Greedy-Gnorm: A Gradient Matrix Norm-Based Alternative to Attention Entropy for Head Pruning

Yuxi Guo yuxiguo03@gmail.com
SWUFE-UD Institute of Data Science
Southwestern University of Finance and Economics
Chengdu, Sichuan, China

Paul Sheridan paul.sheridan.stats@gmail.com
School of Mathematical and Computational Sciences
University of Prince Edward Island
Charlottetown, PE, Canada

Editor: TBD

Abstract

Attention head pruning has emerged as an effective technique for transformer model compression, an increasingly important goal in the era of Green AI. However, existing pruning methods often rely on static importance scores, which fail to capture the evolving role of attention heads during iterative removal. We propose Greedy-Gradient norm (Greedy-Gnorm), a novel head pruning algorithm that dynamically recalculates head importance after each pruning step. Specifically, each head is scored by the elementwise product of the ℓ_2-norms of its Q/K/V gradient blocks, as estimated from a hold-out validation set and updated at every greedy iteration. This dynamic approach to scoring mitigates against stale rankings and better reflects gradient-informed importance as pruning progresses. Extensive experiments on BERT, ALBERT, RoBERTa, and XLM-RoBERTa demonstrate that Greedy-Gnorm consistently preserves accuracy under substantial head removal, outperforming attention entropy. By effectively reducing model size while maintaining task performance, Greedy-Gnorm offers a promising step toward more energy-efficient transformer model deployment.

Keywords: Attention head pruning, Gradient-based importance, Green AI, Transformer compression, Transformer models

1 Introduction

Transformer architectures have become foundational in natural language processing, serving as the backbone of contemporary large language models. While these models achieve remarkable accuracy across a wide range of tasks, their computational demands and parameter redundancy raise concerns about energy efficiency, deployment cost, and feasibility on resource-constrained devices (Strubell et al., 2019). The attention mechanism, in particular, is computationally expensive and potentially overparameterized, motivating research into techniques for reducing redundancy with minimal loss of performance.

A prominent strategy is attention head pruning, which reduces model size by removing those heads that are determined to be of least importance (Voita et al., 2019). This approach complements other such compression methods as model quantization (Wang et al., 2024) and knowledge distillation (Muralidharan et al., 2024; Sreenivas et al., 2024), and can be integrated with them to achieve additional efficiency gains. By directly targeting redundant heads, pruning can reduce both memory footprint and computation time, thereby enabling more sustainable and deployable transformer models (Wang et al., 2021; Lagunas et al., 2021; Shim et al., 2021).

Existing pruning methods have key limitations. A commonly used approach is attention entropy (AE), which assumes that heads with lower entropy produce more concentrated, or peaked, attention distributions (i.e., greater probability mass is placed on a few key positions) and are therefore more important (Voita et al., 2019; Wang and Tu, 2020). However, AE is problematic when attention becomes diffused over long input sequences, as the resulting small probability values make the entropy computation vulnerable to numerical underflow. Moreover, most existing methods rely on static importance scores that are computed once before pruning (Michel et al., 2019; Hao et al., 2020; McCarley et al., 2021). Since gradients and model dynamics evolve after each head removal, such static scores may become stale, potentially leading to suboptimal pruning decisions.

In this paper, we introduce Greedy-Gnorm, a dynamic, gradient-driven pruning strategy that recalculates transformer model head importance after every pruning step. Each head is scored by Gnorm, defined as the elementwise product of the ℓ_2-norms of the model's Q/K/V gradient matrices, estimated on a hold-out validation set. By recomputing scores at each greedy iteration, Greedy-Gnorm adapts to evolving gradients and avoids the pitfalls of static ranking. When computing AE values, we impose a small lower bound $\varepsilon > 0$ on each attention probability to prevent $\log(0)$ numerical underflow errors while leaving the pruning order essentially in tact. We refer to this technique as the ε-rectified entropy variant. To avoid numerical instability in AE-based methods, we compute entropy using an ε-rectified distribution, where a small constant ε is added to each attention term to prevent taking the logarithm of 0. This stabilization does not alter the head-importance ranking, so the resulting pruning order remains identical to the non-rectified case.

We validate Greedy-Gnorm on four widely used transformer model families: BERT (Devlin et al., 2018), ALBERT (Lan et al., 2019), RoBERTa (Liu et al., 2019), and XLM-RoBERTa (Conneau et al., 2019). Our experiments show that Greedy-Gnorm consistently preserves task accuracy under substantial head removal and exhibits stable behavior across architectures. Compared with AE and a random pruning baseline, our method yields more reliable pruning trajectories. After pruning, BERT, ALBERT, RoBERTa, and XLM-RoBERTa retain high accuracy, showing only modest declines relative to their unpruned counterparts. For BERT, we retain approximately 20% of the heads while maintaining 90.08% accuracy, compared to 96.82% accuracy before pruning. Compared with the AE scoring approach and random baseline, Greedy-Gnorm produces smoother, more reliable pruning trajectories and achieves higher accuracy at equivalent pruning rates.

The remainder of this paper is organized as follows. Section 2 introduces transformer model notation, surveys pruning and efficiency methods for these models, and situates our work within this context. Section 3 formalizes the Greedy-Gnorm algorithm, with emphasis on the method's prune–recompute procedure and the ε-rectified entropy variant used to ensure numerical stability during head ranking. Section 4 details datasets, models, baselines, and reports results and ablations. Finally, Section 5 discusses limitations and future directions, and Section 6 provides an overview of our contributions.

2 Background

This section establishes the notation for transformer models adopted in the remainder of the paper and summarizes essential background on attention head pruning.

2.1 Transformer Model Notation

Let $\mathcal{X}$ denote a set of input sentences. Each sentence $x \in \mathcal{X}$ has a length of t_x tokens after subword tokenization. Let the input be a tokenized sequence $x = (x_1, \ldots, x_{t_x})$ after subword tokenization for a sentence. For an input sentence x, each token x_i is an index in the vocabulary $\mathcal{V}$ with $|\mathcal{V}| = V$ (i.e., $x_i \in \{1, \ldots, V\}$).

Consider a transformer model $\mathcal{M}$ with $L > 0$ layers, each containing $H > 0$ heads, yielding a total of $N = L \times H$ heads. We assume the model has been trained on a set $\mathcal{X}$. Let $d_{\mathcal{M}} > 0$ denote the dimension of the vector (i.e., embedding size). We assume that input and output sizes are held constant at $d_{\mathcal{M}}$ across the L layers. An embedding layer is a matrix $E \in \mathbb{R}^{V \times d_{\mathcal{M}}}$ that maps each token to a vector $E[x_i] \in \mathbb{R}^{d_{\mathcal{M}}}$. Stacking these row vectors yields the embedding sequence in a matrix $X \in \mathbb{R}^{t_x \times d_{\mathcal{M}}}$ (i.e., a matrix with one $d_{\mathcal{M}}$-dimensional row per token). We assume the model preserves this width across all L layers, so the input and output of each layer are in $\mathbb{R}^{t_x \times d_{\mathcal{M}}}$.

In multi-head attention, with H attention heads per layer, each global projection (i.e., the weights matrices W_Q, W_K, W_V) in each layer is split into H head-specific projection blocks. We conceptually partition the projection output channels into H equal parts. Let d_h denote the per-head output width. That is, the number of columns in a single head's $Q/K/V$ projection block. Thus each head uses a matrix of dimension $d_{\mathcal{M}} \times d_h$, where d_h is a head-wise constant given by $d_h = d_{\mathcal{M}}/H$. This choice ensures that concatenating the H heads produces an output of width $Hd_h = d_{\mathcal{M}}$ (i.e., the multi-head attention block preserves the model width, so the output has the same dimensionality as the input). The full projection can thus be viewed as the concatenation of H head-specific blocks.

