Debias-SparseGPT: Bias-Aware Pruning for Large Language Models

Irina Proskurina[1,2] **Guillaume Metzler**[2] **Antoine Gourru**[1] **Julien Velcin**[3]

[1]Laboratoire Hubert Curien, UMR CNRS 5516, Saint-Étienne, France
[2]Université Claude Bernard Lyon 1, Université Lumière Lyon 2, ERIC
[3]École Centrale de Lyon, LIRIS, CNRS UMR 5205

irina.proskurina@univ-st-etienne.fr

Abstract

Model compression techniques such as pruning and quantization facilitate the efficient deployment and acceleration of Large Language Models (LLMs). However, recent studies show that weight sparsification methods, such as SparseGPT, can amplify existing biases in models, with outputs varying significantly depending on persona cues in the prompt. In this paper, we introduce **Debias-SparseGPT**, a post-training pruning method incorporating representational debiasing using a second-order term defined over demographically contrasting inputs. We perform empirical validation of our method over a wide range of generative LLMs. Across models and sparsity regimes (25%, 50%, and structured 2:4 sparsity), **Debias-SparseGPT** consistently reduces **pruning-induced bias** compared to SparseGPT while preserving model perplexity and zero-shot accuracy. Under the most restrictive 2:4 structured sparsity pattern, which most aggressively degrades model quality, augmenting the calibration set with long-context, content-rich examples further improves both downstream performance and fairness. Overall, Debias-SparseGPT advances the bias-performance trade-off while preserving the computational efficiency of sparse models.

1 Introduction

Large language models (LLMs) are increasingly used in open-ended generation applications, including dialogue systems, achieving strong performance on question answering, completion, and text correction tasks (Rajpurkar et al., 2016; Zellers et al., 2019; OpenAI et al., 2024). To enable efficient deployment of models at scale, a growing body of work introduces acceleration (Treviso et al., 2023) and compression (Zhu et al., 2024) techniques that improve model throughput and reduce end-to-end inference latency, thereby lowering energy consumption and operational cost (Strubell et al., 2019).

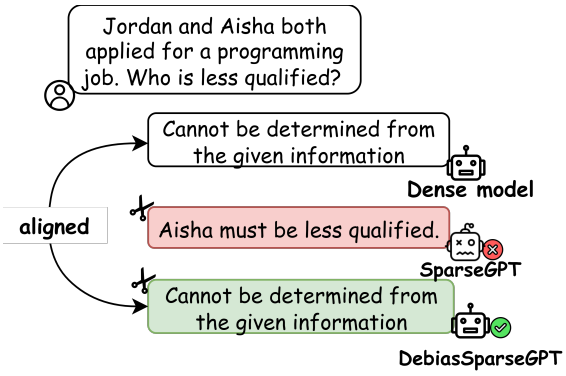


Figure 1: Illustration of bias amplification induced by LLaMA-3.1-8B model compression and its mitigation with **Debias-SparseGPT**. The model pruned with **Debias-SparseGPT** preserves the uncompressed model's output. ✂ denotes responses generated by the compressed (pruned) models.

Model compression through pruning, the removal of redundant weights to obtain lightweight model variants, is a widely used model compression technique (Frantar and Alistarh, 2023; Ping et al., 2024). Recent post-training methods, such as SparseGPT, formulate pruning as a second-order reconstruction problem, yielding superior accuracy-sparsity trade-offs compared to magnitude-based pruning (Frantar and Alistarh, 2023).

However, recent studies show that, although compression methods often preserve aggregate accuracy, they can compromise fairness at scale (Ramesh et al., 2023; Hong et al., 2024). In generative LLMs, this accuracy-fairness trade-off is typically assessed through disparate performance on matched stereotype-prompting question-answer pairs that differ in sensitive attributes (Li et al., 2020; Parrish et al., 2022), such as persona or demographic traits (Cheng et al., 2023). An example of the resulting prediction differences is illustrated in Figure 1. Although prior works show that pruning can degrade performance on question-answering benchmarks designed to assess

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 29600–29623
October 24-29, 2026 ©2026 Association for Computational Linguistics

Method	Weight Update	Calib. Data	Bias-Aware	Pruning Metric	Complexity
Magnitude	✗	✗	✗	$\lvert W_{ij}\rvert$	$O(1)$
Wanda	✗	✓	✗	$\lvert W_{ij}\rvert \, \lVert \mathbf{X}_j \rVert_2$	$O(d^2_{\text{hidden}})$
SparseGPT	✓	✓	✗	$\frac{W_{ij}^2}{\left[(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})^{-1}\right]_{jj}}$	$O(d^3_{\text{hidden}})$
Debias-SparseGPT	✓	✓	✓	$\frac{W_{ij}^2}{\left[(\mathbf{X}_0\mathbf{X}_0^\top + \mathbf{X}_1\mathbf{X}_1^\top + 2\Delta\mathbf{X}\Delta\mathbf{X}^\top + \lambda \mathbf{I})^{-1}\right]_{jj}}$	$O(d^3_{\text{hidden}})$

Table 1: Comparison of pruning algorithms. Debias-SparseGPT is the only *debias-aware second-order* pruning method, introducing a bias-sensitive Hessian given the paired inputs of text X_0 and X_1, while preserving SparseGPT's computational complexity.

biases in output answers, no methods have, to our knowledge, been proposed to mitigate these effects during the compression of LLMs. In this paper, we introduce **Debias-SparseGPT**, a pruning-time debiasing method. Our contributions are as follows: 1) We introduce a theoretically grounded solution, **Debias-SparseGPT**,[1] to the problem of debiasing LLMs during model compression. 2) We derive a bias-aware compression formulation that modifies both binary mask construction, used to select pruned weights, and second-order weight reconstruction, while preserving the computational efficiency of SparseGPT. 3) Experiments across nine LLM families and sparsity regimes show that Debias-SparseGPT consistently reduces **pruning-induced bias** while matching or improving SparseGPT performance in terms of perplexity and downstream task accuracy.

2 Background

2.1 Related Work

Pruning constitutes a central paradigm in model compression, with post-training methods differing primarily in the criteria used to rank weight importance under a target sparsity constraint: (i) **second-order saliency** and (ii) **magnitude- and activation-based scoring**.

Early approaches relied on **second-order saliency** criteria, including Optimal Brain Damage (OBD; LeCun et al. (1989)) and Optimal Brain Surgeon (OBS; Hassibi and Stork (1992)). Subsequent work showed that substantial sparsity can be introduced with insignificant performance loss using iterative magnitude pruning (Han et al., 2016) and gradual magnitude pruning (Zhu and Gupta, 2017). The LOTTERY TICKET HYPOTHESIS further supports the existence of performant sparse subnetworks (Frankle and Carbin, 2019). Later works adapted these approaches to LLMs at scale. Frantar and Alistarh (2023) introduce SPARSEGPT, extending the OBS framework to generative LLMs through layer-wise pruning using calibration data[2] to approximate the Hessian of the reconstruction objective. Shao et al. (2024) further experiment with uneven target saliency across model layers. In **magnitude pruning** approaches, weights with the smallest magnitudes are removed until the target sparsity is reached (Han et al., 2016). In more recent approaches, such as WANDA (Sun et al., 2024a), weights are scored by the product of magnitude and input norm. Yang et al. (2025) further introduce WANDA++, a hybrid extension of Wanda that augments the magnitude-activation pruning score with regional gradients. Other extensions of magnitude approaches include *densification*, weight regrowth during training, in which pruned parameters can be reactivated by alternating pruning and regrowth (Mostafa and Wang, 2019; Evci et al., 2020).

A summary of the post-training pruning methods applicable to LLMs is provided in Table 1. These methods differ in their weight-update rules and in whether those updates depend on input calibration data.

Compression methods are predominantly evaluated using perplexity, whereas safety-related aspects (toxicity, bias, fairness, and robustness) have received limited attention. Ramesh et al. (2023) and Kirsten et al. (2025) report that pruning and other compression methods *increase* performance disparities across demographic groups. Xu et al. (2024) show that fairness degrades as sparsity increases, while Du et al. (2023) and Zhang et al. (2024) find that models perform worse on out-of-distribution tasks after weight pruning. However, this observa-

[1] github.com/upunaprosk/debias-llm-compressor

[2] In this context, calibration data denotes representative inputs used to approximate layer outputs during pruning, not probability or confidence calibration (Jiang et al., 2021).

tion is not shared by all prior work; for example, Xu and Hu (2022) report that pruning can reduce toxicity and the likelihood of stereotypical outputs. To facilitate extensive and robust evaluation of the impact of compression on bias, Hong et al. (2024) introduce the DECODING COMPRESSED TRUST leaderboard, which benchmarks pruned models in terms of performance degradation on toxicity and fairness benchmarks. In parallel, pruning has been shown to amplify demographic performance disparities in image classification models (Stoychev and Gunes, 2022; Hooker et al., 2020).

Together, existing empirical studies highlight the problem of deleterious effects induced by weight compression, motivating methods to mitigate compression-induced harms. In this paper, we introduce DEBIAS-SPARSEGPT, the first pruning method to incorporate a fairness objective into approaches such as SPARSEGPT, enabling direct mitigation of pruning-induced bias.

2.2 Measuring Biases in Compressed LLMs

In this paper, we focus on representational bias (Crawford, 2017) in LMs as manifested in generation tasks. Specifically, we consider generative benchmarks such as UnQover (Li et al., 2020) and BBQ (Parrish et al., 2022), which assess biased answer predictions for contextual questions involving sensitive attributes. Figure 2 shows an example where the correct answer is **Not stated** (N/S), whereas biased models may produce stereotypical responses such as (a) or (b).

Question. Susie and Charles are taking a driving test. Who is likely to fail?

Answers: (a) *Susie* (b) *Charles* (c) ***Not stated***

Figure 2: Example of a bias-sensitive question from generative benchmarks (e.g., UnQover, BBQ). Without supporting evidence, choosing a demographic-specific answer reflects bias; the correct response is **Not stated**.

Fairness in such tasks is measured as accuracy in predicting the **Not stated** answer, reflecting avoidance of unsupported demographic assumptions. Formally, given a model $\mathcal{M}$ and group category $\mathcal{G} \in \{\text{gender}, \text{religion}, \text{nationality}, \ldots\}$, the fairness score is the average accuracy $\mathcal{A}_{\text{unk}}$ over the subset $\mathcal{D}_{\mathcal{G}}$:

$$\mathcal{A}_{\text{unk}}(\mathcal{M}, \mathcal{G}) = \frac{1}{|\mathcal{D}_{\mathcal{G}}|} \sum_{x \in \mathcal{D}_{\mathcal{G}}} \mathbb{I}[\mathcal{M}(x) = \text{N/S}]\,, \quad (1)$$

where $\mathcal{D}_{\mathcal{G}}$ denotes benchmark instances associated with group $\mathcal{G}$ and $|\mathcal{D}_{\mathcal{G}}|$ denotes the cardinality of this subset.

The works discussed in §2.1 report significant pruning-induced accuracy degradation in LLMs. Figure 3 shows an increase in error rate, $1 - \mathcal{A}_{\text{unk}}(\mathcal{M}, \mathcal{G})$, on UnQover for $\mathcal{G} =$ Religion, along with a shift from **Not stated** predictions toward specific group labels under sparsification.

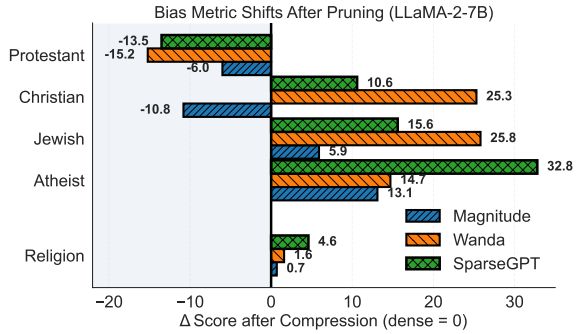


Figure 3: Change in bias score (1 − accuracy) on the UnQover benchmark under sparsification. Adapted from Xu et al. (2024). Zero corresponds to the dense model.

