{j=1}^n \|\varphi^j\|}$.

In fact, we can also demonstrate a pseudo-optimality of Top-$\frac{k}{n}$. For a mask ω, let

$$Q(\omega) \triangleq \mathbb{E}_\omega \sum_m \left\| \Phi(x^m) - \Phi^\omega(x^m) \right\|^2,$$

and introduce the coherence of the discarded components

$$\mu \triangleq \max_{\substack{i \ne j \\ i,j \notin \mathrm{Top}_k}} \left[\frac{\langle \varphi^i, \varphi^j \rangle}{\|\varphi^i\| \, \|\varphi^j\|} \right]_+ .$$

The following proposition shows that an arbitrarily small error for Opt-$\frac{k}{n}$ implies a correspondingly small error for Top-$\frac{k}{n}$, whereas the converse does not hold, thereby highlighting the advantage of Top-$\frac{k}{n}$.This proposition bounds the approximation error of a single block but does not guarantee unbiased gradients. Nevertheless, our ablations show that Top-$\frac{k}{n}$ maintains high gradient similarity in practice (Fig. 2).

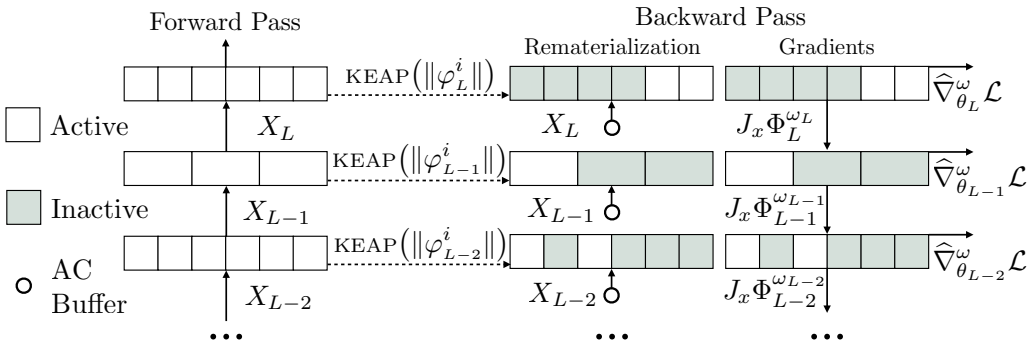


Figure 1: **High-level overview of KEAP.** During an initial forward pass, all components φ^i of a layer are computed. KEAP then selects the most important components according to their ℓ^2 norms. During rematerialization, only this selected fraction is recomputed and used for the backward pass.

Proposition 3 (Top-k versus optimal Bernoulli sampling)**.** *Let ω^{Top} and ω^{Opt} be the masks of Top-$\frac{k}{n}$ and Opt-$\frac{k}{n}$. Then*

$$Q(\omega^{\mathrm{Top}}) \;\le\; \big(1+(n-k-1)\mu\big)\sum_{i\notin \mathrm{Top}_k}\|\varphi^i\|^2 \;\le\; \big(1+(n-k-1)\mu\big)\,Q(\omega^{\mathrm{Opt}}).$$

In particular $Q(\omega^{\mathrm{Top}}) \le (n-k)\,Q(\omega^{\mathrm{Opt}})$ since $\mu \le 1$, and $Q(\omega^{\mathrm{Top}}) \le Q(\omega^{\mathrm{Opt}})$ whenever $\mu = 0$. Conversely, no reverse bound holds: for $n-k \ge 2$ there is no constant $C(n,k)$ such that $Q(\omega^{\mathrm{Opt}}) \le C(n,k)\,Q(\omega^{\mathrm{Top}})$.

3.3 Implementation of KEAP

We note that both Opt-$\frac{k}{n}$ and Top-$\frac{k}{n}$ require the statistic $\|\varphi^i\|$, which can be computed during the first forward pass at negligible additional cost, and even overlapped with other computations, which is an advantage of our method. The KEAP strategy is illustrated in Fig. 1. We next compare the FLOP and activation-memory costs of KEAP with vanilla backpropagation and standard activation-checkpointing strategies. These FLOP reductions provide a useful reference for computational savings. To better realize these gains in wall-clock time, we also introduce strategies that group our approximation across tokens and channels, enabling larger matrix multiplications; we evaluate these strategies separately in App. D.

Vanilla Backpropagation. We analyze the vanilla backward pass on a per-token basis using Eq. (1) and Eq. (2). For an FFN with hidden dimension n, the forward and backward costs are $3\times 2nD$ and $3\times 4nD$ FLOPs, respectively. In Llama-style models, one typically has $n=\frac{8}{3}D$ [17]. For multi-head attention with n heads of dimension d, each head projects tokens to a d-dimensional subspace, typically with $D=nd$. Per head and token, the forward and backward costs are $4\times 2dD$ and $4\times 4dD$ FLOPs, respectively. We neglect the cost of the attention operation itself compared to matrix multiplications, as it is limited in practice [11]. The activation memory stored during the forward pass depends on the backpropagation implementation, which is sometimes combined with AC-like strategies without being stated explicitly [38].

Vanilla AC. We now describe AC applied to a full FFN or attention layer, as defined in Eq. (1) or Eq. (2). For the FFN, AC introduces a computational overhead of $3\times 2nD$, while reducing the memory footprint to solely D. For the attention layer, the overhead is $4\times 2ndD$ and the memory footprint is also reduced to D.

Selective AC. Selective AC generally keeps all activations resulting from linear computations [30, 40]. Thus, for FFN and attention layers, the memory footprint per token becomes $2n+2D$ and $5D$, respectively, but no additional recomputation overhead is introduced. As above, we neglect the memory of the attention operation itself. This strategy is typically preferred over vanilla AC to reduce computational overhead.

KEAP. The goal of KEAP is to combine the memory benefit of Vanilla AC while recomputing only a fraction of the layer components. For FFN, the activation memory footprint is reduced to D, as for Vanilla AC, and the total cost per token becomes $3 \times 2nD + 3 \times 2kD + 3 \times 4kD = 6nD + 18\frac{k}{n}nD$. For attention, if we skip entire heads, the memory footprint is reduced to D and the total cost per token becomes $4 \times 2ndD + 4 \times 2kdD + 4 \times 4kdD = 8D^2 + 24\frac{k}{n}D^2$. This is done, for instance, using batch-level approximation granularity. However, this may increase variance of our gradient approximation. If masks are estimated at token-level granularity, the full keys and values must be stored, as shown in Eq. (2). Although they could be recomputed on the fly like the other activations, we instead store them to avoid additional computational overhead. Consequently, the total cost per token becomes $4 \times 2ndD + 2 \times 2kdD + 2 \times 4kdD + 2 \times 4ndD = 16D^2 + 12\frac{k}{n}D^2$ but the memory footprint per token increases to $D + 2nd = 3D$, still smaller than $5D$.

4 Numerical Experiments

4.1 Training setup

Architecture hyperparameters. We closely follow the tuned baselines described in [26]. All models use a Llama-3-style architecture with RoPE positional embeddings [49]. We use the Llama-2 7B tokenizer, with vocabulary size 32k because it incurs a smaller memory overhead compared to the Llama-3 tokenizer. We did not observe any significant discrepancy relative to the Llama-3 tokenizer family. We consider two model configurations. The first is a 134M-parameter model with hidden size 768, 12 layers, and 12 attention heads, which is used for ablation studies. The second is a 488M-parameter model with hidden size 1280, 20 layers, and 20 attention heads, which is used for the main numerical results in Sec. 4.2. Indeed, this model scale provides a practical compromise between computational cost and empirical relevance, making it suitable for assessing the potential scalability of our method [22].