The projection matrices for head h ($1 \le h \le H$) in layer ℓ ($1 \le \ell \le L$) are given by

$$Q^{(\ell,h)} \;=\; X\,W_Q^{(\ell,h)}, \quad K^{(\ell,h)} \;=\; X\,W_K^{(\ell,h)}, \quad V^{(\ell,h)} \;=\; X\,W_V^{(\ell,h)}, \tag{1}$$

with $W_Q^{(\ell,h)}, W_K^{(\ell,h)}, W_V^{(\ell,h)} \in \mathbb{R}^{d_{\mathcal{M}} \times d_h}$, and hence $Q^{(\ell,h)}, K^{(\ell,h)}, V^{(\ell,h)} \in \mathbb{R}^{t_x \times d_h}$. The row-wise attention matrix is defined as

$$A^{(\ell,h)}(x) \;=\; \mathrm{softmax}\!\left(\frac{Q^{(\ell,h)} K^{(\ell,h)\top}}{\sqrt{d_h}}\right) \in [0,1]^{t_x \times t_x}, \tag{2}$$

where $\sum_{j=1}^{t_x} a_{ij}^{(\ell,h)}(x) = 1$ for all i. The head output is $Z^{(\ell,h)} = A^{(\ell,h)} V^{(\ell,h)} \in \mathbb{R}^{t_x \times d_h}$, and the multi-head output of layer ℓ is $\mathrm{Concat}_h\,(Z^{(\ell,h)})\,W_O^{(\ell)}$ with $W_O^{(\ell)} \in \mathbb{R}^{(Hd_h) \times d_{\mathcal{M}}}$, where $\mathrm{Concat}_h(\cdot)$ denotes the concatenation of all head outputs along the feature dimension. We will refer to the head-specific parameter blocks $W_Q^{(\ell,h)}, W_K^{(\ell,h)}, W_V^{(\ell,h)}$ when defining gradient-based scores. The gradients of these matrices have the same dimensions as the corresponding blocks.

2.2 Related Work

Here we review head-importance criteria and pruning strategies relevant to our approach. We begin with preliminaries and AE scoring and its numerical caveats, then cover gradient-based measures and the need for step-wise recomputation, and finally summarize depth-wise, width-wise, and length-wise pruning families and tooling.

2.2.1 Attention Entropy

For a sentence $x \in \mathcal{X}$ with length t_x, consider the attention-score matrix $A^{(\ell,h)}(x) \in [0,1]^{t_x \times t_x}$ of head (ℓ, h) as defined in Eq. (2). Writing its entries as $a_{ij}^{(\ell,h)}(x)$, the expected AE is defined as

$$\mathbb{E}\big[AE(\ell,h)\big] = -\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\frac{1}{t_x}\sum_{i=1}^{t_x}\sum_{j=1}^{t_x} a_{ij}^{(\ell,h)}(x), \log a_{ij}^{(\ell,h)}(x). \tag{3}$$

AE serves as a proxy for the "focus" of an attention head, quantifying how concentrated or diffuse its attention distribution is. However, AE disregards gradients and can be numerically brittle for long sequences, where many small $a_{ij}^{(\ell,h)}$ values make $-a \log a$ unstable under finite precision, occasionally leading to underflow or NaNs.

2.2.2 Gradient-based Importance

Gradient-aware scores use the loss derivatives $\partial\mathcal{L}(x)/\partial A^{(\ell,h)}(x)$ with respect to a head's attention. Let $A^{(\ell,h)}(x) \in [0,1]^{t_x \times t_x}$ be the attention of head (ℓ, h), and collect all heads in layer ℓ as

$$A^{(\ell)}(x) = \big[A^{(\ell,1)}(x), \dots, A^{(\ell,H)}(x)\big]. \tag{4}$$

The attribution score for head (ℓ, h) is computed using a gradient-based self-attention analysis method (Hao et al., 2020). Self-attention attribution for head (ℓ, h) is

$$\begin{aligned}\mathrm{Attr}^{(\ell,h)}(x) &= A^{(\ell,h)}(x) \;\odot\; \int_0^1 \nabla_{A^{(\ell,h)}} F\big(x;\, \alpha\, A^{(\ell)}(x)\big)\, d\alpha \qquad (5)\\ &\approx \frac{1}{m}\, A^{(\ell,h)}(x) \;\odot\; \sum_{k=1}^{m} \nabla_{A^{(\ell,h)}} F\big(x;\, \tfrac{k}{m}\, A^{(\ell)}(x)\big) \in \mathbb{R}^{t_x \times t_x}, \qquad (6)\end{aligned}$$

where $F(\cdot;\cdot)$ is a differentiable scalarization of the model output and $\odot$ denotes the Hadamard product. Then define head importance by

$$I^{(\ell,h)} = \mathbb{E}_{x\sim X}\big[\max(\mathrm{Attr}^{(\ell,h)}(x))\big]. \tag{7}$$

where $F(\cdot)$ is a differentiable scalarization of the model output and $\odot$ is the Hadamard product. Taylor-based importance (TIS) (Zhong and Zhou, 2024) can be written using the inner product

$$\mathrm{TIS}^{(\ell,h)} \;=\; \mathbb{E}_x\left|A^{(\ell,h)}(x)\;\frac{\partial\mathcal{L}(x)}{\partial A^{(\ell,h)}(x)}\right|. \tag{8}$$

Beyond these, differentiable subset selection (Li et al., 2023) and Shapley-based head valuation (Held and Yang, 2022) have been explored. However, once any head is pruned, gradients and representations shift, so importance must be re-evaluated at each step; otherwise scores become stale and may mislead subsequent choices (Michel et al., 2019; McCarley et al., 2021; Parnami et al., 2021; Shim et al., 2021). This motivates our greedy, recompute-as-you-pruning procedure.

2.2.3 Other Transformer Pruning Strategies

Transformer compression has been explored along three structural dimensions: depth, width, and sequence length, each targeting different computational bottlenecks.

One line of work focuses on reducing model depth. For example, static layer removal and sensitivity-driven pruning reduces depth with post-hoc fine-tuning (McCarley et al., 2021). Dynamic policies skip layers per input to trade quality for cost (Bapna et al., 2020). Recent Mixture-of-Depths dynamically allocates compute across layers (Raposo et al., 2024), and layer importance can be optimized via NAS/analysis (Klein et al., 2024; Zhang et al., 2024). Related architectural simplifications include average-attention networks (Zhang et al., 2018), reordering sublayers (Press et al., 2020), and light-weight stacks (Mehta et al., 2021).

Another research direction explores pruning along the width dimension, primarily within layers. Head pruning ranges from early evidence of redundancy (Michel et al., 2019; Voita et al., 2019) to structured gating (Shim et al., 2021), search-based A* (Parnami et al., 2021), differentiable subsets (Li et al., 2023), Shapley valuations (Held and Yang, 2022), and sparse attention with cascaded token-then-head sparsity (Wang et al., 2021). Feedforward neural network/channel/filter pruning targets the dominant MLP cost, often with block structure for hardware efficiency (Lagunas et al., 2021; Liu et al., 2021; Yu et al., 2022). Tooling such as TextPruner supports practical pipelines (Yang et al., 2022).

A complementary body of work targets sequence length reduction. Token-level methods prune, merge, skip, or drop tokens to shorten effective context; policies can be static or dynamic/adaptive (Lee et al., 2022, 2025). System-oriented work reduces KV-cache traffic via selective fetching or cache designs (He and Wu, 2024; Sun et al., 2024), complementary to head pruning.

3 Methodology

In the section, we present the Greedy-Gnorm algorithm and introduce a simple technique designed to prevent underflow errors in AE calculations.

3.1 The Greedy-Gnorm Algorithm

Our goal is to prune a transformer model of redundant attention heads while preserving task accuracy. To this end, we propose Greedy-Gnorm, a greedy pruning scheme that leverages importance scores reflecting the current state of a partially pruned model. Each iteration consists of two steps: (1) compute the current Greedy-Gnorm score matrix based on a current gradient matrix using backpropagation, and (2) prune the attention head with the lowest score. This prune-recompute cycle continues until either a pruning budget is exhausted or an accuracy-based stopping criterion is satisfied, thereby avoiding the staleness of one-shot scores and producing more stable pruning trajectories. In what follows, we formalize notation, describe the construction of the scoring matrix, and analyze the algorithmic complexity of the proposed method.

3.1.1 Greedy-Gnorm Algorithm High-level Description

The Greedy-Gnorm routine is described in Algorithm 1. We adopt the notation of Section 2.1. Consider a transformer $\mathcal{M}$, given input X, the output of original $\mathcal{M}$ is $F_0(X)$. Let $M \in \{0,1\}^{L\times H}$ be the mask (1=kept, 0=pruned).