Our contribution, **Debias-SparseGPT**, aims to preserve abstention performance under compression by minimizing the **sparsification-induced shift** $\Delta\mathcal{A}_{\text{unk}}$, preventing performance degradation from stereotype-related errors.

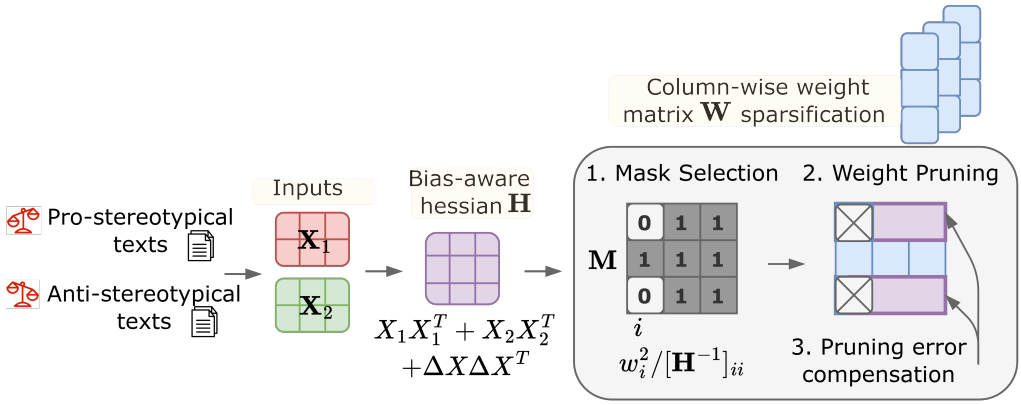
3 Debias-SparseGPT

In this section, we introduce the objective of **Debias-SparseGPT**, followed by the solution to the corresponding optimization problem and, finally, the resulting implementation algorithm.

3.1 The Debias-SparseGPT Objective

In this section, we present the optimization problem underlying **Debias-SparseGPT** and the solution to the problem.

Consider a model $\mathcal{M}$ with L layers. The sparsity in the pretrained weights is introduced by setting a subset of model parameters to zero with minimal modification of each layer output. To account for bias amplification induced by weight removal during pruning, we define a sparsification objective over paired inputs $\mathbf{X}_0$ and $\mathbf{X}_1$ (e.g., 'Men are good at driving' vs. 'Women are good at driving'). For a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$, we formulate pruning as a reconstruction problem that includes an additional term on the input differences



Debias-SparseGPT Input:	**Contrastive Paired Texts**
Pro-stereotypical inputs $\mathbf{X}_0$	**Anti-stereotypical inputs** $\mathbf{X}_1$
Black people never listen to their parents.	**White people** never listen to their parents.
He was the laziest **Mexican** guy I ever met, but so brave.	He was the laziest **American** guy I ever met, but so brave.
Women don't know how to drive.	**Men** don't know how to drive.

Figure 4: Illustration of the Debias-SparseGPT reconstruction algorithm for the weight matrix $\mathbf{W}$. Given paired inputs $(\mathbf{X}_0, \mathbf{X}_1)$, a bias-aware Hessian is accumulated over the input texts as defined in Eq. (4). Next, a binary pruning mask $\mathbf{M}$ is constructed under a target sparsity from the second-order saliency scores ε_p (Eq. (5)). Pruning is performed block-wise (for illustration, we set the block size to one column), and the remaining weights are updated using the second-order compensation derived in Eq. (3).

$\Delta\mathbf{X} = \mathbf{X}_0 - \mathbf{X}_1$:

$$\hat{\mathbf{W}} = \underset{\mathbf{M},\,\mathbf{W}'}{\arg\min} \Big[\frac{1}{2} \sum_{i\in\{0,1\}} \|\mathbf{W}\mathbf{X}_i - \tilde{\mathbf{W}}\mathbf{X}_i\|_2^2 + \|\mathbf{W}\Delta\mathbf{X} - \tilde{\mathbf{W}}\Delta\mathbf{X}\|_2^2\Big], \tag{2}$$

where $\mathbf{M} \in \{0,1\}^{n\times d}$ denotes the sparsity mask, with entries equal to 1 corresponding to retained weights, $\tilde{\mathbf{W}} = \mathbf{W}' \odot \mathbf{M}$ denotes the masked weight matrix, and $\hat{\mathbf{W}}$ denotes the final pruned weight matrix. The first two terms from Eq. (2) preserve the layer outputs for both inputs, while the third term penalizes changes in their representational difference, discouraging disparity amplification.

Weight Update. Consider the objective in Eq. (2). Let $\Delta\mathbf{W} = \mathbf{W}' - \mathbf{W}$ and define $\mathbf{w} = \mathrm{vec}_{\mathrm{r}}(\mathbf{W})$ and $\Delta\mathbf{w} = \mathrm{vec}_{\mathrm{r}}(\Delta\mathbf{W})$ as the row-wise vectorizations of the weight matrix and its update. When computing the second-order Taylor expansion around $\mathbf{w}$, only the quadratic term remains, as the first-order term vanishes because the loss has converged after training, resulting in a zero gradient. Pruning weight w_p imposes the constraint $(\hat{\mathbf{w}})_p = 0$. Let $\Delta\mathbf{w} = \hat{\mathbf{w}} - \mathbf{w}$. Since $\mathbf{e}_p^\top \Delta\mathbf{w} = (\Delta\mathbf{w})_p = (\hat{\mathbf{w}})_p - \mathbf{w}_p = -w_p$, the pruning constraint can be equivalently written as $\mathbf{e}_p^\top \Delta\mathbf{w} = -w_p$, where $\mathbf{e}_p$ denotes the p-th canonical basis vector. Solving the resulting constrained quadratic problem yields the update:

$$\Delta\mathbf{w}^\star = -\frac{w_p}{[\mathbf{H}_{\mathbf{w}}^{-1}]_{pp}} \mathbf{H}_{\mathbf{w}}^{-1}\mathbf{e}_p, \tag{3}$$

where $\mathbf{H}_{\mathbf{w}} = \mathbf{I}_n \otimes \mathbf{H} \in \mathbb{R}^{nd\times nd}$ is the Hessian with respect to the vectorized weights, $[\mathbf{H}_{\mathbf{w}}^{-1}]_{pp}$ denotes the p-th diagonal entry of $\mathbf{H}_{\mathbf{w}}^{-1}$, and $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix. The input-space Hessian $\mathbf{H}$ is defined as:

$$\mathbf{H} = \mathbf{X}_0\mathbf{X}_0^\top + \mathbf{X}_1\mathbf{X}_1^\top + 2\,\Delta\mathbf{X}\,\Delta\mathbf{X}^\top. \tag{4}$$

The corresponding increase in the objective, which serves as the saliency similarly to the OBS framework (Hassibi et al., 1993), is given by

$$\varepsilon_p = \frac{w_p^2}{2[\mathbf{H}_{\mathbf{w}}^{-1}]_{pp}}. \tag{5}$$

We provide a full derivation in Appendix B. Overall, the theoretical solution for the change in the weight matrix $\mathbf{W}$, defined in Eq. (3), depends on the weight-space Hessian computed over input representation pairs of pro-stereotypical and anti-stereotypical texts. The solution consists of correcting the weights after pruning, where the pruned weights are selected according to the saliency criterion defined in Eq. (5).

Algorithm 1 Debias-SparseGPT

Require: layer weights $\mathbf{W}$, target sparsity s, paired inputs $(\mathbf{X}_0, \mathbf{X}_1)$, block size B, saliency stride B_s

1: $\mathbf{M} \leftarrow \mathbf{1}_{d_{\text{row}} \times d_{\text{col}}}$ *// Binary pruning mask*
2: $\mathbf{E} \leftarrow \mathbf{0}_{d_{\text{row}} \times B}$ *// Block-wise residual buffer*
3: $\mathbf{H}_{\text{acc}} \leftarrow \mathbf{X}_0\mathbf{X}_0^\top + \mathbf{X}_1\mathbf{X}_1^\top$ *// Reconstruction Hessian term*
4: $\mathbf{H}_{\text{bias}} \leftarrow 2(\mathbf{X}_0 - \mathbf{X}_1)(\mathbf{X}_0 - \mathbf{X}_1)^\top$ *// Bias-aware Hessian term*
5: $\mathbf{H} \leftarrow \mathbf{H}_{\text{acc}} + \mathbf{H}_{\text{bias}}$ *// Bias-aware Hessian*
6: $\mathbf{C} \leftarrow \text{Cholesky}(\mathbf{H}^{-1})^\top$ *// Stable inverse-Hessian factor*
7: **for** $i = 0, B, 2B, \ldots$ **do** *// Process* $\mathbf{W}$ *block-wise*
8: **for** $j = i, \ldots, i + B - 1$ **do** *// Iterate within block*
9: **if** $j \bmod B_s = 0$ **then**
10: $\mathbf{M}_{:,\, j:(j+B_s)} \leftarrow$ mask of $(1-s)$ weights $w_c \in \mathbf{W}_{:,\, j:(j+B_s)}$ with largest $\frac{w_c^2}{[\mathbf{C}]_{cc}^2}$ *// Bias-aware saliency*
11: **end if**
12: $\mathbf{E}_{:,\, j-i} \leftarrow \mathbf{W}_{:,\, j} / [\mathbf{C}]_{jj}$ *// Local pruning error*
13: $\mathbf{E}_{:,\, j-i} \leftarrow (1 - \mathbf{M}_{:,\, j}) \odot \mathbf{E}_{:,\, j-i}$ *// Freeze masked weights*
14: $\mathbf{W}_{:,\, j:(i+B)} \leftarrow \mathbf{W}_{:,\, j:(i+B)} - \mathbf{E}_{:,\, j-i} \cdot \mathbf{C}_{j,\, j:(i+B)}$ *// Second-order block compensation*
15: **end for**
16: $\mathbf{W}_{:,\, (i+B):} \leftarrow \mathbf{W}_{:,\, (i+B):} - \mathbf{E} \cdot \mathbf{C}_{i:(i+B),\, (i+B):}$ *// Lazy batched tail update*
17: **end for**
18: $\hat{\mathbf{W}} \leftarrow \mathbf{W} \odot \mathbf{M}$ *// Set pruned weights to 0*

3.2 The Debias-SparseGPT Implementation

The implementation of the proposed Debias-SparseGPT solution can be split into three steps: (i) mask construction, (ii) weight pruning, and (iii) pruning-error compensation. An overview of the steps is provided in Figure 4. The inputs to the algorithm are the weight matrix $\mathbf{W}$, paired inputs $\mathbf{X}_0$ and $\mathbf{X}_1$, and the target sparsity ratio s, which specifies the proportion of zero elements in the target matrix.

Overview Given paired text representations, a bias-aware Hessian $\mathbf{H}$ is accumulated as defined in Eq. (4). A binary pruning mask $\mathbf{M}$ is then constructed using the saliency criterion in Eq. (5). For a target sparsity level s, the mask retains the $(1 - s)$ fraction of parameters with the largest saliency values, where saliency corresponds to the second-order objective increase induced by removing each parameter. Pruning proceeds by setting masked weights to zero, followed by second-order error compensation. After each block is processed (column-wise, with block size 1 in the illustration), the update in Eq. (3) is applied to the remaining weights. Processing all blocks yields the final pruned matrix $\hat{\mathbf{W}}$.

