DON'T BE GREEDY, JUST RELAX! PRUNING LLMS VIA FRANK-WOLFE

Anonymous authors
Paper under double-blind review

ABSTRACT

Pruning is a common technique to reduce the compute and storage requirements of Neural Networks. While conventional approaches typically retrain the model to recover pruning-induced performance degradation, state-of-the-art Large Language Model (LLM) pruning methods operate layer-wise, minimizing the per-layer pruning error on a small calibration dataset to avoid full retraining, which is considered computationally prohibitive for LLMs. However, finding the optimal pruning mask is a hard combinatorial problem and solving it to optimality is intractable. Existing methods hence rely on greedy heuristics that ignore the weight interactions in the pruning objective. In this work, we instead consider the convex relaxation of these combinatorial constraints and solve the resulting problem using the Frank-Wolfe (FW) algorithm. Our method drastically reduces the per-layer pruning error, outperforms strong baselines on state-of-the-art GPT architectures, and remains memory-efficient. We provide theoretical justification by showing that, combined with the convergence guarantees of the FW algorithm, we obtain an approximate solution to the original combinatorial problem upon rounding the relaxed solution to integrality.

1 INTRODUCTION

Pruning after training (Han et al., 2015; Gale et al., 2019; Hoefler et al., 2021; Zimmer et al., 2023; 2025) reduces the inference-time compute and memory footprint of Neural Networks with minimal impact on predictive performance. Conventional approaches obtain such *sparse* models by removing parameters using simple criteria such as their magnitude and then typically require full retraining to recover pruning-induced performance degradation. The drastic increase in model size accompanying the rise of LLMs has, however, reshaped the pruning landscape.

At LLM scale, full retraining is often considered prohibitively expensive or even infeasible, resulting in a surge of interest in pruning criteria that do not require retraining. In addition, classical magnitude pruning performs no better than random pruning for LLMs (Sun et al., 2023; Yin et al., 2023), an observation attributed to activation outliers (Dettmers et al., 2022) and highly important *super-weights* (Yu et al., 2025) in sufficiently large *Transformer* models (Vaswani et al., 2017). Consequently, state-of-the-art methods (Frantar & Alistarh, 2023; Sun et al., 2023; Zhang et al., 2024) prune *layerwise*: they decompose pruning into per-layer subproblems and treat layers sequentially and independently, estimating parameter importance on a small calibration set by minimizing a per-layer *local* pruning loss. Specifically, for a single layer with calibration input matrix $X \in \mathbb{R}^{d_{in} \times B}$ and weights $W \in \mathbb{R}^{d_{out} \times d_{in}}$, the objective is

$$\min_{M} \|WX - (M \odot W)X\|_F^2, \quad \text{s.t. } M \in \{0,1\}^{d_{out} \times d_{in}}, \|M\|_0 \leq k \qquad \text{(MASK SELECTION)}$$

where $M \in \{0,1\}^{d_{out} \times d_{in}}$ is a binary mask that enforces the target sparsity, e.g., $\|M\|_0 \leq k$ for unstructured pruning, and $\odot$ denotes the Hadamard product. Here, $B = N \cdot L$, where N is the number of samples in the calibration batch and L the sequence length.

However, even for a single layer, selecting the optimal pruning mask is a hard quadratic binary optimization problem. Solving (MASK SELECTION) to optimality is computationally intractable at LLM scale because the combinatorial constraint—choosing k out of $d_{out} \times d_{in}$ elements—results in

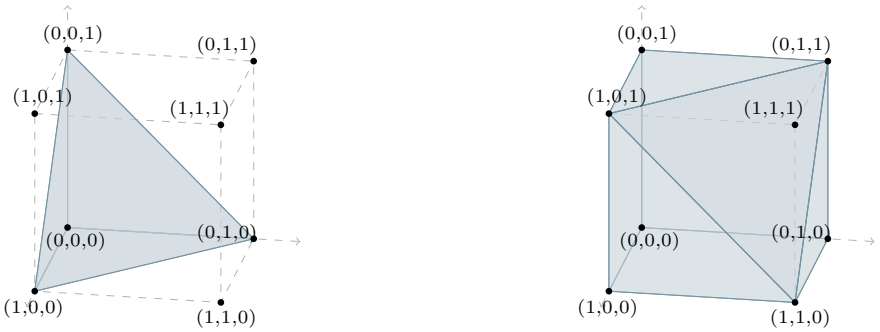

Figure 1: Visualization of $\mathcal{C}_k$ for $d_{\text{out}} = 3$, $d_{\text{in}} = 1$. **Left**: $k = 1$, **Right**: $k = 2$.

a search space that grows exponentially with the parameter count. Prior methods such as SparseGPT and Wanda therefore resort to greedy heuristics that ignore weight interactions to remain tractable[1].