Algorithm 1 The Greedy-Gnorm algorithm.

```
Input: Initial model $F_0$; the set $\mathcal{X}$; number of layers $L$; heads per layer $H$; $M \in \{0,1\}^{L\times H}$
    and initialize $M \leftarrow \mathbf{1}_{L\times H}$; number of total heads $N \leftarrow L \times H$.
Output: Pruned model $F$ and accuracies of each pruned model.
 1: $F \leftarrow F_0$                                              ▷ current model
 2: $M \leftarrow \mathbf{1}_{L\times H}$                           ▷ mask (1 = keep, 0 = pruned)
 3: for $n = 0$ to $N - 1$ do                                       ▷ greedy re-computation loop
 4:     $(G_{Q(n)}, G_{K(n)}, G_{V(n)}) \leftarrow \textsc{ComputeGnorm}(F, \mathcal{X}, M)$
 5:     $S(n) \leftarrow G_{Q(n)} \odot G_{K(n)} \odot G_{V(n)}$    ▷ elementwise product, $S \in \mathbb{R}^{L\times H}$
 6:     $(\ell^\star, h^\star) \leftarrow \arg\min\{\, S(n)_{\ell h} \mid M_{\ell h} = 1 \,\}$   ▷ least-important head
 7:     $F \leftarrow \textsc{PruneHead}(F, \ell^\star, h^\star)$    ▷ apply head-gating or equivalent
 8:     $M_{\ell^\star h^\star} \leftarrow 0$                        ▷ update mask
 9:     $\textsc{GetAccuracy}(F, \mathcal{X})$                       ▷ get the accuracy of model on $\mathcal{X}$
10: end for
11: return $(F, M)$
```

Following Algorithm 1, we describe the full pruning procedure step by step and explain how each quantity is computed. At initialization (lines 1–2), the algorithm loads the pretrained transformer F_0 and sets up the binary mask $M \in \{0,1\}^{L\times H}$, where $M_{\ell h} = 1$ indicates that the head (ℓ, h) is active. The model F is iteratively updated under this mask throughout the pruning process.

The outer loop (line 3) performs $N = L \times H$ pruning iterations, each removing one attention head. At the beginning of each iteration, the subroutine ComputeGnorm (line 4) computes three gradient-norm matrices $G_{Q(n)}$, $G_{K(n)}$, and $G_{V(n)}$, corresponding respectively to the query, key, and value projection blocks $W_Q^{(\ell,h)}$, $W_K^{(\ell,h)}$, and $W_V^{(\ell,h)}$.

Line 5 combines these matrices through an elementwise product, which captures the joint strength of the three attention projections. $S(n)$ therefore acts as a unified importance score matrix: heads with smaller values of $S(n)_{\ell h}$ are considered less influential.

Next, line 6 identifies the least-important active head $(\ell^\star, h^\star)$ by finding the minimum entry in $S(n)$ over unpruned positions ($M_{\ell h} = 1$). The selected head is removed from the model in line 7 via the PruneHead operation, which structurally removes the corresponding attention head using the TextPruner framework (see Appendix B for implementation details and discussion of gradient collapse). The mask M is updated accordingly in line 8 to record this removal.

After pruning, the model's performance is immediately re-evaluated on $\mathcal{X}$ (line 9) using GetAccuracy. This greedy prune–recompute cycle continues until all heads have been ranked and sequentially pruned. The final output (F, M) (line 10) contains the fully pruned model and the final mask, where $M = \mathbf{0}_{L\times H}$ after all heads have been removed, indicating that the pruning process has ranked and eliminated every attention head.

3.1.2 Technical Details

To execute Algorithm 1, we need the gradient-norm matrices $G_{Q(n)}$, $G_{K(n)}$, and $G_{V(n)}$, each summarizing the average magnitude of parameter gradients for query, key, and value projections across all layers and heads. The elementwise product $S(n) = G_{Q(n)} \odot G_{K(n)} \odot G_{V(n)}$ thus aggregates their joint contribution and serves as a unified importance score matrix. In the following section, we formally define $G_{Q(n)}$, $G_{K(n)}$, $G_{V(n)}$, and $S(n)$.

We next define the Gnorm score, which quantifies the importance of each attention head based on its gradient matrix norms. For layer ℓ and head h, $W_Q^{(\ell,h)}, W_K^{(\ell,h)}, W_V^{(\ell,h)}$ are the head-specific projection blocks with dimensions as defined in Section 2.1. Write $q^{(\ell,h)}$ for the collection of scalars in $W_Q^{(\ell,h)}$, with $q_{a,b}^{(\ell,h)}$ its entry at a-th row and b-th column in the $W_Q^{(\ell,h)}$; analogously $k^{(\ell,h)}$ and $v^{(\ell,h)}$ for $W_K^{(\ell,h)}$ and $W_V^{(\ell,h)}$.

After pruning n heads, the current model is $F_n(\cdot)$ and we define our gradient matrix for Q of ℓ'th layer, h'th head and denote the Euclidean norm matrix for each head as follows

$$\begin{aligned}
\mathbf{G}_{q(n)}^{(\ell,h)}(X) &= \nabla_{W_Q^{(\ell,h)}} \|F_n(X)\| = \left[\frac{\partial \|F_n(X)\|}{\partial q_{a,b}^{(\ell,h)}} \right]_{1\le a\le d_{\mathcal{M}},\, 1\le b\le d_h} && (9)\\
\|\mathbf{G}_{q(n)}^{(\ell,h)}(X)\| &= \sqrt{\sum_{a=1}^{d_{\mathcal{M}}}\sum_{b=1}^{d_h}\left(\frac{\partial \|F_n(X)\|}{\partial q_{a,b}^{(\ell,h)}}\right)^2} && (10)\\
\mathbf{G}_{q(n)} &= \left(\|\mathbf{G}_{q(n)}^{(\ell,h)}(X)\|\right)_{1\le \ell\le L,\, 1\le h\le H} \in \mathbb{R}^{L\times H} && (11)\\
&= \left(\sqrt{\sum_{a=1}^{d_{\mathcal{M}}}\sum_{b=1}^{d_h}\left(\frac{\partial \|F_n(X)\|}{\partial q_{a,b}^{(\ell,h)}}\right)^2}\right)_{1\le \ell\le L,\, 1\le h\le H} \in \mathbb{R}^{L\times H}. && (12)
\end{aligned}$$

where $\mathbf{G}_{q(n)}^{(\ell,h)}$ means the gradient matrix of Q weights at ℓ^{th} layer h^{th} head. Correspondingly, $\mathbf{G}_{k(n)}^{(\ell,h)}$ and $\mathbf{G}_{v(n)}^{(\ell,h)}$ mean the gradient matrix of K and V weights at in ℓ^{th} layer h^{th} head. $\mathbf{G}_q(n)$,$\mathbf{G}_k(n)$ and $\mathbf{G}_v(n)$ are Euclidean norm matrices of Q, K and V in each head. We use $\|F_n(X)\|$ as a scalarization of the model output (e.g., logits ℓ_2 norm). Note that any differentiable scalar objective (e.g., task loss) can be used.

Based on different inputs in datasets, we need to consider the expectations of the gradient matrix norms:

$$\begin{aligned}
\mathbf{G}_{Q(n)} &= \mathbb{E}_X\big[\mathbf{G}_{q(n)}\big] & (13)\\
&= \Big(\mathbb{E}_X\big[\big\|\mathbf{G}^{(\ell,h)}_{q(n)}(X)\big\|\big]\Big)_{1\le\ell\le L,\ 1\le h\le H} \in \mathbb{R}^{L\times H} & (14)\\
&= \left(\frac{1}{|\mathcal{X}|}\sum_X \sqrt{\sum_{a=1}^{d_{\mathcal{M}}}\sum_{b=1}^{d_h}\left(\frac{\partial\big\|F_n(X)\big\|}{\partial q^{(\ell,h)}_{a,b}}\right)^2}\,\right)_{1\le\ell\le L,\ 1\le h\le H} \in \mathbb{R}^{L\times H}. & (15)
\end{aligned}$$

$\mathbf{G}_{Q(n)}$ is the matrix of expected norms for $L \times H$ heads, and analogously for $\mathbf{G}_{K(n)}$ and $\mathbf{G}_{V(n)}$. We take the expectation over a set $\mathcal{X}$ of sentences, where each X denotes the embedding result of a single sentence $x \in \mathcal{X}$ input to the transformer model.

Here, $G_{Q(n)}(\ell,h)$ denotes the entry in the ℓ-th row and h-th column of $\mathbf{G}_{Q(n)}$, with $G_{K(n)}(\ell,h)$ and $G_{V(n)}(\ell,h)$ defined analogously:

$$G_{Q(n)}(\ell,h) = \mathbb{E}_X\left[\big\|\mathbf{G}^{(\ell,h)}_{q(n)}(X)\big\|\right], \tag{16}$$

After pruning n heads, we define the Gnorm score matrix $\mathbf{S}(n) \in \mathbb{R}^{L\times H}$ as the elementwise product:

$$\mathbf{S}(n) \;=\; \mathbf{G}_{Q(n)} \odot \mathbf{G}_{K(n)} \odot \mathbf{G}_{V(n)},\ \mathbf{S}(n) \in \mathbb{R}^{L\times H}. \tag{17}$$

At each greedy step, we prune the head with the smallest $S(n)_{ij}$ and then recompute gradients and $\mathbf{S}(n{+}1)$. So $\mathbf{S}(n)$ tells that after pruning n heads, which head is the least important to do next pruning.