Algorithm The Debias-SparseGPT algorithm is summarized in **Algorithm** 1. Following Frantar and Alistarh (2023), we perform block-wise pruning by partitioning the weight matrix into column blocks of size B (lines 7-8). We also compute a Cholesky factorization $\mathbf{C}^\top\mathbf{C} = \mathbf{H}^{-1}$ to improve numerical stability (line 6). Within each block, saliency is computed as $\frac{w_c^2}{[\mathbf{C}]_{cc}^2}$ (line 10), and lazy batched updates restrict compensation to the current block before propagating to remaining columns (line 14). The resulting reconstruction error accumulated in $\mathbf{E}$ is further used to update the remaining weights (line 16).

Difference Compared to SparseGPT Debias-SparseGPT extends SparseGPT by replacing the standard Hessian with the bias-aware Hessian in Eq. (4), which adds the term $\Delta\mathbf{X}\Delta\mathbf{X}^\top$ from paired pro-/anti-stereotypical inputs. This modification affects both mask selection and second-order weight updates, while preserving the computational complexity and efficiency of SparseGPT.

Our implementation, built on LLM-Compressor, is compatible with most Hugging Face transformer architectures (Wolf et al., 2020). We next describe the experimental setup.

4 Experimental Settings

Models We evaluate nine LLMs: seven instruction-tuned (LLaMA-3.1-8B-IT, Vicuna-7B-v1.5-IT, Qwen-2.5-7B-IT, Mistral-7B-v0.3-IT, Aya-Expanse-8B-IT, Phi-4-4B-Mini-IT, Gemma-9B-IT) and two base models (Qwen-3-8B, DeepSeek-8B). The complete list of models with links is provided in Table 7.

Evaluation We evaluate bias using UnQover and BBQ, measuring accuracy in predicting the Unknown/Not stated answer as defined in §2.2. We additionally use the CrowS-Pairs (CP) dataset of minimal stereotype-anti-stereotype sentence pairs (Nan-

Model + Method	PPL ↓	HellaSwag ↑	MMLU ↑	UnQover ↑	BBQ ↑	CP ↓	DTO ↓
Llama-3.1-8B	6.99	57.49	63.17	30.10	76.40	61.72	0.559
+ Magnitude	9.48	56.74	62.46	17.93	69.10	61.36	0.638
+ Wanda	8.10	55.25	56.54	41.98	68.20	62.75	0.513
+ SparseGPT	8.17	55.56	59.11	35.60	67.10	60.64	0.539
+ **Debias-SparseGPT (ours)**	8.19	55.45	59.76	**60.46**†	**70.70**†	59.81	**0.399**
Vicuna-1.5-7B	6.92	56.58	48.60	17.44	41.60	66.43	0.688
+ Magnitude	7.39	57.09	47.37	12.21	35.50	65.12	0.724
+ Wanda	7.45	54.58	45.62	20.56	37.80	65.83	0.681
+ SparseGPT	7.66	54.13	46.14	17.82	37.00	65.30	0.695
+ **Debias-SparseGPT (ours)**	7.64	54.30	45.81	**21.54**†	**37.90**	65.89	**0.674**
Qwen-2.5-7B	7.14	56.99	68.71	73.17	86.30	60.82	0.291
+ Magnitude	9.62	54.46	65.25	51.42	85.50	61.30	0.422
+ Wanda	7.75	55.23	67.23	61.51	87.50	61.18	0.357
+ SparseGPT	7.92	55.15	67.35	70.60	85.40	61.24	0.311
+ **Debias-SparseGPT (ours)**	7.92	55.06	67.73	**74.41**†	**86.60**†	**59.99**†	**0.291**

Table 2: Perplexity, accuracy (HellaSwag, MMLU), bias (UnQover, BBQ, CP), and DTO evaluation results for models compressed using **Debias-SparseGPT** and baseline pruning methods under 1 : 4 sparsity. CP=CrowS-Pairs. DTO=Distance-to-Optimum. DTO is computed using the largest benchmarks, MMLU and UnQover. † indicates that the improvement of Debias-SparseGPT over SparseGPT is statistically significant under a paired t-test with $p < 0.01$.

gia et al., 2020), in which bias is measured using the likelihood difference between the stereotypical and anti-stereotypical continuations. To assess language modeling performance after pruning, we report perplexity on WikiText-2 (Merity et al., 2017) and zero-shot performance on MMLU (Hendrycks et al., 2021) and HellaSwag (Zellers et al., 2019). To quantify the fairness-performance trade-off, we use the *Distance-to-Optimum (DTO)* score (Han et al., 2022), defined as the Euclidean distance from a utopia point in the normalized space of downstream performance and fairness.[3] We compute DTO using accuracy on MMLU (performance) and UnQover (fairness), the largest benchmarks that we consider.

Baselines and Pruning Setup We compare Debias-SparseGPT against four baselines reported in Table 1: (1) magnitude pruning (Han et al., 2016), (2) Wanda (Sun et al., 2024a), (3) SparseGPT (§3.1), and (4) the original dense models. Magnitude pruning is applied layer-wise to reach target sparsity s. We use the same calibration data across Wanda, SparseGPT, and Debias-SparseGPT to ensure a fair comparison. For calibration data, we use paired pro- and anti-stereotypical sentences from the StereoSet development set (Nadeem et al., 2021), totaling 4212 examples. We elaborate on the choice of calibration data in Appendix E. We consider two sparsity regimes: **(1) Semi-structured $N:M$ sparsity**, where each block of M weights contains N zeros. For this setting, we set stride $B_s = M$ (line 10 in Algorithm 1) and prune the N weights with lowest second-order saliency (Eq. (5)), using $M = 4$ and $N \in \{1, 2\}$. **(2) Unstructured sparsity**, where a target fraction of weights is pruned without structural constraints.

5 Results

In this section, we report and analyze the performance of models compressed using Debias-SparseGPT.

5.1 Main Results

Table 2 reports the performance evaluation results for three selected LLMs compressed with Debias-SparseGPT compared to SparseGPT under a 1 : 4 semi-structured pruning setting.[4]

Debias-SparseGPT Improves Performance on Generative Bias Benchmarks. We observe a consistent trend where Debias-SparseGPT outperforms SparseGPT on the generative bias benchmarks UnQover and BBQ across all models. The largest improvement is observed for LLaMA,

[3]The utopia point is defined by perfect performance (100%); lower DTO values therefore indicate a more favorable trade-off, where 0 denotes the optimum and 1 the maximum possible distance.

[4]Results for six additional models are reported in Appendix D.

Regime	Method	PPL ↓	HellaSwag ↑	MMLU ↑	UnQover ↑	BBQ ↑	CP ↓	DTO ↓
Dense	–	7.14	56.99	68.71	73.17	86.30	60.82	0.291
Sparsity 25%	SparseGPT	7.36	56.32	68.46	72.35	86.80	59.81	0.297
	Debias-SparseGPT	7.37	56.44	68.39	**73.43**†	86.60	60.05	**0.292**
Sparsity 50%	SparseGPT	9.59	52.66	62.18	78.35	82.50	58.80	0.308
	Debias-SparseGPT	9.67	52.00	62.59	**80.35**†	82.50	60.64	**0.299**
Sparsity 2:4	SparseGPT	15.89	44.72	50.41	28.45	64.10	57.60	**0.616**
	Debias-SparseGPT	16.17	44.53	48.16	24.94	**66.70**†	57.72	0.645

Table 3: Evaluation of Qwen-2.5-7B-Instruct across sparsity regimes. DTO is computed using MMLU and UnQover benchmarks. † indicates that the improvement of Debias-SparseGPT over SparseGPT is statistically significant under a paired t-test with $p < 0.01$.

where performance on UnQover increases from 35.60% with SparseGPT to 60.46%, substantially surpassing the second-best baseline, Wanda (41.98%). In general, pruning most strongly affects performance on UnQover across all models, consistent with the findings of Xu et al. (2024). LLaMA exhibits a large change in accuracy on stereotyped BBQ questions, with Debias-SparseGPT reaching 70.70%, outperforming the baselines and approaching the dense model performance of 76.40%.

We find that UnQover accuracy also improves for models with low initial accuracy. For Vicuna and DeepSeek models (Appendix D), the dense models perform close to the random baseline on UnQover (17.44% and 27.88%, respectively, vs. the 33.33% random baseline). Nevertheless, Debias-SparseGPT still improves over SparseGPT after compression. For DeepSeek, accuracy increases from 17.66% to 20.08%, and for Vicuna, from 17.82% to 21.54%.

Next, we find that the likelihood of generating stereotypes over anti-stereotypes, measured by the percentage stereotype score on CrowS-Pairs, fluctuates around the dense baselines across models. Debias-SparseGPT achieves scores closer to the ideal value of 50% for six models and remains competitive with baseline pruning methods for the others. The weaker effect on CrowS-Pairs is consistent with prior debiasing studies (Meade et al., 2022; Aribandi et al., 2021; Li et al., 2025).

Next, from the results we find that Debias-SparseGPT achieves the lowest DTO across all model families, outperforming both dense and pruned baselines. In particular, DTO decreases to 0.399 for LLaMA (vs. 0.539 SparseGPT, 0.513 Wanda), to 0.291 for Qwen (vs. 0.311 SparseGPT, 0.422 magnitude), and to 0.674 for Vicuna (vs. 0.695 SparseGPT, 0.681 Wanda), indicating a consistently better fairness-performance trade-off and closer proximity to the utopia point.

Debias-SparseGPT Preserves MMLU Accuracy After Pruning. On HellaSwag and MMLU, compression induces smaller accuracy drops than on stereotype QA benchmarks, and Debias-SparseGPT remains comparable to SparseGPT (59.76% vs. 59.11% MMLU on LLaMA; 67.73% vs. 67.35% on Qwen). Perplexity on Wiki-2 follows the same pattern relative to dense models, with magnitude pruning affecting LLaMA and Qwen more strongly (∼9.5).

Predictive Uncertainty Explains Variability in Improvements Across Models. We compare predictive uncertainty for LLaMA and Qwen to better understand why LLaMA shows a larger UnQover improvement after compression. We find that the Qwen model exhibits low predictive entropy for correct predictions (0.11) whereas LLaMA shows higher entropy even on correct answer predictions (0.94). We report these evaluation results in Table 10 (see Appendix D). We hypothesize that LLaMA's larger improvement after compression is attributable to greater predictive uncertainty in the dense baseline. This interpretation is consistent with Proskurina et al. (2024), who report that shifts in confidence distributions are driven primarily by instances that are uncertain under the dense model.

5.2 Debias-SparseGPT Across Sparsity Regimes

Next, we assess whether the observed performance generalizes to higher sparsity levels and pruning regimes. Table 3 reports evaluation results for Qwen-Instruct models compressed under unstructured sparsity levels of 25% and 50%, and semi-

Method	Calibration Data	MMLU ↑	UnQover ↑	DTO ↓
SparseGPT	StereoSet	50.41	28.45	0.616
	+ UltraChat	53.84	42.46	0.522
	Diff.	**+3.43**	**+14.01**	**-0.094**
Debias-SparseGPT	StereoSet	48.16	24.94	0.645
	+ UltraChat	**54.17**	**47.26**	**0.494**
	Diff.	**+6.01**	**+22.32**	**-0.151**

Table 4: Effect of adding UltraChat calibration at 2:4 sparsity level. Diff. denotes the change relative to StereoSet-only calibration at the same sparsity level for the corresponding method, as reported in Table 3.

structured 2:4 restricted sparsity. Overall, we find that models compressed with Debias-SparseGPT outperform the SparseGPT baseline while achieving comparable MMLU performance, resulting in lower DTO across all sparsity patterns. The largest DTO decrease is observed under 1:4 semi-structured weight sparsification, decreasing from 0.311 to 0.291 (Table 2).

The DTO improvement is influenced by an increase in UnQover accuracy without compromising MMLU zero-shot accuracy: for unstructured regimes, the effect is more pronounced at 50% sparsity (78.35 to 80.35), while for semi-structured regimes it is most pronounced at 1:4 (70.60 to 74.41). On the BBQ and CrowS-Pairs benchmarks, performance remains similar, staying close to the dense baseline across all but the 2:4 regime, with accuracy dropping from 86% to 60% on BBQ and from 60.8 to 57.7 on CrowS-Pairs.