Training hyperparameters. We follow standard training practice and tune the baselines using a linear-warmup cosine-decay learning-rate schedule [34]. Warmup is applied during the first 5% of training steps, followed by cosine decay over the remaining steps, with the learning rate decaying to zero. The tuning sweep covers learning rates over three logarithmic scales, $\{10^{-2}, 10^{-3}, 10^{-4}\}$, with linear multipliers $\{1, 5, 7\}$, and weight-decay values ranging from 10^{-1} to 10^{-3} [35]. This sweep allows us to cross-validate the best learning rate and weight decay. We use the Muon optimizer [27], which we found to provide good stability, with its standard settings: momentum 0.95 with Nesterov momentum. We also clip the gradient norm to 1.0. All models are trained on FineWeb [39], which contains approximately a total of 15T tokens. We use batch size 512 and sequence length 2048, corresponding to $1{,}048{,}576$ tokens per training step. Dense baselines are trained to Chinchilla-optimal token counts [22], corresponding to 2555 and 9307 training steps for the 134M and 488M models, or 2.68B and 9.76B processed tokens, respectively. In practice, the number of attention heads is small ($n \sim 10^2$) compared to the FFN hidden dimension ($n \sim 10^3$), so we apply the clipping strategy of Prop. 2 only to FFN layers. Numerically, setting $\epsilon = 0.06$ gives good performance without loss explosion.

Evaluation protocol. We report corresponding perplexity (PPL) on a held-out FineWeb evaluation set containing 25M tokens (disjoint from the training set), and we also report the evaluation loss in App. E. To evaluate the impact of our procedure, we compare it against two baselines: a reference model and a compute-matched AC baseline. AC does not change the model's backward computation, but it increases the FLOPs of rematerialized modules by 33.3%. Therefore, the compute-matched AC baseline uses a smaller model whose total training FLOPs match the budget considered in App. B. Specifically, when AC is applied only to FFN layers, we reduce the FFN hidden dimension in the baseline. When AC is applied only to attention layers, we reduce the number of attention heads n. When KEAP is applied to both FFN and attention layers, we apply both reductions while keeping the model width D fixed. An alternative would be to apply a global model width multiplier [22, 51, 55], but we did not observe significant differences with this approach. This reduction leads to slight FLOP differences for attention layers depending on whether token- or batch-level granularity is used; we discuss this further in App. B. We also tune the learning rate for the compute-matched AC baselines and find that the selected value remains unchanged relative to the dense baseline. For KEAP, we retain the dense baseline's learning rate without additional tuning,

Method	$\frac{k}{n}$	AC-FLOPs	PPL	ΔPPL
AC applied on FFN layers				
Llama-488M	1	1.33	12.03	–
Llama-419M	1	1	12.44	3.41%
Rand	0.66	1	12.84	6.73%
	0.33	0.66	14.70	22.19%
Opt	0.66	1	12.58	4.57%
	0.33	0.66	13.66	13.55%
Top	0.66	1	**12.23**	**1.66%**
	0.33	0.66	12.46	3.57%
AC applied on Attention layers				
Llama-488M	1	1.33	12.03	–
Llama-468M	1	1	**12.17**	**1.16%**
Rand	0.66	1	12.55	4.32%
	0.33	0.83	13.43	11.64%
Opt	0.66	1	12.27	2.00%
	0.33	0.83	12.74	5.90%
Top	0.66	1	**12.17**	**1.16%**
	0.33	0.83	12.45	3.49%
AC applied on Attention+FFN layers				
Llama-488M	1	1.33	12.03	–
Llama-400M	1	1	12.56	4.41%
Rand	0.66	1	13.47	11.97%
	0.33	0.72	17.91	48.88%
Opt	0.66	1	12.81	6.48%
	0.33	0.72	14.82	23.19%
Top	0.66	1	**12.33**	**2.49%**
	0.33	0.72	12.90	7.23%

(a) Token-level granularity

Method	$\frac{k}{n}$	AC-FLOPs	PPL	ΔPPL
AC applied on FFN layers				
Llama-488M	1	1.33	12.03	–
Llama-419M	1	1	**12.44**	**3.41%**
Rand	0.66	1	13.06	8.56%
	0.33	0.66	15.55	29.26%
Opt	0.66	1	12.89	7.15%
	0.33	0.66	14.30	18.87%
Top	0.66	1	12.61	4.82%
	0.33	0.66	13.40	11.39%
AC applied on Attention layers				
Llama-488M	1	1.33	12.03	–
Llama-456M	1	1	**12.22**	**1.58%**
Rand	0.66	1	12.97	7.81%
	0.33	0.66	14.47	20.28%
Opt	0.66	1	12.79	6.32%
	0.33	0.66	13.77	14.46%
Top	0.66	1	12.35	2.66%
	0.33	0.66	12.54	4.24%
AC applied on Attention+FFN layers				
Llama-488M	1	1.33	12.03	–
Llama-387M	1	1	**12.59**	**4.66%**
Rand	0.66	1	14.25	18.45%
	0.33	0.66	20.23	68.16%
Opt	0.66	1	13.80	14.71%
	0.33	0.66	17.06	41.81%
Top	0.66	1	12.85	6.82%
	0.33	0.66	14.14	17.54%

(b) Batch-level granularity

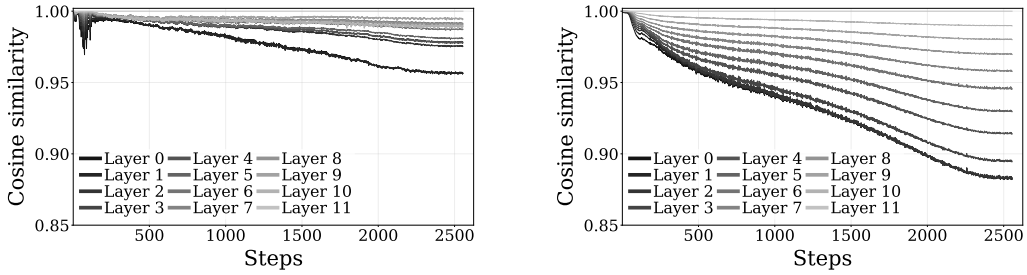
Table 1: **KEAP performance**. We report the perplexity of a Llama-488M model under different AC strategies, using per-token (**a**) and per-batch (**b**) granularity during re-computation as described in Sec. 3.3. Δ PPL is the relative perplexity degradation compared to the vanilla baseline. AC-FLOPs denotes the analytic FLOP cost of the targeted FFN and attention-projection computations relative to standard backpropagation for the same modules. (details can be found in App. B).

to assess how closely approximate rematerialization reproduces training with exact gradients under the same optimization settings. All baselines are trained on the same number of tokens.

Compute environment. All the main experiments use BF16 mixed precision on 4 NVIDIA H100 or A100 GPUs, which improves speed without observable performance degradation. All experiments are implemented using the TorchTitan [31] large-scale training framework.