At each greedy step, we prune the head with the smallest $S(n)_{ij}$ and then recompute gradients and $\mathbf{S}(n{+}1)$. Thus, $\mathbf{S}(n)$ indicates, after pruning n heads, which of the remaining heads is least important for the next step.

The algorithm proceeds iteratively, ranking all heads by their Gnorm scores until every head has been evaluated or pruned. In practice, pruning can be stopped earlier when accuracy begins to drop sharply, which marks the empirical “inflection point” beyond which further pruning causes disproportionate degradation. This point provides a principled trade-off between compactness and performance.

3.2 Time Complexity

Let the calibration set $\mathcal{X}$ be divided into B mini-batches for gradient computation. In each greedy iteration, ComputeGnorm performs a full backward pass for every batch in each layer to obtain the $Q/K/V$ gradients of all heads, yielding a per-iteration cost of ComputeGnorm $= \Theta(L\,B)$. Because pruning proceeds iteratively, recomputing gradients after each head removal, the total number of iterations is proportional to N. Hence, the overall computational complexity of Greedy-Gnorm is $\mathcal{O}(N\,L\,B)$. The scoring and selection steps, which form the score tensor S and locating $\arg\min S$, require only $\mathcal{O}(N)$ time per iteration and are negligible compared to the cost of backpropagation.

3.3 A Technique for Underflow Error Avoidance

AE scores heads via $-\sum_i a_i \log a_i$ averaged over tokens and examples. On long sequences, attention may become diffuse resulting in many entries a_i approaching machine zero. This leads to $\log a_i$ underflow errors (or $a_i = 0$ after softmax rounding), producing log(0) and NaNs that corrupt averages and break pruning order. Although Greedy-Gnorm itself is gradient-based, we employ AE as a baseline. Hence we need a numerically stable way of computing AE scores.

Let $\mathbf{a} = (a_1, \ldots, a_n)$ denote a row of a head's attention matrix (a probability vector with $\sum_i a_i = 1$, $a_i \geq 0$). Information entropy is

$$A(\mathbf{a}) = -\sum_{i=1}^{n} a_i \log a_i. \tag{18}$$

To avoid underflow we consider two ε-rectified variants with a small $\varepsilon > 0$ (chosen so that $a_i + \varepsilon < 1$):

$$B(\mathbf{a}) = -\sum_{i=1}^{n} a_i \log\big(a_i + \varepsilon\big), \tag{19}$$

$$C(\mathbf{a}) = -\sum_{i=1}^{n} \big(a_i + \varepsilon\big) \log\big(a_i + \varepsilon\big). \tag{20}$$

The function B clips the *log* argument away from 0 while keeping the original weights a_i, removing log(0) but leaving the averaging weights unchanged. The function C also shifts the weights (from a_i to $a_i + \varepsilon$), making the entropy itself less sensitive to tiny entries and thus more numerically stable. For sufficiently small ε, both act as smooth perturbations cof A. In practice we adopt C for stronger stability (no NaNs) while preserving head rankings for typical ε (see Appendix C for discussion).

We choose C for rectification for two reasons: (i) it fully removes log(0) and (ii) it remains order-preserving for small ε. Let $F(\mathbf{a}) = \sum_i a_i \log a_i$. For two distributions $\mathbf{x}, \mathbf{y}$ with $F(\mathbf{y}) - F(\mathbf{x}) = \delta > 0$, consider $\mathbf{x}' = \mathbf{x} + \varepsilon\mathbf{1}$, $\mathbf{y}' = \mathbf{y} + \varepsilon\mathbf{1}$. Along the line segment joining $\mathbf{x}$ and $\mathbf{y}$, the directional derivative involves $\nabla F(\mathbf{a}) = (\log a_i + 1)_i$. Continuity implies that for sufficiently small ε, the sign of the line integral remains unchanged and thus the ordering remains unchanged:

$$\delta' = F(\mathbf{y}') - F(\mathbf{x}') = \int_{\mathbf{x}'}^{\mathbf{y}'} \nabla F(\mathbf{a}) \cdot d\mathbf{l} \ > \ 0. \tag{21}$$

However, for $B(\mathbf{a})$, it is uncertain to find whether the pruning order changes. At least, the pruning order induced by AE is preserved under $C(\mathbf{a})$ while eliminating underflow/NaN in practice.

4 Experiments

In this section, we present empirical evidence demonstrating that our proposed attention head pruning method outperforms both the AE approach and a random pruning baseline.

4.1 Setup

We design our experiments to evaluate whether Greedy-Gnorm effectively identifies and preserves the most important attention heads under varying pruning budgets, while maintaining competitive accuracy on selected classification tasks. Conceptually, our goal is to assess the degree to which dynamically updated gradient norms yield more stable pruning trajectories as compared with static and random baselines. In each experiment, we progressively removes heads according to a ranking criterion, and measure the resulting performance on downstream classification tasks.

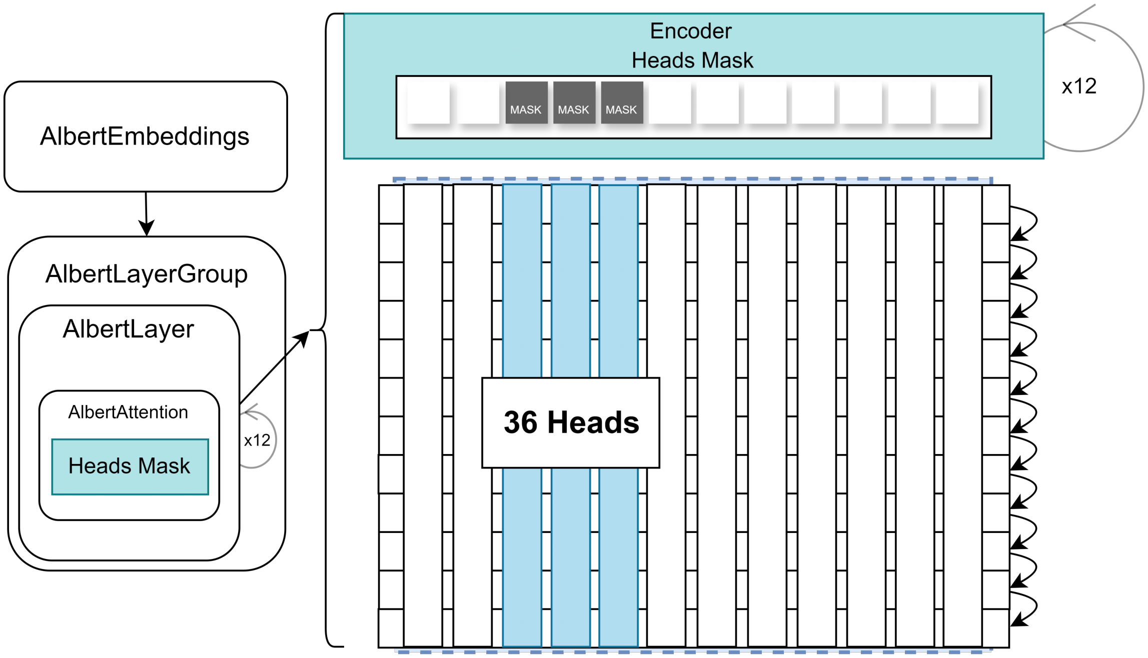

Figure 1: ALBERT tied-head structure (appearing as a vertical band in the mask visualization). Head h shares its parameters across all L layers, gating a single mask entry disables the same head index in every layer. One mask position affects 12 heads across layers.

Models and tasks. We evaluate the Greedy-Gnorm algorithm on four pretrained transformer models, each paired with a distinct downstream task: (1) BERT on financial sentiment classification, (2) ALBERT on Multi-Genre Natural Language Inference, (3) RoBERTa on tweet sentiment analysis, and (4) XLM-RoBERTa on language identification. In the case of ALBERT, parameter sharing ties all 12 attention layers, so one mask position simultaneously affects 12 heads, as illustrated in Figure 1.

Pruning methods. We compare five pruning strategies to evaluate the role of dynamic gradient-based ranking:

Greedy-Gnorm: A method that dynamically updates gradient norms to estimate head importance during iterative pruning. Scores are recomputed after each pruning step from the current model state.

AE: A static baseline that ranks heads by their AE. Scores are computed once on the unpruned model and then held fixed.

Inverse-AE: A static baseline that prunes the most informative (i.e., high-entropy) heads first, providing a complementary perspective to AE.

Inverse-Gnorm: A dynamic baseline that removes the most gradient-active heads first, thereby retaining the least important ones to illustrate the effectiveness of Greedy-Gnorm appraoch. If pruning in reverse Gnorm order (removing the highest-scoring heads first) causes a rapid accuracy collapse, this will validate that our score faithfully captures head importance.

Random pruning: A baseline that discards heads uniformly at each pruning step, providing a capacity-matched control to isolate the effect of head selection from mere capacity reduction.