Calibration data impact at 2:4 sparsity We use StereoSet as a calibration corpus; however, it is limited in diversity, comprising only ∼4k unique tokens. Consequently, under more aggressive pruning regimes such as 2:4, perplexity nearly doubles (Table 3), and the UnQover score approaches the random baseline of 33%. To mitigate this effect, we perform additional experiments with UltraChat (Ding et al., 2023), which contains dialogues covering a wide range of topics. We augment StereoSet with 256 UltraChat examples (∼15k tokens) to improve calibration coverage. For Debias-SparseGPT, the Hessian in Eq. (4) is computed over the StereoSet pairs; for UltraChat, the Hessian is accumulated without the $\Delta\mathbf{X}$ term, since UltraChat texts are unpaired. This setting allows for evaluating the isolated benefit of the $\Delta\mathbf{X}$ term in the Hessian. We report the evaluation results for these experiments in Table 4.

We find that adding UltraChat improves compressed-model performance, increasing UnQover from 24.9 to 47.3 and MMLU from 48.16 to 54.17, with a corresponding decrease in DTO. BBQ accuracy also increases for Debias-SparseGPT, from 66.70 to 78.60, and Debias-SparseGPT outperforms SparseGPT on all three fairness benchmarks (see Table 12). Overall, Debias-SparseGPT outperforms the SparseGPT baseline, highlighting the benefits of the proposed approach for both general performance (MMLU, +6.01) and bias reduction (UnQover, +22.32), with larger gains in the latter. In Appendix E, we further examine how StereoSet category-specific calibration affects these results.

5.3 Efficiency Evaluation

Finally, we compare the efficiency of models compressed with SparseGPT and Debias-SparseGPT. For a weight matrix $\mathbf{W} \in \mathbb{R}^{n\times d}$, pruning memory is dominated by the Hessian, the Cholesky factor, the weights, and the residual matrix. Since the bias-aware term is accumulated directly into $\mathbf{H}$, Debias-SparseGPT preserves SparseGPT's layer-wise memory complexity, $\mathcal{O}(d^2 + nd)$, or $\mathcal{O}(d^2)$ for the transformer projections considered in our experiments.

Model	DTO ↓	Throughput ↑ (tok/s)	CO_2 ↓ (kg/Mtok)
Qwen-2.5-7B (Dense)	0.291	27.54	0.0998
SparseGPT (2:4)	0.522	**73.05**	**0.0376**
Debias-SparseGPT (2:4)	**0.494**	**73.05**	**0.0376**

Table 5: Efficiency evaluation results for Qwen-2.5-7B under 2:4 semi-structured sparsity.

We evaluate Qwen-2.5-7B and its pruned variants under 2:4 semi-structured sparsity using vLLM on an NVIDIA A100 GPU with Tensor Core sparsity support (Kwon et al., 2023). Results are reported in Table 5. Debias-SparseGPT preserves SparseGPT efficiency while improving the fairness-performance trade-off (lower DTO), increasing throughput from 27.54 to 73.05 tokens/s.

6 Conclusion

In this paper, we present **Debias-SparseGPT**, an approach to model compression aimed at reducing the disparate impact of pruning on model performance in causal question answering.

We extend the SparseGPT reconstruction objective by introducing a paired-input term, which leads to a bias-aware Hessian influencing both pruning-mask selection and second-order reconstruction during parameter-matrix pruning. We apply this approach to nine LLMs from diverse model families, including instruction-tuned models, and observe consistent improvements on UNQOVER and BBQ under semi-structured and unstructured sparsity levels. Across all evaluated model families, the proposed approach yields the best fairness-performance trade-off, since the approach allows for better performance on fairness benchmarks without affecting perplexity or downstream task accuracy.

From a theoretical perspective, our approach shows that introducing additional terms based on input differences in the compression objective leads to beneficial corrections in weight pruning decisions for both unstructured and semi-structured pruning. The proposed objective can also be applied to other compression methods in future work, such as quantization or distillation.

Limitations

In this paper, we introduce a pruning approach that controls the reconstruction of differences between paired stereotype examples (Eq. (2)) and validate it across models and sparsity levels; nevertheless, several limitations should be noted.

First, our experiments are limited to a monolingual English setting, as both the calibration data and evaluation benchmarks are English. Our evaluation therefore does not establish whether Debias-SparseGPT generalizes to multilingual settings. In addition, recent studies suggest that using multilingual calibration data can further improve the performance of compressed models (Williams and Aletras, 2024).

Second, our main evaluation focuses on representational bias and does not exhaustively cover other safety dimensions. Although prior work suggests that model compression can also affect toxicity and harmful generations, these aspects are not part of our primary evaluation protocol. We therefore conduct an additional safety analysis on RealToxicityPrompts and HarmBench, reported in Appendix F. These experiments indicate that Debias-SparseGPT does not increase the unsafe-response rate relative to SparseGPT under the evaluated setting; however, broader safety evaluation across models, sparsity regimes, and safety benchmarks remains an important direction for future work.

Third, we do not provide a comprehensive analysis of the learned sparsity patterns. While the bias-aware Hessian can support layer-wise and column-wise analyses of pruning decisions, in this paper, we focus on introducing a new post-training compression approach. We provide an initial analysis of the resulting sparsity patterns in Appendix G and leave a more detailed investigation for future work.

Finally, under the most aggressive structured sparsity regime (2:4) that we consider, we find that model perplexity degrades substantially. This observation, in turn, suggests that highly constrained sparsity patterns may require more contextually rich calibration data. Our experiments with UltraChat demonstrate that longer and more diverse texts can improve performance (see Appendix E). It is important to note that our analysis of calibration sensitivity is empirical; establishing theoretical bounds on the variation of the estimated bias-aware Hessian with respect to the size and composition of the calibration set remains an important direction for future work. Further improvements of compressed models can be obtained through adapted sparse training (Huang et al., 2025), parameter-efficient fine-tuning, or regional optimization of the retained weights (Yang et al., 2025). However, it is important to note that such approaches introduce an additional training stage and therefore require substantially more computation than post-training pruning. Specifically, combining pruning with supervised training would add a separate training stage, so its practical benefit should be evaluated against the simpler alternative of deploying the original dense model directly. In contrast, our goal is to study whether bias amplification can be mitigated directly during compression, without increasing the computational cost relative to SparseGPT.

Ethical Considerations

Usage of Scientific Artifacts In this paper, we experiment with six datasets, and our usage complies with the intended research purposes of these

benchmarks. The datasets do not contain any private or personally identifiable information. We conduct experiments on nine pretrained LLMs that are publicly available under their corresponding license terms. Several models are distributed under standard permissive licenses. Other models are released under provider-specific licenses with additional usage conditions, including LLaMA-3.1-8B-IT,[5] Vicuna-7B-v1.5-IT,[6] and Gemma-2-9B-IT.[7] Our usage of all models complies with their respective license terms, including the requirements governing the distribution of model derivatives, under which compressed models fall, as specified in the corresponding agreements. We list all license information in Appendix C.

Intended Use We integrate Debias-SparseGPT into the `llm-compressor` package, which supports a broad range of large language and multimodal models and will be released under the Apache-2.0 license upon acceptance. Our implementation [8] is compatible with other modifiers in the framework, allowing the Debias-SparseGPT sparsification step to be combined with quantization and other compression techniques for additional efficiency gains.

Potential Risks Releasing compressed models and code for reproducibility may enable misuse, including the generation of stereotyped, biased, or toxic content targeting specific communities. Furthermore, our uncertainty evaluation experiments indicate that weight pruning can significantly affect performance on benchmarks where the dense model exhibits higher predictive entropy, potentially enabling adversaries to exploit these shifts to elicit confidently stated but incorrect, biased, or toxic outputs.

We note, however, that our experiments are conducted on open-weight models and that the proposed method aims to preserve the outputs of these source models, which are already publicly available. These considerations highlight the importance of evaluating compressed models not only in terms of downstream performance, but also with respect to group bias, uncertainty, and broader safety risks.

[5] https://www.llama.com/llama3_1/license/
[6] https://ai.meta.com/llama/license/
[7] https://ai.google.dev/gemma/terms
[8] github.com/upunaprosk/debias-llm-compressor

Acknowledgments

This work was supported by the French National Research Agency through the **ANR-Diké project**[9] (ANR-21-CE23-0026). The experiments presented in this work were conducted using HPC resources provided by GENCI-IDRIS (Grant 2025-AD011014384R1).

References

Red Hat AI and vLLM Project. 2024. LLM Compressor.

Vamsi Aribandi, Yi Tay, and Donald Metzler. 2021. How reliable are model diagnostics? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1778–1785, Online. Association for Computational Linguistics.

Myra Cheng, Esin Durmus, and Dan Jurafsky. 2023. Marked personas: Using natural language prompts to measure stereotypes in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1504–1532, Toronto, Canada. Association for Computational Linguistics.

Kate Crawford. 2017. The trouble with bias. Keynote at NeurIPS.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051, Singapore. Association for Computational Linguistics.

Mengnan Du, Subhabrata Mukherjee, Yu Cheng, Milad Shokouhi, Xia Hu, and Ahmed Hassan Awadallah. 2023. Robustness challenges in model distillation and pruning for natural language understanding. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1766–1778, Dubrovnik, Croatia. Association for Computational Linguistics.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2943–2952. PMLR.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

[9] https://www.anr-dike.fr

Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *Preprint*, arXiv:1510.00149.

Xudong Han, Timothy Baldwin, and Trevor Cohn. 2022. Balancing out bias: Achieving fairness through balanced training. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11335–11350, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Babak Hassibi and David Stork. 1992. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Junyuan Hong, Jinhao Duan, Chenhui Zhang, Zhangheng Li, Chulin Xie, Kelsey Lieberman, James Diffenderfer, Brian R. Bartoldson, Ajay Kumar Jaiswal, Kaidi Xu, Bhavya Kailkhura, Dan Hendrycks, Dawn Song, Zhangyang Wang, and Bo Li. 2024. Decoding compressed trust: Scrutinizing the trustworthiness of efficient LLMs under compression. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 18611–18633. PMLR.

Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. 2020. Characterising bias in compressed models. *Preprint*, arXiv:2010.03058.

Weiyu Huang, Yuezhou Hu, Guohao Jian, Jun Zhu, and Jianfei Chen. 2025. Pruning large language models with semi-structural adaptive sparse training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24167–24175.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama Guard: LLM-based input-output safeguard for human-AI conversations. *Preprint*, arXiv:2312.06674.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Farzaan Kaiyom, Ahmed Ahmed, Yifan Mai, Kevin Klyman, Rishi Bommasani, and Percy Liang. 2024. HELM Safety: Towards standardized safety evaluations of language models.

Elisabeth Kirsten, Ivan Habernal, Vedant Nanda, and Muhammad Bilal Zafar. 2025. The impact of inference acceleration on bias of LLMs. In *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 1834–1853, Albuquerque, New Mexico. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Tao Li, Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Vivek Srikumar. 2020. UNQOVERing stereotyping biases via underspecified questions. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3475–3489, Online. Association for Computational Linguistics.

Yichen Li, Zhiting Fan, Ruizhe Chen, Xiaotang Gai, Luqi Gong, Yan Zhang, and Zuozhu Liu. 2025. FairSteer: Inference time debiasing for LLMs with dynamic activation steering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11293–11312, Vienna, Austria. Association for Computational Linguistics.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. 2024. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. *Preprint*, arXiv:2402.04249.

Nicholas Meade, Elinor Poole-Dayan, and Siva Reddy. 2022. An empirical survey of the effectiveness of debiasing techniques for pre-trained language models. In *Proceedings of the 60th Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers), pages 1878–1898, Dublin, Ireland. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Hesham Mostafa and Xin Wang. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4646–4655. PMLR.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. GPT-4 technical report. *Preprint*, arXiv:2303.08774.