4.2 Results

Tab. 1 presents the main results for KEAP on models that are at most 488M parameters, for both token- and batch-level granularities. Vanilla denotes standard AC, while Top-, Opt-, and Rand-$\frac{k}{n}$ are the selective variants described in Sec. 3. We report FLOPs relative to vanilla backpropagation. These ratios provide a hardware-independent reference for the amount of arithmetic performed, but do not imply proportional wall-clock speedups, which also depend on the selection granularity and kernel efficiency discussed in App. D. For example, a Llama-488M model with standard AC has an AC-FLOP ratio of 1.33 due to the second forward recomputation. We evaluate KEAP with $\frac{k}{n} \in \left\{\frac{1}{3}, \frac{2}{3}\right\}$. We further evaluate KEAP with token-level Top-$\frac{2}{3}$ selection applied to FFN layers on



(a) Attention

(b) FFN

Figure 2: **Average cosine similarity between exact and KEAP activation gradients** for a Llama-134M model Φ, using KEAP Top-$\frac{1}{3}$ with token-level granularity. These plots show that gradients remain well aligned across layers, with only minor degradation from error propagation with depth.

a 953M-parameter Llama-style model using the same training protocol. We obtain a perplexity of 9.959 compared to 9.858 for the dense baseline (a 1.02% increase; App. C). On the same model, batch-level FFN KEAP reduces training-step time by 18% and 26% with Top-$\frac{2}{3}$ and Top-$\frac{1}{3}$ selection, respectively, relative to Full AC on four NVIDIA H100 GPUs (App. D).

Rand-$\frac{k}{n}$ vs. Opt-$\frac{k}{n}$ vs. Top-$\frac{k}{n}$. As discussed in Sec. 3.2, Opt-$\frac{k}{n}$ is constructed to minimize the mean-squared approximation error, and therefore consistently improves over Rand-$\frac{k}{n}$ method. Nevertheless, for a fixed $\frac{k}{n}$, our Top-$\frac{k}{n}$ approximation achieves substantially better performance than both baselines. Despite being biased, Top-k selection rules are actually well known, for instance in gradient compression, to outperform random or variance-optimal baselines in practice [6]. In particular, relative to the standard Llama-488M, token-level Top-$\frac{2}{3}$ incurs at most a 2.5% increase in perplexity in every setting.

Smaller models vs. Top-$\frac{k}{n}$. The smaller models provide a reference for allocating the same analytic FLOP budget to reduced model capacity rather than approximate rematerialization. At token-level granularity, compute-matched smaller models perform worse than or the same as KEAP Top-$\frac{2}{3}$, suggesting that KEAP applied to the original model can be preferable to reducing the corresponding FFN or attention dimensions in terms of accuracy. In particular, for attention, the accuracy of the Llama-468M model is the same as KEAP. This is unsurprising: reducing attention removes few parameters and thus leaves little room for KEAP to improve over it. For FFN, KEAP outperforms the FLOP-matched smaller model. The ablations below further show that Top-$\frac{k}{n}$ preserves a high-quality gradient approximation. Finally, batch-level granularity performs poorly, as discussed below.

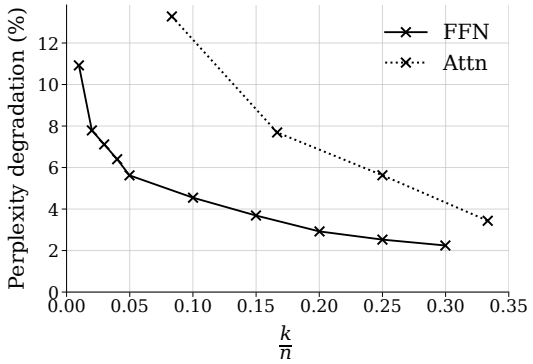


Figure 3: **Perplexity degradation versus $\frac{k}{n}$** for KEAP Top-$\frac{k}{n}$ with token-level granularity applied on FFN and Attn for a Llama-134M model.

Token-level vs. batch-level granularity. Tab. 1a compared to Tab. 1b shows that token-level gradient approximations consistently outperform batch-level ones, and batch-level approximations perform worse than a smaller model. This behavior is expected: as discussed in Sec. 3.3, token-level granularity yields a more accurate gradient approximation in expectation at the cost of worse (but expected) hardware utilisation we discuss later in App. D.

FFN vs. Attention Rematerialization. Applying KEAP to FFN layers generally performs better than applying it to attention layers, or to both modules simultaneously. This is unsurprising: viewing the FFN as a one-hidden-layer network (Eq. (1)), the amplitudes of its gradient atoms are largely

predictable from the forward pass. By contrast, this connection is less direct for attention, which has a more nonlinear gradient structure. Nevertheless, applying KEAP jointly to FFN and attention layers preserves a useful approximation of the dense gradient while retaining a substantially lower activation-memory footprint than Selective AC.

4.3 Ablation experiments

We now ablate token-level KEAP Top-$\frac{k}{n}$ on a smaller 134M Llama-style model.

Cosine similarity between approximate and exact gradients. In Fig. 2, we report the cosine similarity between the exact activation gradients and the KEAP activation gradients for both FFN and attention layers, while training with the approximate gradient $\nabla\Phi^{\omega}$ from KEAP. This allows us to verify that the resulting optimization dynamics not only achieve good performance, but also remain close to those of standard training. We use the Top-$\frac{k}{n}$ approximation with $\frac{k}{n} = \frac{1}{3}$ on all attention and FFN layers. Cosine similarities are computed at each layer, averaged over all tokens in a batch, and tracked throughout training. The fact that they remain above 0.85 over the entire run indicates that the sparse reconstruction preserves the direction of the dense local gradient per step. This is notable, since ω is constructed solely from information available during the forward pass. Finally, we observe that cosine similarity tends to decrease with depth, likely due to a progressive degradation of the gradient approximation across layers. This behavior is expected and does not appear to hinder training.

Ablation on $\frac{k}{n}$. In Fig. 3, we investigate how performance degrades as the retained fraction $\frac{k}{n}$ varies, when applying KEAP either exclusively to FFN layers or exclusively to attention layers. We terminate each sweep once the performance drop exceeds 10%. The results show that, due to the adaptivity of the selection rule, retaining only a small fraction of separable components can still preserve strong performance. The useful components appear highly concentrated: retaining only 5% of the FFN coordinates still keeps the performance drop below 5%.

5 Conclusion

In this work, we introduce KEAP, a method for approximate rematerialization that exploits exact forward-pass checkpoints to preserve high-quality gradient estimates. Using information collected during the initial forward pass, KEAP recomputes only a subset of the original FFN and attention components during rematerialization. On Llama-style models with up to 488M parameters, retaining two thirds of these components limits the perplexity increase to at most 2.5% while maintaining the activation-memory benefits of full Activation Checkpointing. At token-level granularity, KEAP also achieves comparable or better model quality than smaller FLOP-matched models, indicating that approximating rematerialization can preserve model capacity more effectively than reducing model size. Future work will focus on further optimizing finer-grained implementations to more fully translate the FLOP reductions into wall-clock speedups on modern accelerators.