Together, these strategies allow us to test whether dynamically updated, gradient-informed scores yield more stable and effective pruning trajectories than static, heuristic, or random approaches.

Evaluation metrics. Model performance is evaluated using classification accuracy on the held-out test sets. For each pruning method, we plot the accuracy curve against pruning rate to visualize robustness. Additionally, we report model sizes (in megabytes) before and after pruning to quantify compression efficiency. All results are averaged over three independent runs to mitigate randomness.

4.2 Results

We now present the empirical results and interpret the key findings from all experiments. Our analysis proceeds from verifying the gradient dynamics induced by pruning, to comparing Greedy-Gnorm against static and random baselines, and finally to examining the resulting model compression outcomes.

4.2.1 Gradients Change After Pruning

We begin by examining whether removing one attention head influences the gradients of others, a necessary condition for our dynamic scoring approach to be meaningful. As illustrated in Figure 2, pruning a single head causes nontrivial changes in the Q-gradient matrices across both the pruned and unpruned layers. Even after 64×64 pooling (from 768×768 matrices), the heatmaps exhibit consistent deviations from 0, indicating that pruning one head alters the gradient flow throughout the network. This observation empirically supports our assumption that head importance should be recomputed iteratively, since local pruning decisions induce global gradient redistribution.

4.2.2 Main Findings

Table 1 reports accuracy and model size before and after pruning across all architectures. It is clear that Greedy-Gnorm consistently reduces model size while preserving competitive accuracy. Greedy-Gnorm achieves substantial compression (e.g., $\approx$ 22.5% size reduction

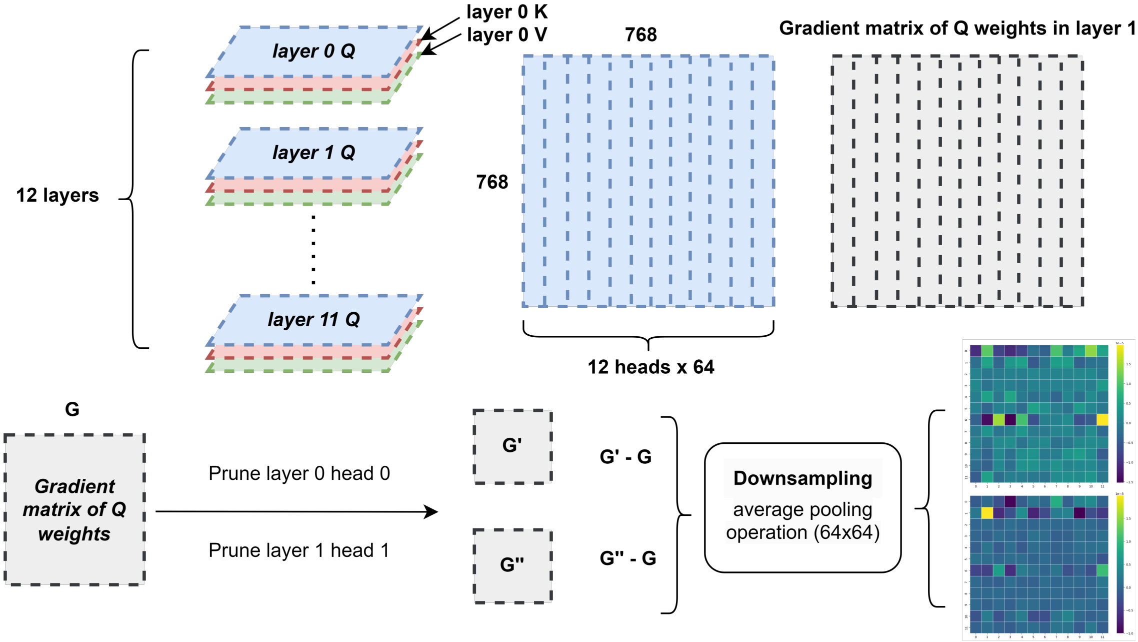

Figure 2: Gradient changes after pruning (downsampled by 64 × 64 pooling from 768 × 768 to 12 × 12). Colors differ from 0 across settings, indicating nonlocal gradient shifts.

on BERT) with competitive post-pruning accuracy. Similar trends hold for RoBERTa and XLM-RoBERTa, while ALBERT's already compact design yields smaller absolute savings.

Table 1: Greedy-Gnorm solutions. Percent reductions in both model size and accuracy are shown for readability.

	Accuracy (%)			Size (MB)		
Model	Before Pruning	After Pruning	Pct. Decrease	Before Pruning	After Pruning	Pct. Reduction
BERT	96.82	90.08	6.97%	390.13	302.29	22.52%
ALBERT	84.48	77.76	7.95%	44.58	42.33	5.05%
RoBERTa	87.80	86.40	1.59%	1355.60	1110.42	18.09%
XLM-RoBERTa	99.73	90.97	8.78%	1060.71	981.88	7.43%

Greedy-Gnorm outperforms static baselines. We next compare Greedy-Gnorm with the dynamic (i.e., Inverse-Gnorm) and static (i.e., AE and Inverse-AE) baselines across all four models. As shown in Figure 3, Greedy-Gnorm exhibits a markedly smoother accuracy decay curves as pruning progresses. For RoBERTa, approximately 70% of the heads suffice to retain near-original performance. A similar "useful head" fraction is observed for other architectures, suggesting that a relatively small subset of heads drives most of the task-relevant computation. In contrast, AE underperforms Greedy-Gnorm across all models. This is most notable on BERT, where Greedy-Gnorm preserves around 90% accuracy with fewer than 20% of heads retained. These results demonstrate that dynamic gradient-based ranking provides more reliable pruning trajectories than static entropy-based criteria.

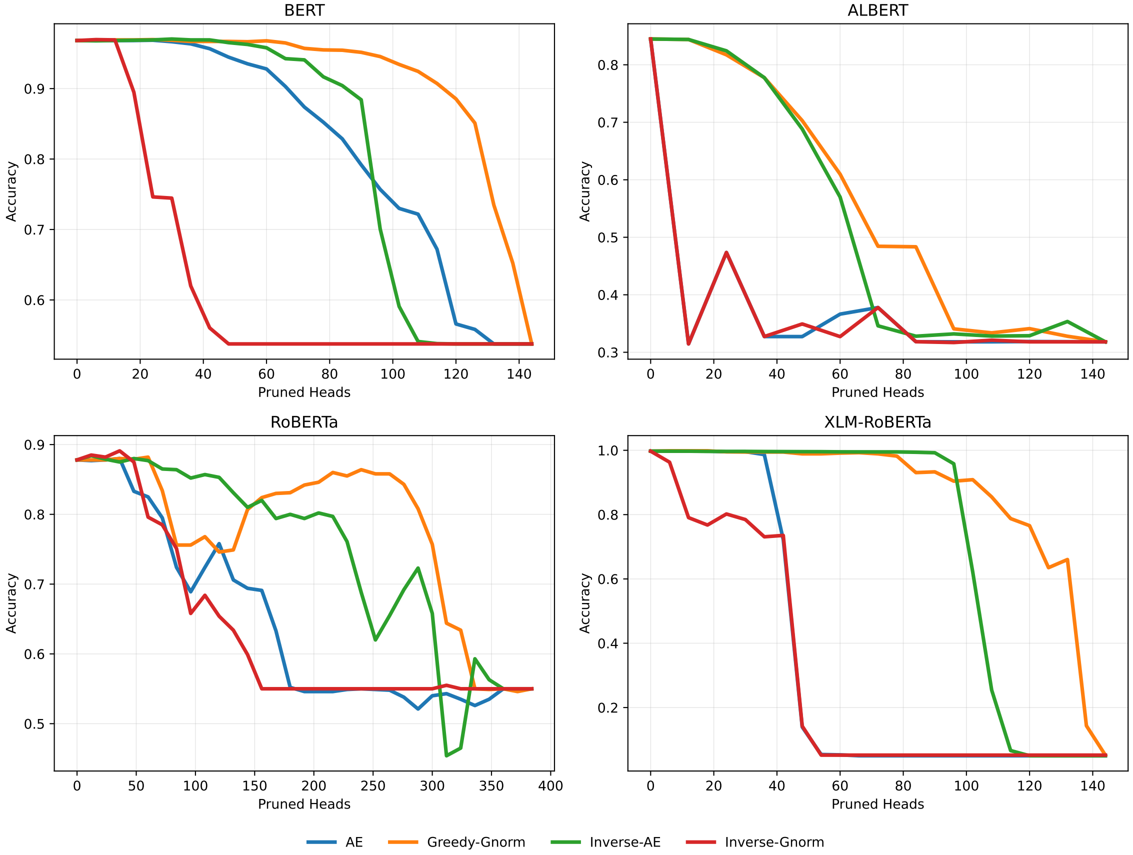

Figure 3: Greedy-Gnorm vs. AE (and inverse variants). Greedy-Gnorm is more stable and preserves accuracy under deeper pruning.