In this work, we instead consider the convex relaxation of these combinatorial constraints: we approximate (MASK SELECTION) by optimizing over the convex hull of all masks, transforming the combinatorially hard problem into a tractable convex program

$$\min_{M} \|WX - (M \odot W)X\|_F^2, \quad \text{s.t. } M \in [0,1]^{d_{out} \times d_{in}}, \|M\|_1 \leq k \qquad \text{(RELAXED MASK SEL.)}$$

where M is now continuous with entries in $[0, 1]$, and the cardinality constraint is replaced by an L_1-norm budget, see Figure 1 for a visualization. The resulting convex program can be solved efficiently using the first-order Frank-Wolfe (FW) algorithm (Lacoste-Julien et al., 2013; Zeng & Figueiredo, 2014; Carderera et al., 2021; Braun et al., 2022). Notably, FW is projection-free and moves toward extreme points of the feasible set (i.e., binary masks) via a Linear Minimization Oracle (LMO), which is efficient to compute and naturally yields sparse updates.

Our method, which we term SparseFW, reduces the per-layer pruning error by up to 80% compared to state-of-the-art methods such as Wanda (Sun et al., 2023), and outperforms them on benchmark GPT architectures such as Qwen 2.5, LLaMA 3, Yi 1.5, and Gemma 2, with consistent gains in final WikiText perplexity and zero-shot accuracy. SparseFW is efficient, requires little memory overhead, easily adapts to unstructured and semi-structured sparsity patterns, is simple to implement, and scales to large models. Furthermore, unlike competing methods, SparseFW comes with strong theoretical justification: we show that, combined with the convergence guarantees of FW, rounding the relaxed solution to integrality yields an approximate solution to the original combinatorial problem.

Contributions. We summarize our contributions as follows.

1. **SparseFW: A projection-free method for layerwise pruning.** We formulate the layerwise mask selection problem as a convex program over the convex hull of binary masks and propose to solve it with the Frank-Wolfe (FW) algorithm, which is projection-free and leverages an efficient LMO that naturally yields sparse updates. SparseFW is memory-efficient, simple to implement, scales to large models, and can be used to induce both unstructured and semi-structured sparsity patterns.
2. **Strong empirical performance at LLM scale.** SparseFW reduces the per-layer pruning error by up to 70% compared to state-of-the-art methods such as Wanda, and delivers consistent gains in final WikiText perplexity and zero-shot accuracy across modern GPT architectures (e.g., Qwen 2.5, LLaMA 3, Yi 1.5, Gemma 2).
3. **Theoretical guarantees.** We provide approximation guarantees that connect the relaxed solution returned by FW after rounding to integrality to an approximate solution of the original combinatorial mask selection problem.

Our work demonstrates that classical constrained optimization techniques are not only feasible for pruning LLMs but can drastically improve upon state-of-the-art performance.

[1]We discuss these methods in detail in Section 2.

Related work. *Pruning after training* (Hoefler et al., 2021) is among the most popular approaches to reduce the resource demands of neural networks during inference. *Magnitude pruning* (Janowsky, 1989; Han et al., 2015) is the de facto default pruning criterion for convolutional architectures, and has been shown to yield pruned models that perform competitively, despite its simplicity (Gale et al., 2019; Zimmer et al., 2023). Various other criteria exist to decide which weights to consider unimportant (cf. LeCun et al., 1989; Hassibi & Stork, 1993; Molchanov et al., 2016; Yeom et al., 2019). With the rise of LLMs, magnitude pruning is being replaced by criteria that account for the peculiarities of LLMs (in particular, large activation outliers, cf. e.g. Dettmers et al., 2022; Yin et al., 2023) and that aim to avoid requiring retraining (Kwon et al., 2022; Frantar & Alistarh, 2023; Sun et al., 2023), which is generally considered computationally prohibitive for large models. Most importantly for our work, SparseGPT (Frantar & Alistarh, 2023), Wanda (Sun et al., 2023), and RIA (Zhang et al., 2024) address the mask selection problem (MASK SELECTION) using a greedy pruning approach, where the selection of weights to prune is performed iteratively. Our approach, on the other hand, relaxes the combinatorial constraint and takes weight interactions into account.