Acknowledgements

This work was supported by funding managed by the French National Research Agency (ANR) under the France 2030 program through the DAIMOS project (grant ANR-25-EXNU-0002) and PEPR IA (grant SHARP ANR-23-PEIA-0008). Part of this work was granted access to the JeanZay HPC/AI resources of IDRIS under the allocation AD011015884R1, AD011017481, AD011016641, AD011017661 and AD011017766.

References

[1] Adrien Aguila-Multner, Olivier Beaumont, Lionel Eyraud-Dubois, and Julia Gusak. Optimized forward-backward rematerialization for memory-efficient pipeline parallel training, 2025. URL https://inria.hal.science/hal-05151601.

[2] Killian Bakong, Laurent Massoulié, Edouard Oyallon, and Kevin Scaman. Unbiased approximate vector-jacobian products for efficient backpropagation, 2026. URL https://arxiv.org/abs/2602.14701.

[3] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153):1–43, 2018.

[4] Olivier Beaumont, Lionel Eyraud-Dubois, Julien Herrmann, Alexis Joly, and Alena Shilova. Optimal re-materialization strategies for heterogeneous chains: How to train deep neural networks with limited memory. *ACM Trans. Math. Softw.*, 50(2), June 2024. ISSN 0098-3500. doi: 10.1145/3648633. URL https://doi.org/10.1145/3648633.

[5] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of CNNs. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[6] Aleksandr Beznosikov, Samuel Horváth, Peter Richtárik, and Mher Safaryan. On biased compression for distributed learning. *Journal of Machine Learning Research*, 24(276):1–50, 2023. URL http://jmlr.org/papers/v24/21-1548.html.

[7] Ayan Chakrabarti and Benjamin Moseley. Backprop with approximate activations for memory-efficient network training. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/30c8e1ca872524fbf7ea5c519ca397ee-Paper.pdf.

[8] Ping Chen, Zhuohong Deng, Ping Li, Shuibing He, Hongzi Zhu, Yi Zheng, Zhefeng Wang, Baoxing Huai, and Minyi Guo. Adacc: An adaptive framework unifying compression and activation recomputation for LLM training, 2025. URL https://arxiv.org/abs/2508.00806.

[9] Brian Chmiel, Itay Hubara, Ron Banner, and Daniel Soudry. Minimum variance unbiased N:M sparsity for the neural gradients. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vuD2xEtxZcj.

[10] Amit Daniely, Nevena Lazic, Yoram Singer, and Kunal Talwar. Short and deep: Sketching and neural networks. In *International Conference on Learning Representations Workshop*, 2017. URL https://openreview.net/pdf?id=r1br_2Kge.

[11] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: fast and memory-efficient exact attention with IO-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

[12] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

[13] Epoch AI. Exponential growth of parameters in notable AI systems, 2026. URL https://ourworldindata.org/grapher/exponential-growth-of-parameters-in-notable-ai-systems.

[14] Louis Fournier, Adel Nabli, Masih Aminbeidokhti, Marco Pedersoli, Eugene Belilovsky, and Edouard Oyallon. WASH: Train your ensemble with communication-efficient weight shuffling, then average. In *OPT 2024: Optimization for Machine Learning*, 2024.

[15] Nidham Gazagnadou, Mark Ibrahim, and Robert M. Gower. RidgeSketch: A fast sketching based solver for large scale ridge regression. *SIAM Journal on Matrix Analysis and Applications*, 43(3):1440–1468, 2022. doi: 10.1137/21M1422963. URL https://doi.org/10.1137/21M1422963.

[16] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf.

[17] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[18] Scott Gray, Alec Radford, and Diederik P. Kingma. GPU kernels for block-sparse weights. Technical report, OpenAI, 2017. URL https://cdn.openai.com/blocksparse/blocksparsepaper.pdf.

[19] Andreas Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software*, 1(1):35–54, 1992. doi: 10.1080/10556789208805505. URL https://doi.org/10.1080/10556789208805505.

[20] Andreas Griewank and Andrea Walther. Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Trans. Math. Softw.*, 26(1):19–45, March 2000. ISSN 0098-3500. doi: 10.1145/347837.347846. URL https://doi.org/10.1145/347837.347846.

[21] Julia Gusak, Xunyi Zhao, Théotime Le Hellard, Zhe Li, Lionel Eyraud-Dubois, and Olivier Beaumont. HiRemate: Hierarchical approach for efficient re-materialization of neural networks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 21418–21443. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/gusak25a.html.

[22] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 30016–30030. Curran Associates, Inc., 2022. doi: 10.52202/068431-2176. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

[23] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and Zhifeng Chen. GPipe: Efficient training of giant neural networks using pipeline parallelism. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/093f65e080a295f8076b1c5722a46aa2-Paper.pdf.

[24] Jörn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. i-RevNet: Deep invertible networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJsjkMb0Z.

[25] Paul Janson, Edouard Oyallon, and Eugene Belilovsky. Learned subspace compression for communication-efficient pipeline parallelism. *arXiv preprint arXiv:2606.05484*, 2026.

[26] Paul Janson, Edouard Oyallon, and Eugene Belilovsky. Stabilizing native low-rank LLM pretraining. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 50830–50852. PMLR, 2026. URL https://proceedings.mlr.press/v306/janson26a.html.

[27] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

[28] Taebum Kim, Hyoungjoo Kim, Gyeong-In Yu, and Byung-Gon Chun. BPipe: Memory-balanced pipeline parallelism for training large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 16639–16653. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kim23l.html.

[29] Boao Kong, Xu Huang, Yuqi Xu, Yixuan Liang, Bin Wang, and Kun Yuan. Clapping: Removing per-sample storage for pipeline parallel distributed optimization with communication compression, 2026. URL https://arxiv.org/abs/2509.19029.

[30] Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. In D. Song, M. Carbin, and T. Chen, editors, *Proceedings of Machine Learning and Systems*, volume 5, pages 341–353. Curran Associates, Inc., 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/file/80083951326cf5b35e5100260d64ed81-Paper-mlsys2023.pdf.

[31] Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, Sanket Purandare, Gokul Nadathur, and Stratos Idreos. TorchTitan: One-stop PyTorch native solution for production ready LLM pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=SFN6Wm7YBI.

[32] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT*, 16(2):146–160, June 1976. ISSN 0006-3835. doi: 10.1007/BF01931367. URL https://doi.org/10.1007/BF01931367.

[33] Xiaoxuan Liu, Lianmin Zheng, Dequan Wang, Yukuo Cen, Weize Chen, Xu Han, Jianfei Chen, Zhiyuan Liu, Jie Tang, Joey Gonzalez, Michael Mahoney, and Alvin Cheung. GACT: Activation compressed training for generic network architectures. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 14139–14152. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/liu22v.html.

[34] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

[35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[36] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

[37] Adel Nabli, Louis Fournier, Pierre ERBACHER, Louis Serrano, Eugene Belilovsky, and Edouard Oyallon. ACCO: Accumulate while you communicate for communication-overlapped sharded LLM training. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 69278–69305. Curran Associates, Inc., 2025. doi: 10.52202/085713-2329. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/6454dcd80b5373daaa97e53ce32c78a1-Paper-Conference.pdf.

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[39] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

[40] PyTorch Foundation. Current and new activation checkpointing techniques in PyTorch, March 2025. URL https://pytorch.org/blog/activation-checkpointing-techniques/.

[41] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020. ISBN 9781728199986.