Validating importance scores via inverse pruning. To further validate the meaningfulness of the importance scores, we perform inverse pruning, in which the ranking order is deliberately reversed. This means that the most important heads, as estimated by each scoring method, are pruned first. This "reverse stress test" provides a diagnostic view of how well a given criterion distinguishes essential from redundant components. If the scoring function is informative, inverse pruning should lead to a rapid and monotonic performance collapse. Empirically, this is precisely what we observe. Inverse-Gnorm curves fall sharply after only modest pruning ratios, whereas Inverse-AE shows slower and more erratic degradation, reflecting noisier head importance estimates. The strong asymmetry between normal and Inverse-Gnorm trajectories confirms that our gradient-based metric captures functional salience rather than random variation.

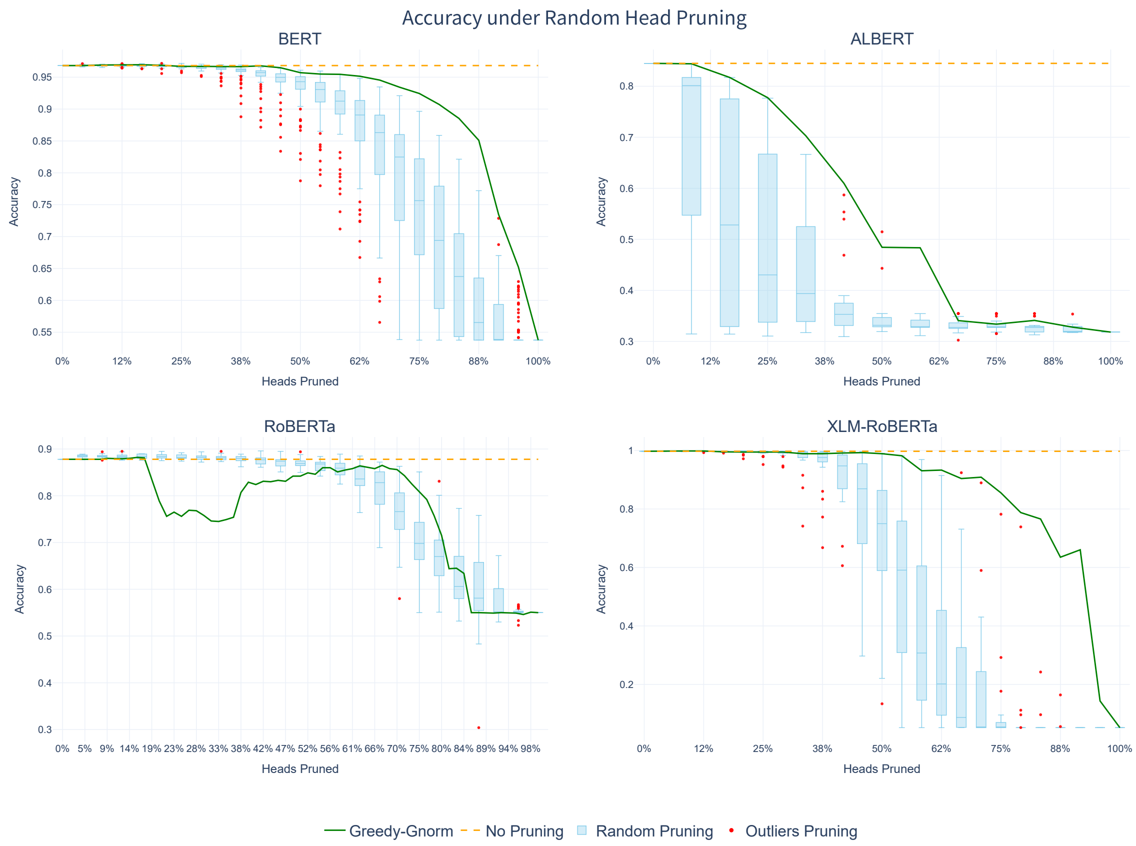

Figure 4: Greedy-Gnorm versus random pruning across pruning rates. Boxplots summarize multiple random masks per rate, while the green curves show Greedy-Gnorm task accuracy. The dashed yellow line shows accuracy with no pruning.

Comparison with random pruning. We further benchmark Greedy-Gnorm against random head removal to quantify the contribution of structured selection. As illustrated in Figure 4, across models, Greedy-Gnorm typically achieves the slowest overall accuracy decay. This slower decay indicates that the retained heads are genuinely informative and important. For RoBERTa, however, the abundance of redundant heads means that modest pruning can occasionally improve accuracy, yielding non-monotonic “rebound” segments. Nevertheless, the overall pruning outcomes on RoBERTa remain satisfactory. The discrepancy widens at higher pruning rates, reflecting that random removal often discards heads that remain functionally important. In contrast, Greedy-Gnorm’s score-driven masking avoids such destructive deletions, producing more stable and reproducible pruning behavior even when over 75% of heads are pruned. This robustness underscores the value of gradient-informed importance estimation in guiding efficient compression.

4.3 An Investigation of Pruning Solutions

We next interpret the learned pruning patterns by visualizing the final head masks (light = pruned, dark = kept) across backbones and then drill down into BERT’s per-module parameter changes.

Figure 5 visualizes the final head-retention masks for all backbones, with light cells indicating pruned heads and dark cells indicating retained heads. Each panel is arranged by layer (rows) and head index (columns). BERT exhibits dispersed retention across depth, ALBERT shows vertical bands due to parameter sharing (shared parameters induce identical keep/prune decisions across reused layers), while RoBERTa and XLM-RoBERTa display layer-dependent selectivity.

Table 2 reports BERT’s per-module parameters before/after pruning. Most reduction concentrates in the Encoder block, as expected for head pruning (Embeddings, Pooler, and task head remain unchanged). Despite a sizable encoder shrinkage (i.e., from 85.05M to 62.03M parameters), post-pruning accuracy remains competitive (i.e., from 96.82% to 90.08%).

Table 2: BERT parameters before/after pruning with Greedy-Gnorm (MB=megabytes).

	Before Pruning			After Pruning		
Layer	Params	Share (%)	MB	Params	Share (%)	MB
Model	102,269,955	100.00	390.13	79,244,355	100.00	302.29
BERT	102,267,648	100.00	390.12	79,242,048	100.00	302.28
Embeddings	16,622,592	16.25	63.41	16,622,592	20.98	63.41
Encoder	85,054,464	83.17	324.46	62,028,864	78.28	236.62
Pooler	590,592	0.58	2.25	590,592	0.75	2.25
Classifier	2,307	0.00	0.01	2,307	0.00	0.01
Weight	2,304	0.00	0.01	2,304	0.00	0.01
Bias	3	0.00	0.00	3	0.00	0.00
Accuracy	96.82%			90.08%		