Frank-Wolfe (FW) or *conditional gradient* algorithms (Frank et al., 1956; Levitin & Polyak, 1966) are widely used in Machine Learning for handling complex structural requirements efficiently (Lacoste-Julien et al., 2013; Zeng & Figueiredo, 2014; Frandi et al., 2015; Jaggi, 2013; Négiar et al., 2020), with numerous theoretical works (Lacoste-Julien, 2016; Hazan & Luo, 2016; Reddi et al., 2016) and accelerated variants (Hazan & Luo, 2016; Yurtsever et al., 2019; Shen et al., 2019; Combettes et al., 2020; Mokhtari et al., 2018; Chen et al., 2018) appearing in the literature. For a comprehensive review, see Braun et al. (2022). Recently, FW has been applied in the context of neural networks (Ravi et al., 2018; Xie et al., 2019; Berrada et al., 2018; Tsiligkaridis & Roberts, 2020), for training neural networks at scale (Pokutta et al., 2020; Pethick et al., 2025), and Miao et al. (2022) as well as Zimmer et al. (2025) use FW-variants for inducing sparsity throughout pretraining.

2 METHODOLOGY

We begin by discussing the preliminaries and demonstrating that three state-of-the-art LLM pruning methods, namely SparseGPT, Wanda, and RIA, address the mask selection problem (MASK SELECTION) using a greedy pruning approach. We then introduce the FW algorithm and our proposed method, SparseFW. Throughout this section, we use lowercase letters for scalars and vectors and uppercase letters for matrices (W, X, M). Matrix entries are denoted W_{ij} for the element in row i, column j. Rows of matrices are denoted with lowercase subscripts: w_i represents the i-th row of matrix W. We use slicing notation, e.g., $X_{j,:}$ denotes the j-th row of matrix X.

2.1 PRELIMINARIES AND GREEDY METHODS

Before discussing SparseGPT, Wanda, and RIA in detail, we first note that the objective in Equation (MASK SELECTION) decomposes into a sum of d_{out} row-wise quadratic functions

$$\|WX - (M \odot W)X\|_F^2 = \sum_{i=1}^{d_{\text{out}}} \| \left(w_i - m_i \odot w_i\right) X\|_2^2, \tag{1}$$

with $w_i \in \mathbb{R}^{d_{in}}$ and $m_i \in \{0,1\}^{d_{in}}$ denoting the i-th row of W and M, respectively. Under unstructured sparsity, the constraint in (MASK SELECTION) couples the rows, making the problem non-separable. In contrast, semi-structured patterns such as $n{:}m$ (prune $M-N$ per block of M weights) enforce equal per-row sparsity levels and hence fully decouple the rows. For simplicity, we will mainly discuss the row-wise formulation of Equation (1) and drop the index i. We now analyze how SparseGPT, Wanda, and RIA tackle the mask selection problem (MASK SELECTION) through greedy pruning—removing one weight at a time. These methods are optimal for their single-weight pruning objective, effectively bypassing weight interactions to simplify the problem.

SparseGPT (Frantar & Alistarh, 2023) is arguably the most popular approach and is largely based on preceding work (Frantar et al., 2022) of the authors. In practice, it prunes small blocks of weights at a time to ensure scalability to large models, instead of single weights in isolation as suggested by the theory; we briefly describe the underlying approach based on single-weight pruning. Instead of focusing solely on mask selection, SparseGPT approximates the problem of finding a sparse replacement $\hat{w}$ for the weight vector w, thus combining the problems of mask selection and

reconstruction of remaining weights by solving

$$\min_{\hat{w}} \|w^\top X - \hat{w}^\top X\|_F^2, \quad \text{s.t. } \|\hat{w}\|_0 \leq k. \tag{2}$$

Since solving this problem exactly is intractable, SparseGPT follows a greedy procedure to approximately solve it: at each step it finds the optimal *single* weight to prune and the corresponding optimal remaining weights, i.e., it solves

$$\min_{\hat{w}, q \in [d_{\text{in}}] \text{ s.t. } e_q^\top \hat{w} = 0} \|(\hat{w} - w)^\top X\|_2^2. \tag{3}$$