[42] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. ZeRO-Offload: Democratizing Billion-Scale model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564. USENIX Association, July 2021. ISBN 978-1-939133-23-6. URL https://www.usenix.org/conference/atc21/presentation/ren-jie.

[43] Minsoo Rhu, Natalia Gimelshein, Jason Clemons, Arslan Zulfiqar, and Stephen W. Keckler. vDNN: virtualized deep neural networks for scalable, memory-efficient neural network design. In *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-49. IEEE Press, 2016.

[44] Stephane Rivaud, Louis Fournier, Thomas Pumir, Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. PETRA: Parallel end-to-end training with reversible architectures. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=0fhzSFsGUT.

[45] Amir Sarfi, Benjamin Thérien, Joel Lidin, and Eugene Belilovsky. Communication efficient LLM pre-training with SparseLoCo. *arXiv preprint arXiv:2508.15706*, 2025.

[46] Yara Shamshoum, Nitzan Hodos, Yuval Sieradzki, and Assaf Schuster. CompAct: Compressed activations for memory-efficient LLM training. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1511–1524, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.71. URL https://aclanthology.org/2025.naacl-long.71/.

[47] Vaibhav Singh, Zafir Khalid, Pietro Cagnasso, Edouard Oyallon, and Eugene Belilovsky. Model parallelism with subnetwork data parallelism. *arXiv preprint arXiv:2507.09029*, 2025.

[48] Emanuele Del Sozzo, Martin Fleming, Kenneth Flamm, and Neil Thompson. How much progress has there been in NVIDIA datacenter GPUs?, 2026. URL https://arxiv.org/abs/2601.20115.

[49] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

[50] Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meProp: Sparsified back propagation for accelerated deep learning with reduced overfitting. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3299–3308. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/sun17c.html.

[51] Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pretraining and finetuning transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=f2OYVDyfIB.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[53] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1–2):1–157, October 2014. ISSN 1551-305X. doi: 10.1561/0400000060. URL https://doi.org/10.1561/0400000060.

[54] Chejian Xu, Wei Ping, Peng Xu, Zihan Liu, Boxin Wang, Mohammad Shoeybi, Bo Li, and Bryan Catanzaro. From 128K to 4M: Efficient training of ultra-long context large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 13122–13133, 2026.

[55] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17084–17097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8df7c2e3c3c3be098ef7b382bd2c37ba-Paper.pdf.

[56] Minghan Yang, Dong Xu, Zaiwen Wen, Mengyun Chen, and Pengxiang Xu. Sketch-based empirical natural gradient methods for deep learning. *J. Sci. Comput.*, 92(3), September 2022. ISSN 0885-7474. doi: 10.1007/s10915-022-01911-x. URL https://doi.org/10.1007/s10915-022-01911-x.

A Proofs

We provide the proofs of the three main theoretical statements. First, we prove that the stochastic version of KEAP gives an unbiased estimator of the parameter gradient over a checkpointed segment. The key point is that each sketched local Jacobian is evaluated at an exact checkpointed activation, so the masks are independent across blocks after conditioning on the forward pass. Second, we derive the optimal Bernoulli sampling probabilities used by Opt-$\frac{k}{n}$ by minimizing the mean-squared sketching error under a fixed expected budget by deriving the KKT condition of the corresponding optimization problem. Finally, we demonstrate that Top-$\frac{k}{n}$ offers an approximation advantage over Opt-$\frac{k}{n}$.

A.1 Proof of Proposition 1

Proposition (Unbiased gradient in the separable setting)**.** *Let $\mathcal{L}\circ\Phi_L\circ\cdots\circ\Phi_1$ be a model and assume that each block Φ_l admits a separable decomposition as in Eq. (Sep). For each block, let ω_l be a random mask such that $\mathbb{E}[\omega_l]=\mathbf{1}$, with masks sampled independently across blocks. Let $\{x_l^{\mathrm{AC}}\}_{l\le L}$ be the checkpointed activations obtained from the exact forward pass, as in Eq. (Recompute forward). Then, the sketched AC backward pass gives an unbiased estimator*

$$\mathbb{E}_\omega\left[\left(\widehat{\nabla}_{\theta_1}^{\omega,\mathrm{AC}}\mathcal{L},\dots,\widehat{\nabla}_{\theta_L}^{\omega,\mathrm{AC}}\mathcal{L}\right)\right]=\nabla_{(\theta_1,\dots,\theta_L)}\mathcal{L}\,,$$

where we have for $1\le l\le L$,

$$\widehat{\nabla}_{\theta_l}^{\omega,\mathrm{AC}}\mathcal{L}:=\left(J_{\theta_l}\Phi_l^{\omega_l}(x_l^{\mathrm{AC}},\theta_l)\right)^\top\left(J_x\Phi_{l+1}^{\omega_{l+1}}(x_{l+1}^{\mathrm{AC}},\theta_{l+1})\right)^\top\cdots\left(J_x\Phi_L^{\omega_L}(x_L^{\mathrm{AC}},\theta_L)\right)^\top\nabla_{x_{L+1}}\mathcal{L}.$$

Proof. For each block l, the sketched operator is

$$\Phi_l^{\omega_l}(x,\theta_l)=\sum_i\omega_l^i\varphi_l^i(x,\theta_l^i),$$

with $\mathbb{E}[\omega_l]=\mathbf{1}$. Hence, for fixed input x_l^{AC}, both the parameter and input Jacobians are unbiased:

$$\mathbb{E}_{\omega_l}\left[J_{\theta_l}\Phi_l^{\omega_l}(x_l^{\mathrm{AC}},\theta_l)\right]=J_{\theta_l}\Phi_l(x_l^{\mathrm{AC}},\theta_l),\qquad\mathbb{E}_{\omega_l}\left[J_x\Phi_l^{\omega_l}(x_l^{\mathrm{AC}},\theta_l)\right]=J_x\Phi_l(x_l^{\mathrm{AC}},\theta_l).$$

The key point is that the activations x_l^{AC} are obtained from the exact forward pass and therefore do not depend on the masks ω_l. Since the masks are independent across blocks, the expectation of the product of the sketched Jacobians factorizes. Thus, for every l,

$$\begin{aligned}\mathbb{E}_\omega\left[\widehat{\nabla}_{\theta_l}^{\omega,\mathrm{AC}}\mathcal{L}\right]&=\mathbb{E}_\omega\left[(J_{\theta_l}\Phi_l^{\omega_l})^\top\left(J_x\Phi_{l+1}^{\omega_{l+1}}\right)^\top\cdots(J_x\Phi_L^{\omega_L})^\top\nabla_{x_{L+1}}\mathcal{L}\right]\\&=(J_{\theta_l}\Phi_l)^\top(J_x\Phi_{l+1})^\top\cdots(J_x\Phi_L)^\top\nabla_{x_{L+1}}\mathcal{L}.\end{aligned}$$