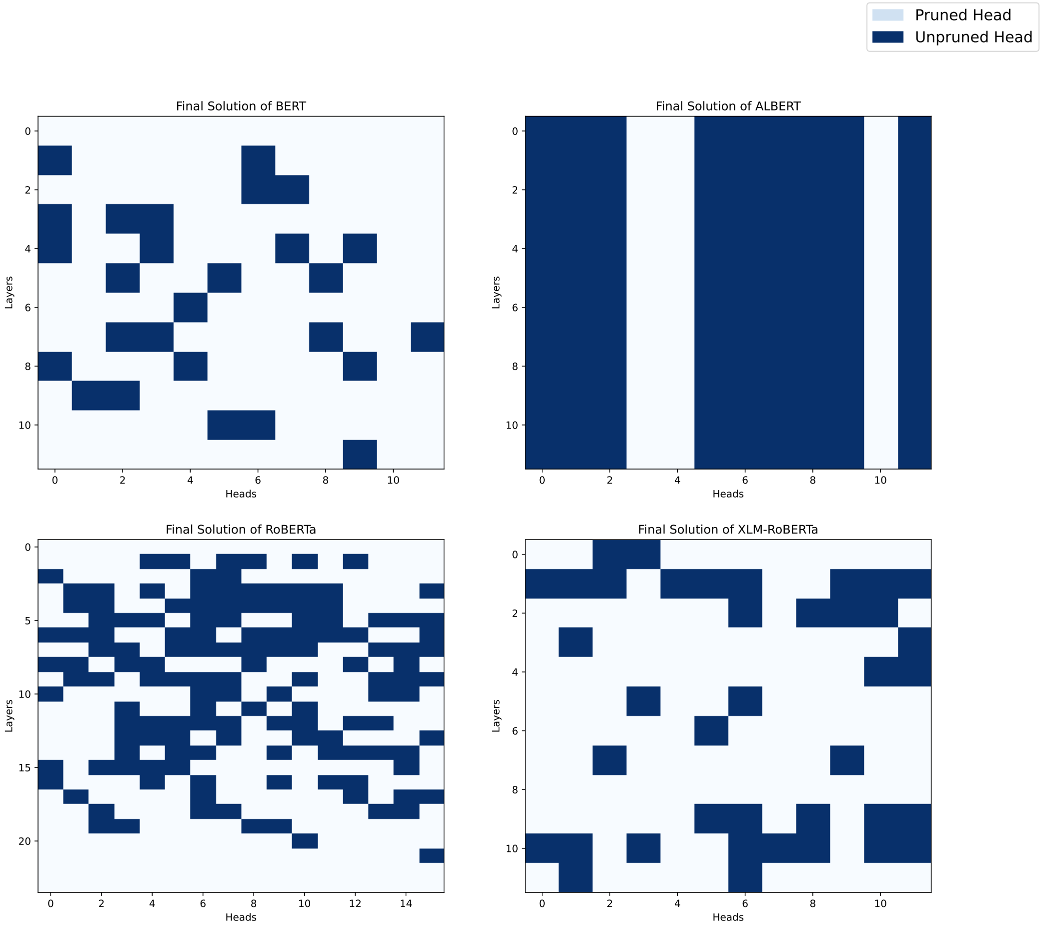

Figure 5: Final pruning masks across models. BERT shows dispersed retention; ALBERT shows vertical bands due to parameter sharing; RoBERTa/XLM-RoBERTa display selective, layer-dependent retention.

5 Limitations and Future Work

Greedy-Gnorm trades stability for extra compute: each pruning step requires a backward pass on a small calibration set to refresh $Q/K/V$ gradients, making it costlier than one-shot scoring (roughly $\mathcal{O}(N\,B)$ for N heads and B batches). The approach also assumes the calibration distribution is representative of deployment; mismatch can mis-rank heads. Our implementation is head-only (no feedforward neural network or token pruning), so it cannot remove all redundancy. Gradients can be noisy for small batches or long contexts.

Future work includes scaling to larger LLMs and additional architectures. Extending to multi-dimensional pruning that jointly selects heads and feedforward neural network width, and integrating structural removal so parameter cuts translate to latency gains. We also plan to combine Greedy-Gnorm with quantization and distillation for compounding compression, and to study alternative scalar objectives (e.g., task loss vs. logit norms).

6 Conclusion

We introduced Greedy-Gnorm, a head-pruning framework that scores heads using the elementwise product of the ℓ_2 norms of their Q/K/V gradient matrices, and dynamically recomputes head importance after each removal. A simple mask-based expansion ensures that gradient matrix is comparable across pruning steps, and an ε-rectified approach to evaluating AE is employed as safeguard against numerical underflow. Numerical experiments with BERT, ALBERT, RoBERTa, and XLM-RoBERTa demonstrate that Greedy-Gnorm consistently retains higher accuracy than the AE baseline at the same number of heads pruned. These results show promise for our greedy approach to head pruning, suggesting that dynamic gradient-based scoring offers an interesting alternative to static importance measures, and providing a foundation for integrating head pruning with broader model compression strategies.

Author Contributions

Yuxi Guo: Conceptualization, Derivations, Numerical experiments, Computer code, Results interpretation, Writing – original draft, Writing – and review & editing. **Paul Sheridan**: Supervision, Results interpretation, Writing – and review & editing. All authors reviewed the results and approved the final version of the manuscript.

Acknowledgments and Disclosure of Funding

This work was supported by Mitacs through the Mitacs Globalink Research Internship program.

Appendix A. Model Parameter Statistics Before and After Pruning

This appendix summarizes the parameter composition and post-pruning statistics of representative transformer models evaluated in this study. Tables 3–5 report detailed layer-wise parameter counts, storage sizes (in megabytes), and accuracy before and after pruning with Greedy-Gnorm. These results illustrate the magnitude of parameter reduction achieved while maintaining competitive accuracy across different architectures.

Table 3: ALBERT parameters before/after pruning with Greedy-Gnorm (MB=megabytes).

	Before Pruning			After Pruning		
Layer	Params	Share (%)	MB	Params	Share (%)	MB
Model	11,685,891	100.00	44.58	11,095,491	100.00	42.33
ALBERT	11,683,584	99.98	44.57	11,093,184	99.98	42.32
Embeddings	3,906,048	33.43	14.90	3,906,048	35.20	14.90
Encoder	7,186,944	61.50	27.42	6,596,544	59.45	25.16
Pooler	590,592	5.05	2.25	590,592	5.32	2.25
Classifier	2,307	0.02	0.01	2,307	0.02	0.01
Weight	2,304	0.02	0.01	2,304	0.02	0.01
Bias	3	0.00	0.00	3	0.00	0.00
Accuracy	84.48%			77.76%		

Table 4: RoBERTa parameters before/after pruning with Greedy-Gnorm (MB=megabytes).

	Before Pruning			After Pruning		
Layer	Params	Share (%)	MB	Params	Share (%)	MB
Model	355,361,794	100.00	1355.60	291,089,474	100.00	1110.42
RoBERTa	354,310,144	99.70	1351.59	290,037,824	99.64	1106.41
Embeddings	52,000,768	14.63	198.37	52,000,768	17.86	198.37
Encoder	302,309,376	85.07	1153.22	238,037,056	81.77	908.04
Classifier	1,051,650	0.30	4.01	1,051,650	0.36	4.01
Dense	1,049,600	0.30	4.00	1,049,600	0.36	4.00
Out_proj	2,050	0.00	0.01	2,050	0.00	0.01
Accuracy	87.80%			86.40%		

Table 5: XLM-RoBERTa parameters before/after pruning with Greedy-Gnorm (MB=megabytes).

	Before Pruning			After Pruning		
Layer	Params	Share (%)	MB	Params	Share (%)	MB
Model	278,059,028	100.00	1060.71	257,395,028	100.00	981.88
RoBERTa	277,453,056	99.78	1058.40	256,789,056	99.76	979.57
Embeddings	192,398,592	69.19	733.94	192,398,592	74.75	733.94
Encoder	85,054,464	30.59	324.46	64,390,464	25.02	245.63
Classifier	605,972	0.22	2.31	605,972	0.24	2.31
Dense	590,592	0.21	2.25	590,592	0.23	2.25
Out_proj	15,380	0.01	0.06	15,380	0.01	0.06
Accuracy	99.73%			90.97%		

Appendix B. Collapsed Gradient Handling after Structural Pruning

Our pruning implementation is based on the TextPruner framework, which performs structural pruning of transformer attention heads rather than simple weight masking. When a head is pruned, TextPruner physically removes its corresponding query, key, value, and output projection blocks from the model. As a result, the associated gradient tensors also shrink in dimensionality so that each pruned head effectively collapses the gradient matrix by eliminating its corresponding sub-blocks.

After pruning, the dimensionality of per-head gradient blocks decreases as certain attention heads are removed. This collapse poses a challenge for maintaining consistent gradient statistics across pruning iterations. The reduced vectors cannot be directly compared to their pre-pruning counterparts. Their length and positional correspondence within the layer both change after pruning. To ensure that gradient-based importance measures remain meaningful over time, we expand each shrunken gradient vector back to its original dimensionality using the binary mask M, inserting zeros at pruned positions. This reconstruction preserves the spatial alignment of gradient entries across pruning steps, allowing the expected gradient-norm signal $S(n)$ to remain comparable and stable as the model structure evolves.

Appendix C. Attention Entropy Rectification

AE is susceptible to numerical underflow because attention probabilities often become extremely small when the sequence length is large. This section formalizes our ε-rectified entropy variants and explains why the chosen rectification preserves the relative head ordering used for pruning.

C.1 Epsilon Fine-Tuning

When the number of tokens is large, attention values are dispersed across many positions, and $log(0)$ may appear in the entropy computation. To address this issue, we introduce

ε-rectified variants of the AE:

$$A(a_1, a_2, \ldots, a_n) = -\sum_{i=1}^{n} a_i \log a_i, \tag{22}$$

$$B(a_1, a_2, \ldots, a_n) = -\sum_{i=1}^{n} a_i \log(a_i + \varepsilon), \tag{23}$$

$$C(a_1, a_2, \ldots, a_n) = -\sum_{i=1}^{n} (a_i + \varepsilon) \log(a_i + \varepsilon). \tag{24}$$

By construction, A is the original entropy, while B and C are its ε-rectified forms. Their relationships can be expressed as:

$$C - A = (C - B) + (B - A). \tag{25}$$

Here ε is a small positive constant such that $a_i + \varepsilon < 1$. It follows that $C - B > 0$ and $B - A < 0$. The term $B - A < 0$ is straightforward, so we omit its proof. Given that $\sum_i a_i = 1$ and $\varepsilon \ll \frac{1}{2} \leq \frac{n-1}{n}$, we can show:

$$\left(\prod_i (a_i + \varepsilon)\right)^{\frac{1}{n}} < \frac{n\varepsilon + 1}{n} < 1, \tag{26}$$

which implies $C - B > 0$.