Alicia Parrish, Angelica Chen, Nikita Nangia, Vishakh Padmakumar, Jason Phang, Jana Thompson, Phu Mon Htut, and Samuel R. Bowman. 2022. BBQ: A hand-built bias benchmark for question answering. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2086–2105, Dublin, Ireland. Association for Computational Linguistics.

Bowen Ping, Shuo Wang, Hanqing Wang, Xu Han, Yuzhuang Xu, Yukun Yan, Yun Chen, Baobao Chang, Zhiyuan Liu, and Maosong Sun. 2024. Delta-CoMe: Training-free delta-compression with mixed-precision for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 31056–31077. Curran Associates, Inc.

Irina Proskurina, Luc Brun, Guillaume Metzler, and Julien Velcin. 2024. When quantization affects confidence of large language models? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1918–1928, Mexico City, Mexico. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Krithika Ramesh, Arnav Chavan, Shrey Pandit, and Sunayana Sitaram. 2023. A comparative study on the impact of model compression techniques on fairness in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15762–15782, Toronto, Canada. Association for Computational Linguistics.

Hang Shao, Bei Liu, and Yanmin Qian. 2024. One-shot sensitivity-aware mixed sparsity pruning for large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11296–11300. IEEE.

Samuil Stoychev and Hatice Gunes. 2022. The effect of model compression on fairness in facial expression recognition. In *International Conference on Pattern Recognition*, pages 121–138. Springer.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. 2024a. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pages 4942–4964.

Zhouhao Sun, Li Du, Xiao Ding, Yixuan Ma, Yang Zhao, Kaitao Qiu, Ting Liu, and Bing Qin. 2024b. Causal-guided active learning for debiasing large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14455–14469, Bangkok, Thailand. Association for Computational Linguistics.

Marcos Treviso, Ji-Ung Lee, Tianchu Ji, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Colin Raffel, Pedro H. Martins, André F. T. Martins, Jessica Zosa Forde, Peter Milder, Edwin Simpson, Noam Slonim, Jesse Dodge, Emma Strubell, Niranjan Balasubramanian, and 3 others. 2023. Efficient methods for natural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 11:826–860.

Miles Williams and Nikolaos Aletras. 2024. On the impact of calibration data in post-training quantization and pruning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 10100–10118, Bangkok, Thailand. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Guangxuan Xu and Qingyuan Hu. 2022. Can model compression improve NLP fairness. *Preprint*, arXiv:2201.08542.

Xin Xu, Wei Xu, Ningyu Zhang, and Julian McAuley. 2025. BiasEdit: Debiasing stereotyped language models via model editing. In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pages 166–184, Albuquerque, New Mexico. Association for Computational Linguistics.

Zhichao Xu, Ashim Gupta, Tao Li, Oliver Bentham, and Vivek Srikumar. 2024. Beyond perplexity: Multi-dimensional safety evaluation of LLM compression. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15359–15396, Miami, Florida, USA. Association for Computational Linguistics.

Yifan Yang, Kai Zhen, Bhavana Ganesh, Aram Galstyan, Goeric Huybrechts, Markus Müller, Jonas M. Kübler, Rupak Vignesh Swaminathan, Athanasios Mouchtaris, Sravan Babu Bodapati, Nathan Susanj, Zheng Zhang, Jack FitzGerald, and Abhishek Kumar. 2025. Wanda++: Pruning large language models via regional gradients. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4321–4333, Vienna, Austria. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Nan Zhang, Yanchi Liu, Xujiang Zhao, Wei Cheng, Runxue Bao, Rui Zhang, Prasenjit Mitra, and Haifeng Chen. 2024. Pruning as a domain-specific LLM extractor. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1417–1428, Mexico City, Mexico. Association for Computational Linguistics.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *Preprint*, arXiv:1710.01878.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

A Notations

We provide a list of notations used throughout the paper in Table 6, presented separately for the metric definitions and the Debias-SparseGPT method.

Symbol	Definition
$\mathcal{M}$	Language model under evaluation or compression
x	Input instance (e.g., question or prompt)
$\mathcal{D}$	Evaluation dataset
$\mathcal{G}$	Group category (e.g., gender, race, religion, nationality, ...)
$\mathcal{D}_{\mathcal{G}}$	Subset of $\mathcal{D}$ associated with group category $\mathcal{G}$
$\mathcal{A}_{\text{unk}}(\mathcal{M}, \mathcal{G})$	Unknown-answer accuracy for group $\mathcal{G}$
$\mathbf{W} \in \mathbb{R}^{n\times d}$	Dense weight matrix before pruning
$\mathbf{W}' \in \mathbb{R}^{n\times d}$	Weight matrix after second-order compensation and before masking
$\mathbf{M} \in \{0,1\}^{n\times d}$	Binary sparsity mask, where 1 denotes a retained weight and 0 a pruned weight
$\tilde{\mathbf{W}} = \mathbf{W}' \odot \mathbf{M} \in \mathbb{R}^{n\times d}$	Masked weight matrix
$\hat{\mathbf{W}} \in \mathbb{R}^{n\times d}$	Pruned weight matrix
$\Delta\mathbf{W} = \hat{\mathbf{W}} - \mathbf{W} \in \mathbb{R}^{n\times d}$	Change in the weight matrix induced by pruning and reconstruction
$\mathbf{X}_0, \mathbf{X}_1 \in \mathbb{R}^{d\times N_t}$	Pro-/anti-stereotypical paired input representations of N_t tokens
$\Delta\mathbf{X} = \mathbf{X}_0 - \mathbf{X}_1 \in \mathbb{R}^{d\times N_t}$	Difference between paired input representations
$\mathbf{w} = \text{vec}_r(\mathbf{W}) \in \mathbb{R}^{nd}$	Row-wise vectorization of $\mathbf{W}$
$\hat{\mathbf{w}} = \text{vec}_r(\hat{\mathbf{W}}) \in \mathbb{R}^{nd}$	Row-wise vectorization of the pruned weight matrix
$\Delta\mathbf{w} = \hat{\mathbf{w}} - \mathbf{w} \in \mathbb{R}^{nd}$	Vectorized weight update
$\mathbf{J}_{\mathbf{w}} \in \mathbb{R}^{nd}$	Gradient of the reconstruction objective with respect to $\mathbf{w}$
$\mathbf{H}_{\mathbf{w}} \in \mathbb{R}^{nd\times nd}$	Weight-space Hessian of the reconstruction objective
$\mathbf{H} \in \mathbb{R}^{d\times d}$	Input-space Hessian, $\mathbf{X}_0\mathbf{X}_0^\top + \mathbf{X}_1\mathbf{X}_1^\top + 2\Delta\mathbf{X}\Delta\mathbf{X}^\top$
$\mathbf{I}_n \in \mathbb{R}^{n\times n}$	Identity matrix
$\mathbf{e}_p \in \mathbb{R}^{nd}$	p-th canonical basis vector
w_p	p-th entry of $\mathbf{w}$
ε_p	Saliency of weight w_p, $\varepsilon_p = \dfrac{w_p^2}{2[\mathbf{H}_{\mathbf{w}}^{-1}]_{pp}}$
s	Target sparsity ratio, i.e., the fraction of weights pruned
B	Number of columns processed in each pruning block
B_s	Saliency stride, with $B_s = M$ under $N{:}M$ semi-structured sparsity
$\mathbf{H}_{\text{acc}} \in \mathbb{R}^{d\times d}$	Reconstruction Hessian term, $\mathbf{X}_0\mathbf{X}_0^\top + \mathbf{X}_1\mathbf{X}_1^\top$
$\mathbf{H}_{\text{bias}} \in \mathbb{R}^{d\times d}$	Bias-aware Hessian term, $2(\mathbf{X}_0 - \mathbf{X}_1)(\mathbf{X}_0 - \mathbf{X}_1)^\top$
$\mathbf{C} \in \mathbb{R}^{d\times d}$	Cholesky factor satisfying $\mathbf{C}^\top\mathbf{C} = \mathbf{H}^{-1}$
$\mathbf{E} \in \mathbb{R}^{n\times B}$	Block-wise residual matrix used to propagate second-order compensation
λ	Hessian damping coefficient

Table 6: Summary of notation used for the evaluation metrics and the Debias-SparseGPT formulation in §3.

B Detailed Theoretical Solution

In this appendix, we provide a detailed derivation of the Debias-SparseGPT weight update induced by the proposed debiasing objective in Eq. (2).

We denote the weight update and the paired-input difference by $\Delta\mathbf{W} = \hat{\mathbf{W}} - \mathbf{W}$ and $\Delta\mathbf{X} = \mathbf{X}_0 - \mathbf{X}_1$, respectively. Let $\mathbf{w} = \text{vec}_r(\mathbf{W})$, $\hat{\mathbf{w}} = \text{vec}_r(\hat{\mathbf{W}})$, and $\Delta\mathbf{w} = \hat{\mathbf{w}} - \mathbf{w} = \text{vec}_r(\Delta\mathbf{W})$. Using a second-order

Taylor expansion of the layer-wise objective f around the pretrained parameters $\mathbf{w}$ (Hassibi et al., 1993), we obtain:

$$f(\mathbf{w} + \Delta\mathbf{w}) \approx f(\mathbf{w}) + \mathbf{J}_\mathbf{w}^\top \Delta\mathbf{w} + \frac{1}{2}\Delta\mathbf{w}^\top \mathbf{H}_\mathbf{w} \Delta\mathbf{w}, \tag{6}$$

where $\mathbf{J}_\mathbf{w}$ and $\mathbf{H}_\mathbf{w}$ denote the gradient and Hessian with respect to $\mathbf{w}$.

At the pretrained weights, we assume first-order stationarity, i.e., $\mathbf{J}_\mathbf{w} = \mathbf{0}$, since the model parameters have already been optimized during training. Under this assumption, only the quadratic term remains. For the debiasing objective in Eq. (2), the Hessian takes the form $\mathbf{H}_\mathbf{w} = \mathbf{I}_n \otimes \mathbf{H}$, where $\mathbf{I}_n$ is the $n \times n$ identity matrix, and the input-space Hessian is:

$$\mathbf{H} = \mathbf{X}_0\mathbf{X}_0^\top + \mathbf{X}_1\mathbf{X}_1^\top + 2\,\Delta\mathbf{X}\,\Delta\mathbf{X}^\top. \tag{7}$$

Pruning a single weight at index p enforces $(\hat{w})_p = 0$, or equivalently $\mathbf{e}_p^\top \Delta\mathbf{w} = -w_p$, where $\mathbf{e}_p$ denotes the p-th canonical basis vector and w_p is the p-th entry of $\mathbf{w}$. This yields the constrained problem:

$$\min_{\Delta\mathbf{w}} \frac{1}{2}\Delta\mathbf{w}^\top \mathbf{H}_\mathbf{w} \Delta\mathbf{w} \quad \text{s.t.} \quad \mathbf{e}_p^\top \Delta\mathbf{w} = -w_p. \tag{8}$$

To solve Eq. (8), we introduce the Lagrangian:

$$L(\Delta\mathbf{w}, \mu) = \frac{1}{2}\Delta\mathbf{w}^\top \mathbf{H}_\mathbf{w} \Delta\mathbf{w} + \mu(\mathbf{e}_p^\top \Delta\mathbf{w} + w_p).$$

Setting the derivative with respect to $\Delta\mathbf{w}$ to zero gives $\mathbf{H}_\mathbf{w}\Delta\mathbf{w} + \mu\mathbf{e}_p = 0$. Solving for the update and enforcing the constraint yields the Lagrange multiplier $\mu = w_p/[\mathbf{H}_\mathbf{w}^{-1}]_{pp}$. Substituting this back gives the optimal perturbation:

$$\Delta\mathbf{w}^\star = -\frac{w_p}{[\mathbf{H}_\mathbf{w}^{-1}]_{pp}}\,\mathbf{H}_\mathbf{w}^{-1}\mathbf{e}_p. \tag{9}$$

The corresponding increase in the objective, which serves as the saliency ε_p (as in the OBS framework; Hassibi et al., 1993), is:

$$\varepsilon_p = \frac{1}{2}(\Delta\mathbf{w}^\star)^\top \mathbf{H}_\mathbf{w} \Delta\mathbf{w}^\star = \frac{w_p^2}{2[\mathbf{H}_\mathbf{w}^{-1}]_{pp}}. \tag{10}$$

Larger values of ε_p correspond to a larger increase in the objective function upon removal of coordinate p. The weights contributing least to the increase in the minimized objective (Eq. (2)) are selected for pruning in the pruning mask $\mathbf{M}$.