All Jacobians above are evaluated at the exact checkpointed activations x_l^{AC} and parameters θ_l. The final expression is exactly the chain-rule expression for the true gradient $\nabla_{\theta_l}\mathcal{L}$. Therefore,

$$\mathbb{E}_\omega\left[\widehat{\nabla}_{\theta_l}^{\omega,\mathrm{AC}}\mathcal{L}\right]=\nabla_{\theta_l}\mathcal{L},\qquad l=1,\dots,L,$$

which proves the claimed unbiasedness of the full gradient estimator. □

A.2 Proof of Proposition 2

Proposition (Optimal Bernoulli sampling probabilities)**.** *Let $\Phi(x)=\sum_{i=1}^n\varphi^i(x,\theta^i)$ and let*

$$\Phi^\omega(x)=\sum_{i=1}^n\omega^i\varphi^i(x,\theta^i),\qquad\omega^i=\frac{z_i}{p_i},\qquad z_i\sim\mathrm{Bernoulli}(p_i),$$

where the z_i are independent. For a collection of inputs $\{x^m\}_m$, define $\|\varphi^i\|^2:=\sum_m\|\varphi^i(x^m,\theta^i)\|^2$. Then the probabilities p_i minimizing Eq. (*) are given by

$$p_i^\star=\min\left(1,\max\left(\varepsilon,\frac{\|\varphi^i\|}{\sqrt{\lambda}}\right)\right),$$

where $\lambda > 0$ is chosen so that

$$\sum_{i=1}^{n} \min\Big(1, \max\Big(\varepsilon, \frac{\|\varphi^i\|}{\sqrt{\lambda}}\Big)\Big) = k.$$

In particular, if the box constraints are inactive, i.e.,$\varepsilon \le \frac{k\|\varphi^i\|}{\sum_{j=1}^{n}\|\varphi^j\|} \le 1$ for all i, then

$$p_i^\star = \frac{k\|\varphi^i\|}{\sum_{j=1}^{n}\|\varphi^j\|}.$$

Proof. Using $\omega^i = z_i/p_i$ with $z_i \sim \mathcal{B}(p_i)$, we have

$$\mathbb{E}[\omega^i] = 1, \qquad \mathrm{Var}(\omega^i) = \frac{1-p_i}{p_i}.$$

Writing $\|\varphi^i\|^2 = \sum_m \|\varphi^i(x^m, \theta^i)\|^2$, independence gives

$$\sum_m \mathbb{E}_\omega \|\Phi(x^m) - \Phi^\omega(x^m)\|^2 = \sum_{i=1}^{n} \frac{1-p_i}{p_i}\,\|\varphi^i\|^2.$$

Hence the optimization problem is

$$\min_{\{p_i\}_i} \sum_{i=1}^{n} \frac{\|\varphi^i\|^2}{p_i} \qquad \text{s.t.} \qquad \sum_{i=1}^{n} p_i = k, \quad \varepsilon \le p_i \le 1.$$

The Lagrangian is

$$\mathcal{L}(p,\lambda) = \sum_{i=1}^{n} \frac{\|\varphi^i\|^2}{p_i} + \lambda\Big(\sum_{i=1}^{n} p_i - k\Big).$$

Ignoring the box constraints for a moment, the stationarity condition gives

$$-\frac{\|\varphi^i\|^2}{p_i^2} + \lambda = 0, \qquad \text{hence} \qquad p_i = \frac{\|\varphi^i\|}{\sqrt{\lambda}}.$$

Therefore, with the constraints $\varepsilon \le p_i \le 1$, the KKT conditions yield

$$p_i^\star = \min\Big(1, \max\Big(\varepsilon, \frac{\|\varphi^i\|}{\sqrt{\lambda}}\Big)\Big),$$

where $\lambda > 0$ is chosen so that

$$\sum_{i=1}^{n} \min\Big(1, \max\Big(\varepsilon, \frac{\|\varphi^i\|}{\sqrt{\lambda}}\Big)\Big) = k.$$

In the unconstrained regime, namely when

$$\varepsilon \le \frac{k\,\|\varphi^i\|}{\sum_{j=1}^{n}\|\varphi^j\|} \le 1 \qquad \text{for all } i,$$

this reduces to

$$p_i^\star = \frac{k\,\|\varphi^i\|}{\sum_{j=1}^{n}\|\varphi^j\|}.$$

□

A.3 Proof of Proposition 3

Proposition (Top-k versus optimal Bernoulli sampling)**.** *Let ω^{Top} and ω^{Opt} be the masks of Top-$\frac{k}{n}$ and Opt-$\frac{k}{n}$. Then*

$$Q(\omega^{\mathrm{Top}}) \;\le\; \big(1+(n-k-1)\mu\big)\sum_{i\notin \mathrm{Top}_k}\|\varphi^i\|^2 \;\le\; \big(1+(n-k-1)\mu\big)\,Q(\omega^{\mathrm{Opt}}).$$

In particular $Q(\omega^{\mathrm{Top}}) \le (n-k)\,Q(\omega^{\mathrm{Opt}})$ since $\mu \le 1$, and $Q(\omega^{\mathrm{Top}}) \le Q(\omega^{\mathrm{Opt}})$ whenever $\mu = 0$. Conversely, no reverse bound holds: for $n-k \ge 2$ there is no constant $C(n,k)$ such that $Q(\omega^{\mathrm{Opt}}) \le C(n,k)\,Q(\omega^{\mathrm{Top}})$.

Proof. Since ω^{Top} is deterministic and discards exactly the components outside Top_k, expanding the squared norm gives

$$Q(\omega^{\mathrm{Top}}) = \Big\|\sum_{i\notin \mathrm{Top}_k}\varphi^i\Big\|^2 = \sum_{i\notin \mathrm{Top}_k}\|\varphi^i\|^2 + 2\sum_{\substack{i<j\\ i,j\notin \mathrm{Top}_k}}\langle\varphi^i,\varphi^j\rangle \;\le\; \sum_{i\notin \mathrm{Top}_k}\|\varphi^i\|^2 + 2\mu\sum_{\substack{i<j\\ i,j\notin \mathrm{Top}_k}}\|\varphi^i\|\,\|\varphi^j\|,$$

by definition of μ. Young's inequality $2\|\varphi^i\|\|\varphi^j\| \le \|\varphi^i\|^2 + \|\varphi^j\|^2$, summed over the unordered pairs outside Top_k, makes each term appear $n-k-1$ times and yields the first inequality.

For the second, recall from the proof of Prop. 2 that $Q(\omega) = \sum_i \frac{1-p_i}{p_i}\|\varphi^i\|^2$. Any feasible p satisfies $p_i \le 1$, so $\frac{1-p_i}{p_i} \ge 1-p_i$ and

$$Q(\omega) \;\ge\; \sum_{i=1}^n \|\varphi^i\|^2 - \sum_{i=1}^n p_i\|\varphi^i\|^2.$$

The linear functional $p \mapsto \sum_i p_i\|\varphi^i\|^2$ over $\{0 \le p_i \le 1,\ \sum_i p_i = k\}$ is maximized at the vertex $p = \mathbf{1}_{\mathrm{Top}_k}$, hence $Q(\omega) \ge \sum_{i\notin \mathrm{Top}_k}\|\varphi^i\|^2$ for every feasible p, and in particular for ω^{Opt}. Finally $\mu \le 1$ by Cauchy–Schwarz, and $\mu = 0$ under non-positive inner products.