In practice, we adopt C as our entropy variant because it is always greater than B and thus provides stronger numerical stability (no NaNs) while maintaining head ranking consistency. When A suffers from underflow, B eliminates the log-zero issue, but its entropy value itself may still approach zero. Using C ensures that the entropy computation remains numerically stable even when a_i is extremely small.

C.2 Invariant Pruning Order

Let

$$F(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{n} a_i \log a_i \tag{27}$$

be continuous on $\mathbb{R}^+$. Suppose two attention distributions $P_1 = (x_1, \ldots, x_n)$ and $P_2 = (y_1, \ldots, y_n)$ satisfy $\sum_i x_i = \sum_i y_i = 1$, and define

$$\delta = F(P_2) - F(P_1) > 0. \tag{28}$$

We now examine whether this inequality still holds after ε-rectification. Define

$$P_1' = (x_1 + \varepsilon, \ldots, x_n + \varepsilon), \tag{29}$$
$$P_2' = (y_1 + \varepsilon, \ldots, y_n + \varepsilon), \tag{30}$$

and let

$$\delta' = F(P_2') - F(P_1'). \tag{31}$$

We ask: does $\delta' > 0$ still hold?

Consider the segment connecting P_1 and P_2 in n-dimensional space, parameterized as

$$\vec{l} = (l_1, l_2, \ldots, l_n) = \frac{\vec{P_1P_2}}{\|\vec{P_1P_2}\|} = \frac{\vec{P_1'P_2'}}{\|\vec{P_1'P_2'}\|}. \tag{32}$$

The difference δ can be expressed as the line integral of the directional derivative of F:

$$\delta = F(P_2) - F(P_1) = \int_{P_1}^{P_2} \nabla F \cdot \vec{l}\, dl > 0, \tag{33}$$

where the gradient is

$$\nabla F = (\log a_1 + 1, \log a_2 + 1, \ldots, \log a_n + 1). \tag{34}$$

Thus,

$$\nabla F \cdot \vec{l} = \sum_{i=1}^{n} l_i \log a_i. \tag{35}$$

Since F is continuous and smooth on $\mathbb{R}^+$, for sufficiently small ε the perturbed gradient remains in a local neighborhood of the original:

$$\nabla F' \cdot \vec{l} = \sum_{i=1}^{n} l_i \log(a_i + \varepsilon) \in N_r(\nabla F \cdot \vec{l}), \tag{36}$$

where r controls the allowed deviation in the integral value. Consequently,

$$\delta' = F(P_2') - F(P_1') = \int_{P_1'}^{P_2'} \nabla F' \cdot \vec{l}\, dl > 0. \tag{37}$$

This result shows that within a sufficiently small ε-neighborhood, the sign of δ remains unchanged, and hence the relative ordering of AE values is preserved. Therefore, using the rectified form C allows AE-based pruning to remain stable and monotonic—avoiding numerical underflow without altering the ranking of heads used for pruning.

References

Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. Controlling computation versus quality for neural sequence models, 2020. URL `https://arxiv.org/abs/2002.07106`.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116, 2019. URL `http://arxiv.org/abs/1911.02116`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Self-attention attribution: Interpreting information interactions inside transformer. *CoRR*, abs/2004.11207, 2020. URL https://arxiv.org/abs/2004.11207.

Qiaozhi He and Zhihua Wu. Efficient LLM inference with Kcache, 2024. URL https://arxiv.org/abs/2404.18057.

William Held and Diyi Yang. Shapley head pruning: Identifying and removing interference in multilingual transformers, 2022. URL https://arxiv.org/abs/2210.05709.

Aaron Klein, Jacek Golebiowski, Xingchen Ma, Valerio Perrone, and Cedric Archambeau. Structural pruning of pre-trained language models via neural architecture search, 2024. URL https://arxiv.org/abs/2405.02267.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M. Rush. Block pruning for faster transformers, 2021. URL https://arxiv.org/abs/2109.04838.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942, 2019. URL http://arxiv.org/abs/1909.11942.

Chonghan Lee, Md Fahim Faysal Khan, Rita Brugarolas Brufau, Ke Ding, and Vijaykrishnan Narayanan. Token and head adaptive transformers for efficient natural language processing. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na, editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4575–4584, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.404/.

Heejun Lee, Geon Park, Youngwan Lee, Jaduk Suh, Jina Kim, Wonyoung Jeong, Bumsik Kim, Hyemin Lee, Myeongjae Jeon, and Sung Ju Hwang. A training-free sub-quadratic cost transformer model serving framework with hierarchically pruned attention, 2025. URL https://arxiv.org/abs/2406.09827.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. Differentiable subset pruning of transformer heads, 2023. URL https://arxiv.org/abs/2108.04657.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Zejian Liu, Fanrong Li, Gang Li, and Jian Cheng. EBERT: Efficient BERT inference with dynamic structured pruning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4814–4823, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.425. URL https://aclanthology.org/2021.findings-acl.425/.

J. S. McCarley, Rishav Chakravarti, and Avirup Sil. Structured pruning of a BERT-based question answering model, 2021. URL https://arxiv.org/abs/1910.06360.

Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Delight: Deep and light-weight transformer, 2021. URL https://arxiv.org/abs/2008.00623.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *CoRR*, abs/1905.10650, 2019. URL http://arxiv.org/abs/1905.10650.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation, 2024. URL https://arxiv.org/abs/2407.14679.

Archit Parnami, Rahul Singh, and Tarun Joshi. Pruning attention heads of transformer models using a* search: A novel approach to compress big NLP architectures. *CoRR*, abs/2110.15225, 2021. URL https://arxiv.org/abs/2110.15225.

Ofir Press, Noah A. Smith, and Omer Levy. Improving transformer models by reordering their sublayers, 2020. URL https://arxiv.org/abs/1911.03864.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024. URL https://arxiv.org/abs/2404.02258.

Kyuhong Shim, Iksoo Choi, Wonyong Sung, and Jungwook Choi. Layer-wise pruning of transformer attention heads for efficient language modeling. *CoRR*, abs/2110.03252, 2021. URL https://arxiv.org/abs/2110.03252.

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, Chenhan Yu, Wei-Chun Chen, Hayley Ross, Oluwatobi Olabiyi, Ashwath Aithal, Oleksii Kuchaiev, Daniel Korzekwa, Pavlo Molchanov, Mostofa Patwary, Mohammad Shoeybi, Jan Kautz, and Bryan Catanzaro. LLM pruning and distillation in practice: The Minitron approach, 2024. URL https://arxiv.org/abs/2408.11796.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP, 2019. URL https://arxiv.org/abs/1906.02243.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models, 2024. URL https://arxiv.org/abs/2405.05254.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *CoRR*, abs/1905.09418, 2019. URL http://arxiv.org/abs/1905.09418.

Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, February 2021. doi: 10.1109/hpca51647.2021.00018. URL http://dx.doi.org/10.1109/HPCA51647.2021.00018.

Jiing-Ping Wang, Ming-Guang Lin, An-Yeu, and Wu. Latte: Low-precision approximate attention with head-wise trainable threshold for efficient transformer, 2024. URL https://arxiv.org/abs/2404.07519.

Wenxuan Wang and Zhaopeng Tu. Rethinking the value of transformer components, 2020. URL https://arxiv.org/abs/2011.03803.

Ziqing Yang, Yiming Cui, and Zhigang Chen. TextPruner: A model pruning toolkit for pre-trained language models. In Valerio Basile, Zornitsa Kozareva, and Sanja Stajner, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 35–43, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-demo.4. URL https://aclanthology.org/2022.acl-demo.4.

Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *Proceedings of the AAAI conference on artificial intelligence*, pages 3143–3151, 2022.

Biao Zhang, Deyi Xiong, and Jinsong Su. Accelerating neural transformer via an average attention network, 2018. URL https://arxiv.org/abs/1805.00631.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. Finercut: Finer-grained interpretable layer pruning for large language models, 2024. URL https://arxiv.org/abs/2405.18218.

Yibo Zhong and Yao Zhou. Rethinking low-rank adaptation in vision: Exploring head-level responsiveness across diverse tasks, 2024. URL https://arxiv.org/abs/2404.08894.