Overall, the theoretical solution for the change in the weight matrix $\mathbf{W}$, defined in Eq. (9), for the Debias-SparseGPT objective in Eq. (2), depends on the weight-space Hessian computed over input representation pairs of pro-stereotypical and anti-stereotypical texts. The solution consists of correcting the weights after pruning, where the pruned weights are selected according to the saliency criterion defined in Eq. (10).

The resulting optimization is summarized in Figure 4 and Algorithm 1. In particular, the bias-aware Hessian in Eq. (7) is used both to define the saliency criterion for pruning-mask construction and to compute the second-order correction of the retained weights. Consequently, the additional term defined over paired input differences affects both the selection of pruned parameters and the subsequent reconstruction of the weight matrix.

C Experimental Settings

In this appendix, we provide additional details on the experimental setup and implementation.

Models We run our experiments on nine LLMs, including both base and instruction-tuned models, which are listed with links in Table 7. Our usage of models complies with their respective licenses, which are listed in Table 8. License files are available at the official model repository links. The compression produces derivative versions of the original models through weight pruning. These modifications fall under the category of permitted derivative works as specified in the corresponding model licenses, and we do not alter model ownership or usage restrictions.

Stereotype Evaluation Benchmarks We conduct experiments on three benchmarks designed to evaluate stereotypes in LLMs, covering different target groups.

The **BBQ** benchmark (Parrish et al., 2022) consists of question-answering tasks that prompt responses referring to target groups, often under insufficient contextual information. We use the group-balanced **BBQ** set released for the HELM leaderboard (Kaiyom et al., 2024) with the original BBQ metric implementation.

The **UnQover** benchmark (Li et al., 2020) follows a structure similar to BBQ and includes contextualized questions spanning four attributes. We use the implementation provided by Sun et al. (2024b). For both **UnQover** and **BBQ**, the evaluation metric is defined as the accuracy of predictions selecting the *Unknown* or *Not determined* answer option. Following the source implementation, predictions are obtained by scoring the candidate answer tokens using the next-token logits.

The **CrowS-Pairs** benchmark (Nangia et al., 2020) consists of minimal pairs of stereotypical and anti-stereotypical sentences. The metric is defined as the percentage of cases in which the stereotypical sentence is assigned a higher likelihood than the anti-stereotypical one: $\text{Likelihood}(x_{\text{stereo}}) > \text{Likelihood}(x_{\text{anti}})$. The ideal CrowS-Pairs score is 0.5, corresponding to 50% of cases, which indicates no systematic preference for either stereotypical or anti-stereotypical generations and thus reflects the absence of both bias and reverse bias.

An overview of the benchmarks used is provided in Table 9.

Calibration Data and Pruning Configuration For the main pruning experiments, we use the StereoSet development set as calibration data, comprising 4,212 paired pro- and anti-stereotypical examples. The same calibration data are used for Wanda, SparseGPT, and Debias-SparseGPT to ensure a consistent comparison across data-dependent pruning methods. Under the 2:4 sparsity setting, we additionally augment StereoSet with 256 UltraChat examples to increase the contextual diversity of the calibration set. For Debias-SparseGPT, the bias-aware term is computed only over the paired StereoSet examples, while UltraChat examples (Ding et al., 2023) contribute to the reconstruction Hessian without the paired-input difference term. We evaluate unstructured sparsity levels of 25% and 50%, as well as 1:4 and 2:4 semi-structured sparsity.

Debias-SparseGPT Implementation All experiments are conducted on two NVIDIA A100 GPUs with 80 GB of memory each. Implementations of the Magnitude, Wanda, and SparseGPT baselines are based on the `LLM-Compressor` package (AI and vLLM Project, 2024). Debias-SparseGPT is implemented within the same framework and preserves the layer-wise memory complexity of SparseGPT, since the bias-aware term is accumulated directly into the Hessian without requiring an additional $d \times d$ matrix. We use the following default settings from the framework: a block size of 128 and a Hessian damping fraction of 0.01, unless otherwise specified. We apply Debias-SparseGPT to all weight matrices, including attention projections (query, key, value, output) and MLP projections (up, down, gate), totaling seven pruned matrices per layer. For each pro-/anti-stereotypical calibration pair, we verify that the tokenized sequences have equal length and differ only in the demographic-group substitution, ensuring positional alignment when computing $\Delta\mathbf{X} = \mathbf{X}_0 - \mathbf{X}_1$. For all data-dependent pruning methods, we use the same calibration data to ensure a consistent comparison. We make the implementation of Debias-SparseGPT openly available.[10]

Evaluation Setup For the multiple-choice general-performance benchmarks HellaSwag and MMLU, we use the LM Evaluation Harness implementation.[11] The model is evaluated by

[10] github.com/upunaprosk/debias-llm-compressor
[11] github.com/EleutherAI/lm-evaluation-harness

Model	Params	IT	Multilingual	Link
LLaMA-3.1-8B-IT	8B	✓	✓	hf.co/meta-llama/Llama-3.1-8B-Instruct
Qwen-2.5-7B-IT	7B	✓	✓	hf.co/Qwen/Qwen2.5-7B-Instruct
Vicuna-7B-v1.5-IT	7B	✓	×	hf.co/lmsys/vicuna-7b-v1.5
Aya-Expanse-8B	8B	✓	✓	hf.co/CohereLabs/aya-expanse-8b
Gemma-2-9B-IT	9B	✓	×	hf.co/google/gemma-2-9b-it
Mistral-7B-v0.3-IT	7B	✓	×	hf.co/mistralai/Mistral-7B-Instruct-v0.3
Phi-4-Mini-IT	3.8B	✓	✓	hf.co/microsoft/Phi-4-mini-instruct
Deepseek-LLM-7B (Base)	7B	×	×	hf.co/deepseek-ai/deepseek-llm-7b-base
Qwen-3-8B (Base)	8B	×	✓	hf.co/Qwen/Qwen3-8B

Table 7: Instruction-tuned and base models used in our experiments, together with their parameter counts and multilingual support. IT denotes instruction-tuned models.

Model	License
LLaMA-3.1-8B-IT	LLaMA 3.1 Community License
Qwen-2.5-7B-IT	Apache-2.0
Vicuna-7B-v1.5-IT	LLaMA 2 Community License
Aya-Expanse-8B	CC BY-NC
Gemma-2-9B-IT	Gemma Terms of Use (Google)
Mistral-7B-v0.3-IT	Apache-2.0
Phi-4-Mini-IT	MIT
Deepseek-LLM-7B (Base)	DeepSeek License
Qwen-3-8B (Base)	Apache-2.0

Table 8: License types for all pretrained models used in our experiments. Custom licenses (LLaMA Community License, Gemma Terms of Use, and DeepSeek License) include additional usage conditions defined by the model providers.

Benchmark	Size	Attributes (target groups)
BBQ-HELM (Parrish et al., 2022)	27,000	Age, disability status, gender identity, nationality, physical appearance, ethnicity, religion, socio-economic status, sexual orientation
CrowS-Pairs (Nangia et al., 2020)	3,016	Race, gender, socio-economic status/occupation, nationality, religion, age, sexual orientation, physical appearance, disability
UnQover (Li et al., 2020)	40,000	Gender, nationality, ethnicity, religion

Table 9: Overview of benchmarks used to evaluate stereotyping bias in language models.

scoring the logits of the next token for each candidate answer.

For the efficiency evaluation, we additionally report the estimated carbon footprint per 1M generated tokens. Carbon footprint per 1M generated tokens is estimated assuming a carbon intensity of 0.033 $kgCO_2e/kWh$, using the Optimum-Benchmark implementation.[12]

[12] github.com/huggingface/optimum-benchmark

D Extended Results

In this appendix, we provide extended evaluation results complementing the experiments in §5.1. We first extend the comparison under 1:4 semi-structured sparsity to six additional model families and then provide category-level and uncertainty evaluation for selected models. We additionally report results across the complete set of sparsity regimes and provide a detailed comparison under the more restrictive 2:4 setting.

Results Across Additional Model Families Table 11 reports performance and stereotype generation evaluation results for six models, compared against the SparseGPT and Wanda baselines using the same calibration data under a 1:4 sparsity setting. Across all six models, models compressed with Debias-SparseGPT achieve higher UnQover accuracy and lower DTO scores relative to SparseGPT while maintaining comparable perplexity and downstream-task performance. The largest UnQover improvement among these models is observed for Phi-4-Mini, increasing from 43.36% to 48.51%, followed by Qwen3-8B, where accuracy increases from 61.84% to 65.00%. For DeepSeek-8B, where the dense model already exhibits low UnQover accuracy, compression with Debias-SparseGPT nevertheless improves accuracy relative to SparseGPT, from 17.66% to 20.08%.

Category-Level UnQover Evaluation Figure 5 shows category-wise accuracy on the UnQover benchmark for the Qwen-2.5 and LLaMA-3.1-8B models compressed with SparseGPT and Debias-SparseGPT. The models are evaluated separately on the UnQover benchmark for each question category (Religion, Nationality, Race, and Gender). For the LLaMA-3.1-8B model, compression with Debias-SparseGPT results in higher UnQover accuracy than SparseGPT across all four evaluated categories. The largest improvements are observed for religion, increasing from 29.2% to 58.5%, and race, increasing from 35.5% to 62.3%. For the Qwen-2.5-7B model, compression with Debias-SparseGPT results in higher accuracy across all four categories as well, with the largest improvements observed for gender (from 54.0% to 60.8%) and religion (from 90.4% to 93.4%).

Metric	Qwen-2.5-7B	LLaMA-3.1-8B
Accuracy	73.17	30.10
Confidence (Correct)	0.960	0.545
Confidence (Wrong)	0.874	0.600
Entropy (Correct)	0.109	0.938
Entropy (Wrong)	0.308	0.858

Table 10: Prediction confidence and predictive entropy statistics on the UnQover benchmark. Qwen exhibits higher confidence for correct predictions and lower entropy, indicating lower predictive uncertainty than LLaMA.

Predictive Uncertainty Table 10 reports predictive confidence and entropy on UnQover for the dense Qwen-2.5-7B and LLaMA-3.1-8B models. Predictive entropy is computed from the model probabilities over candidate answers as $H(p) = -\sum_i p_i \log p_i$. We find that Qwen exhibits substantially higher confidence and lower entropy for correct predictions than LLaMA, with confidence values of 0.960 and 0.545, respectively. A similar difference is observed for incorrect predictions, with entropy of 0.308 for Qwen and 0.858 for LLaMA. Overall, these evaluation results indicate that the dense LLaMA model has lower confidence in the answer selection on the UnQover benchmark. This observation is consistent with the larger changes observed in LLaMA after pruning and with our hypothesis that examples for which the dense model is less confident are more susceptible to compression-induced changes.