For the last claim, let the $n-k$ components outside Top_k be unit vectors summing to zero and the k others have norm $1+\epsilon$, so that Top-$\frac{k}{n}$ indeed retains the latter. Then $Q(\omega^{\mathrm{Top}}) = 0$, while $\sum_i p_i = k < n$ forces $p_i < 1$ for some i and hence $Q(\omega^{\mathrm{Opt}}) > 0$. □

B AC-FLOP derivations

Let n_{FFN} denote the FFN hidden size and $n_{\mathrm{Attention}}$ the number of attention heads. From Sec. 3.2, the FLOPs for the FFN and attention layers without AC are

$$18n_{\mathrm{FFN}}D + 24D^2. \tag{3}$$

Applying KEAP to both FFN and attention layers at token-level granularity gives

$$\left(6n_{\mathrm{FFN}}D + 18\frac{k_{\mathrm{FFN}}}{n_{\mathrm{FFN}}}n_{\mathrm{FFN}}D\right) + \left(16D^2 + 12\frac{k_{\mathrm{Attention}}}{n_{\mathrm{Attention}}}D^2\right). \tag{4}$$

Let $p = \frac{k_{\mathrm{Attention}}}{n_{\mathrm{Attention}}} = \frac{k_{\mathrm{FFN}}}{n_{\mathrm{FFN}}}$. The FLOP ratio between KEAP and the vanilla backward pass is then

$$\frac{6n_{\mathrm{FFN}}D + 18pn_{\mathrm{FFN}}D + 16D^2 + 12pD^2}{18n_{\mathrm{FFN}}D + 24D^2}. \tag{5}$$

For $n_{\mathrm{FFN}} = \frac{8}{3}D$, the formula becomes:

$$\frac{6\times\frac{8}{3}D^2 + 18p\times\frac{8}{3}D^2 + 16D^2 + 12pD^2}{18\times\frac{8}{3}D^2 + 24D^2}. \tag{6}$$

Simplifying,

$$\frac{16D^2 + 48pD^2 + 16D^2 + 12pD^2}{48D^2 + 24D^2} = \frac{32D^2 + 60pD^2}{72D^2}. \tag{7}$$

Method	CE	PPL
Dense	2.288	9.858
KEAP FFN Top-$\frac{2}{3}$	2.298	9.959

Table 2: **KEAP scaling performance** We report the cross-entropy loss and perplexity of a vanilla Llama-953M model versus KEAP Top-$\frac{2}{3}$ with token-level granularity applied to the FFN

Canceling D^2 gives

$$\frac{32 + 60p}{72} = \frac{4}{9} + \frac{5}{6}p. \tag{8}$$

Substituting $p = \frac{1}{3}$ and $p = \frac{2}{3}$ gives FLOP ratios of approximately 0.72 and 1, respectively.

C 1B-Parameter Experiment

To assess whether KEAP remains effective at a larger model scale, we train a Llama-3-style model with approximately one billion parameters. The model has 953M parameters and consists of 16 Transformer layers, a hidden dimension of 2,048, and 32 attention heads. It follows the same training setup described in Sec. 4.1 and is evaluated on the same held-out FineWeb set used in the experiments of Section 4.2. We reuse the learning rate tuned for the 488M-parameter dense baseline for both the dense and KEAP 953M runs, without additional tuning. For KEAP, we apply token-level Top-$\frac{2}{3}$ selection exclusively to the FFN layers. The results are reported in Table 2.

KEAP increases perplexity from 9.858 to 9.959, corresponding to a relative degradation of only 1.02%. This is smaller than the 1.66% relative degradation observed for the corresponding FFN Top-$\frac{2}{3}$ token-level setting on the 488M-parameter model.

On four H100 GPUs, with a sequence length of 2,048 and a local batch size of 16 per GPU, the 953M-parameter model uses 76.5 GiB per GPU without activation checkpointing and 31.0 GiB with Full AC. In this setting, KEAP requires only an additional 11 KiB per GPU for batch-level selection and 0.34 GiB per GPU for token-level selection when the forward-pass selection masks are stored using one bit per decision.

D Results with Block-granularity

The experiments reported in Table 1b use a single mask shared across all samples in a batch, allowing the retained FFN computation to be executed using standard dense GEMMs. For instance, on the 953M-parameter model described in Appendix C, trained on four H100 GPUs with a sequence length of 2,048 and a local batch size of 16 per GPU, batch-level FFN KEAP reduces training-step time by 26% with Top-$\frac{1}{3}$ and by 18% with Top-$\frac{2}{3}$ relative to Full AC. However, token-level selection generally achieves better accuracy, as shown in Tables 1a and 1b, because each token independently selects its most important FFN channels. However, this accuracy comes at the cost of less regular computation, since different tokens may retain different channels.

We explore two complementary ways of navigating this accuracy–efficiency trade-off. First, a mask can be shared across groups of tokens, gradually moving from fine-grained token-level selection toward more structured batch-level selection and producing larger, more regular matrix operations. Second, FFN channels can be partitioned into blocks, with entire blocks selected jointly. Tokens that retain the same blocks can then be packed and processed using grouped GEMMs. Larger channel blocks generally improve computational efficiency, whereas smaller or infrequently selected blocks remain more sensitive to dispatch, memory-movement, and kernel-launch overhead. We therefore study token-group and channel-group sizes jointly, spanning a continuum between fine-grained token-level KEAP and structured batch-level KEAP while keeping the retained fraction k/n fixed. To our knowledge, this joint grouping strategy has not previously been studied for approximate activation-checkpoint rematerialization. These block-granularity accuracy ablations were conducted on NVIDIA V100 GPUs using FP16 mixed precision.

Figure 4 shows the resulting accuracy trade-off. For Top-$\frac{2}{3}$, channel-group sizes of 16 and 32 under token-level selection increase relative perplexity by only 0.93% and 1.52%, respectively. As the

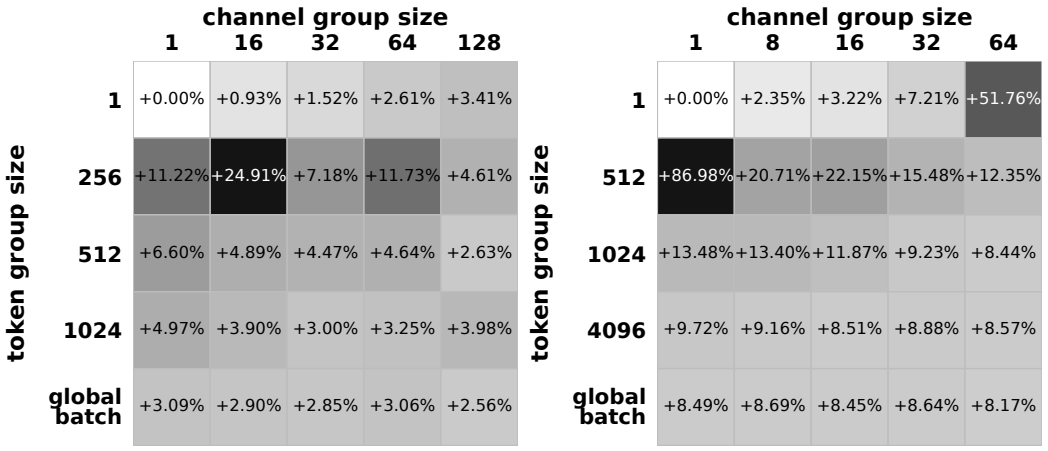


Figure 4: **Effect of token and channel grouping in KEAP FFN.** We train a Llama-134M model while varying the token-group size and FFN channel-group size. Each cell reports the relative perplexity degradation with respect to per-token, per-channel KEAP at the same sparsity fraction. Left: Top-$\frac{2}{3}$. Right: Top-$\frac{1}{3}$.

token-group size increases, performance approaches that of global-batch selection. For example, a token-group size of 512 combined with a channel-group size of 128 yields a 2.63% increase in relative perplexity, compared with 2.56% for global-batch selection at the same channel-group size. Top-$\frac{1}{3}$ is more sensitive to the grouping configuration, although larger token groups similarly approach the global-batch result. The corresponding speedups depend strongly on the target hardware and the implementation of data movement and grouped kernels, making their systematic characterization an important direction for future work.