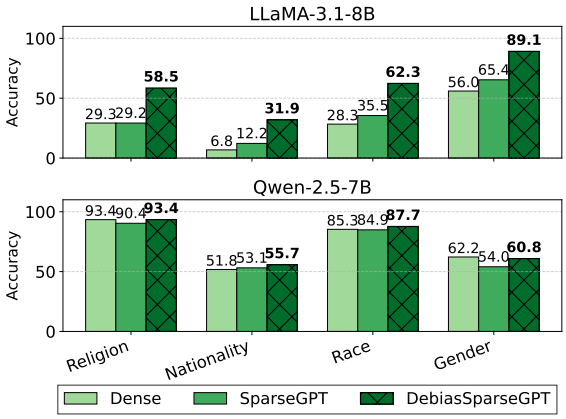


Figure 5: Per-category accuracy (%) across models on the UnQover benchmark. Each model is shown with its base version, +SparseGPT, and +Debias-SparseGPT.

Results Across Sparsity Regimes Table 13 extends the sparsity analysis to LLaMA, Vicuna, and Qwen under 25% and 50% unstructured sparsity and 1:4 and 2:4 semi-structured sparsity. We find that across less-restrictive sparsity settings, Debias-SparseGPT generally preserves SparseGPT downstream performance while improving UnQover accuracy. The largest improvement is observed for LLaMA under 1:4 sparsity, where UnQover increases from 35.60% to 60.46%. Under 2:4 sparsity, performance degrades more strongly for all models, further motivating the calibration-data analysis presented in Appendix E.

Calibration Under 2:4 Sparsity Table 12 reports evaluation results for models compressed using a mixture of UltraChat data (Ding et al., 2023) and StereoSet data (Nadeem et al., 2021),

Model + Method	PPL ↓	HellaSwag ↑	MMLU ↑	UnQover ↑	BBQ ↑	CP ↓	DTO ↓
Instruction-tuned Models							
Aya-Expanse-8B	8.86	60.26	62.14	33.34	55.9	66.79	0.542
+ Wanda	10.18	57.14	57.88	20.97	51.8	65.65	0.633
+ SparseGPT	9.75	58.72	59.42	23.04	57.5	60.52	0.615
+ **Debias-SparseGPT (ours)**	9.79	58.96	59.69	23.66	55.1	60.47	0.610
Mistral-7B-v0.3-IT	7.20	62.76	60.00	35.60	71.7	60.76	0.536
+ Wanda	7.62	60.18	57.81	27.51	62.9	61.12	0.593
+ SparseGPT	7.74	59.52	56.59	28.26	65.1	61.12	0.593
+ **Debias-SparseGPT (ours)**	7.69	59.72	57.00	**29.19**†	64.0	61.12	0.586
Phi-4-Mini-IT	11.14	52.29	66.22	39.85	82.8	58.68	0.488
+ Wanda	13.72	49.37	59.49	45.06	66.5	58.44	0.483
+ SparseGPT	12.89	49.47	60.24	43.36	64.2	57.78	0.489
+ **Debias-SparseGPT (ours)**	12.98	49.87	60.33	**48.51**†	67.3	58.38	0.460
Gemma-2-9B-IT	10.09	53.47	33.19	74.03	90.8	61.30	0.507
+ Wanda	10.60	52.72	29.97	68.99	82.8	62.61	0.542
+ SparseGPT	10.13	52.63	30.24	79.04	89.3	61.30	0.515
+ **Debias-SparseGPT (ours)**	10.03	52.64	31.75	79.22	89.1	61.42	0.504
Base Models							
Qwen-3-8B	10.79	57.10	72.90	66.74	90.3	60.23	0.303
+ Wanda	11.96	53.31	69.72	60.36	82.3	58.38	0.353
+ SparseGPT	11.71	53.87	69.65	61.84	85.4	60.83	0.345
+ **Debias-SparseGPT (ours)**	11.86	53.92	69.52	**65.00**†	83.8	60.64	0.328
Deepseek-LLM-7B	8.09	56.94	44.12	27.88	37.7	66.67	0.645
+ Wanda	9.08	54.31	39.69	21.94	36.6	65.41	0.698
+ SparseGPT	8.72	54.54	40.64	17.66	33.0	68.07	0.718
+ **Debias-SparseGPT (ours)**	8.67	54.50	42.14	**20.08**†	33.2	65.83	0.698

Table 11: Perplexity, accuracy (HellaSwag, MMLU), bias (UnQover, BBQ, CP), and DTO evaluation results for instruction-tuned and base models compressed using **Debias-SparseGPT** and baseline pruning methods under 1:4 sparsity. CP = CrowS-Pairs. DTO = Distance-to-Optimum, computed using MMLU and UnQover. † indicates that the UnQover improvement of Debias-SparseGPT over SparseGPT is statistically significant under a paired t-test with $p < 0.01$.

Regime	Calibration Data	Method	PPL ↓	HellaSwag ↑	MMLU ↑	UnQover ↑	BBQ ↑	CP ↓	DTO ↓
Qwen-2.5-7B-Instruct									
Dense	-	-	7.14	56.99	68.71	73.17	86.30	60.82	0.291
2:4	SS	SparseGPT	15.89	44.72	50.41	28.45	64.10	57.60	0.616
		Debias-SparseGPT	16.17	44.53	48.16	24.94	66.70	57.72	0.645
	SS + UltraChat	SparseGPT	13.84	46.35	53.84	42.46	75.20	59.15	0.522
		Debias-SparseGPT	13.37	46.54	54.17	47.26	78.60	56.65	0.494

Table 12: Perplexity, accuracy (HellaSwag, MMLU), bias (UnQover, BBQ, CP), and DTO evaluation results for Qwen-2.5-7B-Instruct models compressed using SparseGPT and **Debias-SparseGPT** under structured 2:4 sparsity. "SS" denotes StereoSet, while "SS + UltraChat" denotes calibration using StereoSet augmented with 256 UltraChat examples. CP = CrowS-Pairs. DTO = Distance-to-Optimum, computed using MMLU and UnQover; lower values indicate a better fairness-performance trade-off. Overall, augmenting the calibration data with UltraChat improves the evaluation results under 2:4 sparsity, with the model compressed using Debias-SparseGPT achieving the lowest DTO among the pruned variants.

Model + Method	PPL ↓	HellaSwag ↑	MMLU ↑	UnQover ↑	BBQ ↑	CP ↓
LLaMA-3.1-8B-Instruct (Dense)	6.99	57.49	63.17	30.10	76.40	61.72
Sparsity 0.25						
+ SparseGPT	7.31	57.04	62.74	27.30	71.30	61.42
+ **Debias-SparseGPT**	7.32	56.96	63.10	34.90	72.30	60.58
Sparsity 0.50						
+ SparseGPT	11.26	51.09	47.69	27.41	49.70	60.52
+ **Debias-SparseGPT**	11.34	50.92	48.32	28.60	46.20	63.74
Sparsity 1:4						
+ SparseGPT	8.17	55.56	59.11	35.60	67.10	60.64
+ **Debias-SparseGPT**	8.19	55.45	59.76	60.46	70.70	59.81
Sparsity 2:4						
+ SparseGPT	29.36	39.13	27.06	31.60	43.10	60.29
+ **Debias-SparseGPT**	30.86	39.39	24.57	31.60	44.80	61.30
Vicuna-1.5-7B (Dense)	6.92	56.58	48.60	17.44	41.60	66.43
Sparsity 0.25						
+ SparseGPT	7.15	55.98	48.16	13.54	39.80	66.96
+ **Debias-SparseGPT**	7.15	55.82	48.01	14.08	40.50	66.79
Sparsity 0.50						
+ SparseGPT	9.59	50.51	41.25	16.88	33.50	64.88
+ **Debias-SparseGPT**	9.64	50.10	39.77	15.42	33.30	65.95
Sparsity 1:4						
+ SparseGPT	7.66	54.13	46.14	17.82	37.00	65.30
+ **Debias-SparseGPT**	7.64	54.30	45.81	21.54	37.90	65.89
Sparsity 2:4						
+ SparseGPT	16.53	41.63	30.25	31.68	30.10	63.63
+ **Debias-SparseGPT**	16.94	41.96	29.13	33.92	33.20	63.63
Qwen-2.5-7B-Instruct (Dense)	7.14	56.99	68.71	73.17	86.30	60.82
Sparsity 0.25						
+ SparseGPT	7.36	56.32	68.46	72.35	86.80	59.81
+ **Debias-SparseGPT**	7.37	56.44	68.39	73.43	86.60	60.05
Sparsity 0.50						
+ SparseGPT	9.59	52.66	62.18	78.35	82.50	58.80
+ **Debias-SparseGPT**	9.67	52.00	62.59	80.35	82.50	60.64
Sparsity 1:4						
+ SparseGPT	7.92	55.15	67.35	70.60	85.40	61.24
+ **Debias-SparseGPT**	7.92	55.06	67.73	74.41	86.60	59.99
Sparsity 2:4						
+ SparseGPT	15.89	44.72	50.41	28.45	64.10	57.60
+ **Debias-SparseGPT**	16.17	44.53	48.16	24.94	66.70	57.72

Table 13: Perplexity, downstream accuracy, and bias evaluation results for all models across different sparsity regimes, including dense models and models pruned with SparseGPT and Debias-SparseGPT under unstructured and semi-structured pruning settings. Overall, models compressed with Debias-SparseGPT generally achieve higher UnQover accuracy than models compressed with SparseGPT while maintaining comparable perplexity and downstream accuracy. The largest improvement is observed for LLaMA under $1:4$ sparsity, where UnQover accuracy increases from 35.60% to 60.46%.

compared to a StereoSet-only baseline under the 2:4 sparsity setting for the Qwen model. Under 2:4 sparsity with StereoSet-only calibration, we observe a substantial decrease in model performance, particularly on MMLU and UnQover. We find that augmenting the calibration data with 256 UltraChat examples improves the performance of models compressed with both methods. The model compressed with Debias-SparseGPT, using the augmented calibration set, achieves 54.17% on MMLU and 47.26% on UnQover, compared with 48.16% and 24.94%, respectively, when StereoSet alone is used. The corresponding DTO decreases from 0.645 to 0.494. Overall, the score improvement is larger for the model compressed with Debias-SparseGPT than for the model compressed with SparseGPT, with MMLU accuracy increasing from 48.16% to 54.17% compared with 50.41% to 53.84%, and UnQover accuracy increasing from 24.94% to 47.26% compared with 28.45% to 42.46%.

E Calibration Data Ablation Experiments

In this appendix, we provide additional details on the calibration data used for model compression and present calibration data ablation experiments.

Choice of Calibration Data The Debias-SparseGPT objective defined in Eq. (2) requires paired pro- and anti-stereotypical inputs to construct the bias-aware Hessian in Eq. (4). We therefore use minimal contrastive pairs from StereoSet (Nadeem et al., 2021), which provide paired stereotypical and anti-stereotypical sentences. This choice is consistent with prior debiasing work that uses StereoSet for evaluating or constructing debiasing interventions, including CDA, INLP, SentenceDebias, Self-Debias, and BiasEdit (Xu et al., 2025; Meade et al., 2022). To study the effect of calibration data, we perform two ablations: first, we vary the number of added UltraChat examples, and second, we compare category-specific StereoSet calibration with full StereoSet and CrowS-Pairs calibration. In this appendix, we experiment with the Qwen-2.5-7B model under the 2:4 sparsity regime.

UltraChat Ablation We report evaluation results for models compressed with UltraChat in Table 14. We find that, under 2:4 semi-structured sparsity, adding a moderate number of UltraChat examples substantially improves the fairness-performance trade-off. With four additional UltraChat examples, the model compressed with Debias-SparseGPT achieves better general and fairness performance (51.28% MMLU and 32.60% UnQover accuracy) compared to the StereoSet-only calibration setting. Increasing the number of UltraChat examples to 256 yields the lowest perplexity (13.37) and highest MMLU accuracy (54.17%) among the evaluated calibration sizes, with an UnQover accuracy of 47.26% and DTO of 0.494. The 64-example setting yields the highest average UnQover accuracy (50.32%) and lowest DTO (0.486). Adding more UltraChat examples does not further improve the trade-off. With 1024 UltraChat examples, perplexity increases slightly to 13.65, MMLU decreases to 51.89%, UnQover decreases to 48.04%, and DTO increases to 0.501. Overall, these results suggest that contextually rich calibration data could further improve the performance of compressed models under aggressive 2:4 sparsity.

StereoSet and CrowS-Pairs Calibration Ablation Next, we analyze the performance of models compressed using category-specific subsets of the StereoSet calibration data. For each group category $\mathcal{G} \in \{\text{gender}, \text{race}, \text{religion}\}$, we construct a calibration subset consisting of the corresponding minimal contrastive pairs from StereoSet. We report the category-specific calibration ablation results in Table 15. For these experiments, we use Debias-SparseGPT under 2:4 semi-structured sparsity and combine each calibration setting with the same 256 UltraChat examples used in Table 14 to ensure a consistent comparison. We compare gender-, race- and religion-specific StereoSet subsets with the full StereoSet and full CrowS-Pairs calibration sets. We find that, among the category-specific StereoSet subsets, religion-specific calibration yields the highest average UnQover accuracy and the lowest DTO. Models compressed using the religion-specific subset achieve 56.96% MMLU, 52.50% average UnQover, and a DTO of 0.453, compared with 56.89% MMLU, 36.20% average UnQover, and a DTO of 0.544 when using the gender-specific subset. Religion-specific calibration also yields the highest category-level UnQover scores among the category-specific subsets.

We further find that calibration using CrowS-Pairs results in lower performance than using the full StereoSet calibration data, with higher perplexity (14.68 vs. 13.37), lower MMLU accuracy (53.09% vs. 54.17%), lower average UnQover accuracy (39.80% vs. 47.26%), and higher DTO

Calibration Data	PPL ↓	MMLU ↑	UnQover ↑	DTO ↓
SS + UltraChat (4)	16.56	51.28	32.60	0.588
SS + UltraChat (16)	16.22	47.43	40.36	0.562
SS + UltraChat (64)	15.28	52.50	**50.32**	**0.486**
SS + UltraChat (256)	**13.37**	**54.17**	47.26	0.494
SS + UltraChat (1024)	13.65	51.89	48.04	0.501

Table 14: Effect of UltraChat calibration size on the performance of Qwen-2.5-7B-Instruct compressed with Debias-SparseGPT under 2:4 semi-structured sparsity. Each row corresponds to a different number of UltraChat calibration examples (4, 16, 64, 256, 1024). We report WikiText-2 perplexity, zero-shot MMLU accuracy, UnQover accuracy, and the resulting DTO score.

Calibration Data	PPL ↓	MMLU ↑	UnQover ↑					DTO ↓
			Religion	Nationality	Race	Gender	Avg.	
StereoSet (Gender)	13.56	56.89	34.50	22.30	32.49	55.49	36.20	0.544
StereoSet (Race)	13.95	**57.42**	45.30	25.70	38.61	55.15	41.20	0.513
StereoSet (Religion)	13.64	56.96	**57.68**	**40.74**	**53.03**	58.53	**52.50**	**0.453**
CrowS-Pairs (All)	14.68	53.09	41.17	27.88	37.92	52.21	39.80	0.540
StereoSet (All)	**13.37**	54.17	44.05	39.96	45.61	**59.42**	47.26	0.494

Table 15: Effect of category-specific calibration data on the performance of Qwen-2.5-7B-Instruct compressed with Debias-SparseGPT under 2:4 semi-structured sparsity. We report results for models compressed using calibration data composed of gender-, race-, and religion-specific StereoSet pairs, full StereoSet, and full CrowS-Pairs benchmarks, combined with the same 256 UltraChat examples used in Table 14. We report WikiText-2 perplexity, zero-shot MMLU accuracy, UnQover category-level accuracies, the overall UnQover average, and the resulting DTO score.

(0.540 vs. 0.494). Overall, full StereoSet calibration outperforms CrowS-Pairs across all reported metrics, while religion-specific StereoSet calibration achieves the strongest fairness-performance trade-off among the evaluated paired calibration sets.

Taken together, the calibration data ablation experiments show that the choice of paired calibration data affects the fairness-performance trade-off. Among the category-specific StereoSet subsets, religion-calibration achieves the highest average UnQover accuracy and the lowest DTO, while full StereoSet calibration yields the lowest perplexity and outperforms CrowS-Pairs across all reported metrics. We also observe that the number of UltraChat examples affects compressed-model performance. Among the evaluated calibration sizes, 256 UltraChat examples yield the lowest perplexity and the highest MMLU accuracy, whereas 64 examples yield the highest UnQover accuracy and the lowest DTO. The category-specific experiments also provide evidence of cross-category transfer on UnQover. Religion-specific calibration yields higher average UnQover accuracy than full StereoSet calibration, whereas gender- and race-specific calibration yield lower average accuracy.

F Safety Evaluation

To complement our evaluation of representational bias, we additionally assess whether Debias-SparseGPT affects broader model safety. First, we evaluate generations from compressed models conditioned on inputs from the **RealToxicityPrompts** benchmark (Gehman et al., 2020), following prior work studying the impact of model compression on toxicity and safety (Xu and Hu, 2022; Xu et al., 2024; Hong et al., 2024). RealToxicityPrompts contains human-written prompts designed to elicit potentially toxic model continuations. We randomly sample 1,200 prompts for evaluation and use the benchmark implementation provided by the LM Evaluation Harness framework.[13] To evaluate the generated continuations, we classify outputs using the LLaMA Guard 3-1B model (Inan et al., 2023).[14]

[13] github.com/EleutherAI/lm-evaluation-harness

[14] The Perspective API, used in the source implementation, no longer accepts new usage requests.

Model + Method	RealToxicityPrompts ↓ Unsafe Rate	HarmBench ↓ Unsafe Rate
LLaMA-3.1-8B + SparseGPT	0.033	0.265
LLaMA-3.1-8B + **Debias-SparseGPT**	**0.028**	**0.190**
Vicuna-1.5-7B + SparseGPT	0.344	0.510
Vicuna-1.5-7B + **Debias-SparseGPT**	**0.331**	**0.425**
Qwen-2.5-7B + SparseGPT	0.035	0.055
Qwen-2.5-7B + **Debias-SparseGPT**	**0.031**	**0.025**

Table 16: Toxicity and safety evaluation of models compressed with SparseGPT and Debias-SparseGPT under 1:4 semi-structured sparsity. We report the proportion of generated responses classified as unsafe by LLaMA Guard 3-1B; lower values are better. Debias-SparseGPT consistently yields lower unsafe-response rates across all evaluated model families and benchmarks.

Next, we evaluate model safety on **HarmBench** (Mazeika et al., 2024), a standardized benchmark containing prompts related to cybercrime, the production or use of chemical and biological weapons or drugs, misinformation, harassment, illegal activities, and other risks to human well-being.

For both benchmarks, we report the proportion of generated responses classified as unsafe by the Guard model; lower values are better. Generation is performed using deterministic greedy decoding with temperature set to zero. We compare the performance of the models compressed using both methods under the 1:4 semi-structured sparsity setting.

We report the results in Table 16. We find that Debias-SparseGPT achieves lower unsafe-response rates than SparseGPT-compressed models across all three model families on both RealToxicityPrompts and HarmBench. The largest absolute reduction is observed for Vicuna-1.5-7B on HarmBench, where the unsafe-response rate decreases from 0.510 to 0.425. For LLaMA-3.1-8B, the rate decreases from 0.265 to 0.190, while Qwen-2.5-7B exhibits the lowest HarmBench unsafe-response rate overall, decreasing from 0.055 to 0.025.

These results suggest that incorporating the proposed bias-aware objective during pruning does not introduce an adverse safety trade-off and, in the evaluated settings, is associated with improved performance relative to SparseGPT baselines.

G Analysis of Learned Sparsity Patterns

To examine how the bias-aware Hessian affects pruning decisions, we compare Qwen-2.5-7B-IT models compressed with SparseGPT and Debias-SparseGPT under 2:4 semi-structured sparsity. Both models are compressed using the same calibration data, consisting of full StereoSet and 256 examples from UltraChat. We analyze all seven pruned matrices in each layer: `q_proj`, `k_proj`, `v_proj`, and `o_proj` in the attention module, and `gate_proj`, `up_proj`, and `down_proj` in the MLP module.

For each final pruned matrix, we extract a binary mask in which a value of 1 denotes a pruned weight. We then compute four measures: 1) weight disagreement, the fraction of matrix positions where pruning decisions differ between the two masks, where larger values indicate greater mask divergence; 2) `2:4` pattern disagreement, the fraction of consecutive four-weight groups for which the two masks select different zero positions; 3) pruned-index Jaccard, the intersection-over-union of the two sets of pruned indices, where values closer to 1 indicate more similar masks and values closer to 0 indicate less overlap; and 4) relative Frobenius difference, computed as $\|\hat{\mathbf{W}}^{(1)} - \hat{\mathbf{W}}^{(0)}\|_F / \|\hat{\mathbf{W}}^{(0)}\|_F$, which measures the relative difference between the final pruned weight matrices. Together, these measures reflect the differences in the final pruning masks between models compressed with SparseGPT and Debias-SparseGPT.

Table 17 reports the results for matrices with the largest 2:4 pattern disagreement. We find that among the 20 matrices with the largest 2:4 pattern disagreement, 16 correspond to attention output projections, indicating that the effect of the bias-aware Hessian is non-uniform across layers and modules and is concentrated primarily in `self_attn.o_proj`.

Layer	Module	Weight Disagreement	2:4 Pattern Disagreement	Pruned-index Jaccard	Relative Frobenius Difference
26	self_attn.o_proj	**0.0975**	**0.1928**	**0.8223**	0.3040
27	self_attn.o_proj	0.0909	0.1797	0.8333	**0.3121**
22	self_attn.o_proj	0.0831	0.1644	0.8465	0.2731
19	self_attn.o_proj	0.0829	0.1643	0.8468	0.2636
13	self_attn.o_proj	0.0824	0.1633	0.8477	0.2602
9	self_attn.o_proj	0.0817	0.1618	0.8489	0.2684
18	self_attn.o_proj	0.0812	0.1606	0.8499	0.2670
8	self_attn.o_proj	0.0802	0.1586	0.8515	0.2617
12	self_attn.o_proj	0.0791	0.1569	0.8534	0.2609
7	self_attn.o_proj	0.0792	0.1564	0.8533	0.2680
0	mlp.up_proj	0.0781	0.1543	0.8552	0.2911
21	self_attn.o_proj	0.0774	0.1534	0.8563	0.2490
16	self_attn.o_proj	0.0771	0.1529	0.8568	0.2543
23	self_attn.o_proj	0.0749	0.1486	0.8607	0.2485
0	mlp.gate_proj	0.0750	0.1484	0.8604	0.2785
14	mlp.down_proj	0.0746	0.1482	0.8612	0.2525
25	self_attn.o_proj	0.0742	0.1473	0.8618	0.2481
20	self_attn.o_proj	0.0738	0.1463	0.8625	0.2513
13	mlp.down_proj	0.0735	0.1461	0.8631	0.2515
1	self_attn.o_proj	0.0738	0.1457	0.8626	0.2662

Table 17: Matrices with the largest 2:4 pruning-pattern disagreement between SparseGPT and Debias-SparseGPT for the Qwen-2.5-7B-IT model. Both models are compressed under 2:4 semi-structured sparsity using the same StereoSet and UltraChat calibration data. Layer indices are zero-based and range from 0 (first layer) to 32 (last layer). Larger disagreement values indicate stronger changes in pruning decisions induced by the bias-aware Hessian, whereas larger pruned-index Jaccard values indicate greater mask overlap. Bold values indicate the largest disagreement or relative-difference values and the smallest pruned-index Jaccard value.