E Evaluation Loss

We present the results from Tab. 1 as their corresponding cross entropy values in Tab. 3

F Assets and licenses

We use FineWeb [39], available at https://huggingface.co/datasets/HuggingFaceFW/fineweb, under the ODC-By 1.0 license. We use TorchTitan, available at https://github.com/pytorch/torchtitan, under the BSD-3-Clause license. We use the Llama-2 7B tokenizer, available through the Llama-2 model release, under the Meta Llama 2 Community License Agreement: https://ai.meta.com/llama/license/. All models are trained from scratch; we do not use or redistribute pretrained model weights.

G Compute resources

Each run reported in the main results of Sec. 4.2 required less than 30 H100-equivalent GPU-hours with an efficient implementation. The total compute required for the reported main-result experiments was below 1000 H100-equivalent GPU-hours.

H Limitations

Our main evaluation focuses on Llama-style models up to 488M parameters, supplemented by a 953M-parameter scaling experiment, and results may differ at substantially larger scales, on other architectures, or on other data distributions. Most results are based on single training runs due to computational cost, so we do not report run-to-run variance. For attention rematerialization with a token level selector we mention that key-value computations constrain the achievable savings.

Method	$\frac{k}{n}$	AC-FLOPs	CE
AC applied on FFN layers			
Llama-488M	1	1.33	2.487
Llama-419M	1	1	2.521
Rand	0.66	1	2.553
	0.33	0.66	2.688
Opt	0.66	1	2.532
	0.33	0.66	2.614
Top	0.66	1	**2.504**
	0.33	0.66	2.523
AC applied on Attention layers			
Llama-488M	1	1.33	2.487
Llama-468M	1	1	**2.499**
Rand	0.66	1	2.530
	0.33	0.83	2.598
Opt	0.66	1	2.507
	0.33	0.83	2.545
Top	0.66	1	**2.499**
	0.33	0.83	2.522
AC applied on Attention+FFN layers			
Llama-488M	1	1.33	2.487
Llama-400M	1	1	2.531
Rand	0.66	1	2.601
	0.33	0.72	2.885
Opt	0.66	1	2.550
	0.33	0.72	2.696
Top	0.66	1	**2.512**
	0.33	0.72	2.557

(a) Token-level granularity

Method	$\frac{k}{n}$	AC-FLOPs	CE
AC applied on FFN layers			
Llama-488M	1	1.33	2.487
Llama-419M	1	1	**2.521**
Rand	0.66	1	2.570
	0.33	0.66	2.744
Opt	0.66	1	2.557
	0.33	0.66	2.660
Top	0.66	1	2.534
	0.33	0.66	2.595
AC applied on Attention layers			
Llama-488M	1	1.33	2.487
Llama-456M	1	1	**2.503**
Rand	0.66	1	2.562
	0.33	0.66	2.672
Opt	0.66	1	2.549
	0.33	0.66	2.622
Top	0.66	1	2.514
	0.33	0.66	2.529
AC applied on Attention+FFN layers			
Llama-488M	1	1.33	2.487
Llama-387M	1	1	**2.533**
Rand	0.66	1	2.657
	0.33	0.66	3.007
Opt	0.66	1	2.624
	0.33	0.66	2.837
Top	0.66	1	2.553
	0.33	0.66	2.649

(b) Batch-level granularity

Table 3: **KEAP performance**. We report the cross entropy loss of a Llama-488M model under different AC strategies, using per-token (**a**) and per-batch (**b**) granularity during re-computation as described in Sec. 3.3. AC-FLOPs denotes the ratio between the FLOP cost of each AC method and the cost of standard backpropagation. (details can be found in App. B)

I Broader Impacts

This work is a training-efficiency method whose primary broader impact is to reduce the compute and energy overhead of large-model training. More efficient training may also lower the barrier to training larger models, including models used in harmful or poorly governed applications. We do not identify a direct path to negative societal applications beyond this generic dual-use effect of improving neural-network training efficiency.

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 main claims stated in the abstract and introduction are supported by both the theoretical analysis and experiments. The main empirical results are based on a tuned 488M-parameter Llama-style model trained with a compute-optimal token budget. The claims are limited to the model scales studied.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: The paper includes a limitations section discussing the scope of the evaluation, which is limited to Llama-style models up to 1B parameters trained on FineWeb. It also discusses the use of single training runs and the constrained savings for token level KEAP for attention due to key-value computations under token-level selection.

   Guidelines:

   - The answer [N/A] 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:[Yes]

   Justification: The paper states the conditions under which the theoretical results hold and complete proofs are provided in the appendix.

   Guidelines:

   - The answer [N/A] 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 paper provides sufficient details on model architecture, training setup, hyperparameters, and evaluation protocol to reproduce the main results.

   Guidelines:

   - The answer [N/A] 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: The data used in the experiments are publicly available, and the paper describes the data source and training setup. The implementation code is also shared.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental section specifies datasets, model configurations, training procedure, and optimization details needed to understand the results. Furthermore, these details follow the protocol of another peer reviewed paper.

   Guidelines:

   - The answer [N/A] 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: Results are based on single training runs due to computational cost; error bars or variance estimates are not reported.

   Guidelines:

- The answer [N/A] 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 paper reports the GPU types, number of GPUs, numerical precision, model sizes, token budgets, memory and FLOP analyses, and an upper bound on GPU-hours for the reported main-result experiments.

   Guidelines:

   - The answer [N/A] 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: The work conforms to the NeurIPS Code of Ethics. It does not involve human subjects, crowdsourcing, surveillance data, security vulnerabilities, or the release of high-risk models. The experiments use an existing public language-model training dataset and standard training software, and the paper reports relevant compute and reproducibility details.

   Guidelines:

   - The answer [N/A] 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: [Yes]

    Justification: A dedicated section in the appendix discusses the positive impact of reducing compute and energy overhead for large-model training, as well as the generic dual-use concern that more efficient training may lower the barrier to training larger models, including models used in harmful or poorly governed applications.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a pretrained model, image generator, scraped dataset, or other high-risk asset requiring release safeguards. The contribution is a training-efficiency method.

    Guidelines:

    - The answer [N/A] 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: In a dedicated section in the appendix the paper cites the existing datasets, software, and model/tokenizer assets used in the experiments and states that they are used under their respective licenses and terms of use.

Guidelines:

- The answer [N/A] 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: [Yes]

Justification: Source code is released.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The work does not involve crowdsourcing, user studies, or human-subject experiments.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The work does not involve human subjects, so IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] 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.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used as part of the core method, experiments, or scientific contribution. Any use is limited to writing or editing assistance and does not affect the methodology or results.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.