The greedy-best weight index q and the optimal weight reconstruction are then given by

$$w^* = w - \frac{w_q}{[(XX^\top)^{-1}]_{qq}} (XX^\top)^{-1} e_q, \text{ where } q \in \arg\min_{q \in [d_{\text{in}}]} \frac{w_q^2}{((XX^\top)^{-1})_{qq}}.$$

Wanda (Sun et al., 2023) computes a saliency score $S_{i,j} := |W_{i,j}| \, \|X_{j,:}\|_2$ for each weight and then prunes the weights with the smallest saliencies. The authors motivate their approach by the observation that in LLMs, some weights with small magnitudes correspond to large-magnitude features (cf. e.g. Dettmers et al., 2022) and that their removal can lead to significant performance drops, despite their small magnitude. Wanda hence multiplies magnitude saliencies by the corresponding input activation norm to avoid pruning such small-but-important weights.

We argue that Wanda can be seen as a greedy approximation to (MASK SELECTION) and focus on a single row w for simplicity. Again, we write the optimization problem for pruning one variable, but now without modifying the remaining weights:

$$\min_{\hat{w} = (1 - e_q) \odot w, \, q \in [d_{\text{in}}]} \left\{ \|(\hat{w} - w)^\top X\|_2^2 \right\} \tag{4}$$

Plugging the constraints into the objective function directly yields

$$\min_{q \in [d_{\text{in}}]} \left\{ \| \left((1 - e_q) \odot w \right) - w)^\top X \|_2^2 \right\} = \min_{q \in [d_{\text{in}}]} \left\{ w_q^2 (XX^\top)_{qq} \right\} \tag{5}$$

Now note that $w_q^2 (XX^\top)_{qq} = w_q^2 \|X_{q,:}\|_2^2$. Minimizing the latter over q is equivalent to minimizing $|w_q| \, \|X_{q,:}\|_2$, which is exactly the saliency score of Wanda.

While it might seem that this procedure differs from Wanda, as Wanda computes saliency scores once for all weights and not iteratively, the approaches are identical since the saliency scores do not change after pruning a weight. Wanda further enforces row-wise sparsity rather than unstructured sparsity, pruning a fixed number of weights per row. This has been found beneficial for LLMs (Sun et al., 2023); the same does not hold for other transformer-like models.

RIA (Zhang et al., 2024) builds upon Wanda and uses the following saliency score:

$$S_{ij}^{\text{RIA}} := |W_{ij}| \left(\frac{1}{\sum_{k=1}^{d_{\text{in}}} |W_{ik}|} + \frac{1}{\sum_{k=1}^{d_{\text{out}}} |W_{kj}|} \right) \|X_{j,:}\|_2. \tag{6}$$

We employ full-matrix notation since RIA fundamentally depends on the matrix structure for its row- and column-wise renormalization. Letting W' denote the rescaled weight matrix with entries

$$W'_{ij} := W_{ij} \left(\frac{1}{\sum_{k=1}^{d_{\text{in}}} |W_{ik}|} + \frac{1}{\sum_{k=1}^{d_{\text{out}}} |W_{kj}|} \right).$$

Applying Wanda on W' to prune the weights with the smallest saliency scores yields

$$|W'_{ij}| \, \|X_{j,:}\|_2 =: S_{ij}^{\text{RIA}}, \tag{7}$$

which is exactly the saliency score of RIA. The RIA criterion can be interpreted as using the same greedy pruning algorithm as Wanda, but applied to a rescaled weight matrix.

2.2 SOLVING THE CONVEX RELAXATION WITH FRANK-WOLFE

We present an alternative approach to the greedy approximations discussed in the previous section, which is based on relaxing the combinatorial constraints to obtain a convex optimization problem, instead of trying to make the problem tractable by making the pruning decision on a per-weight basis. We solve the convex problem using the FW algorithm, which we introduce in the following.

The Frank-Wolfe Algorithm. When minimizing some objective function $\mathcal{L}$ over a set of constraints $\mathcal{C}$, a classical approach is Projected Gradient Descent (PGD) which iteratively performs a gradient step and then projects the result back to the constraint set to ensure feasibility of the iterates. However, depending on $\mathcal{C}$, this projection step may not admit an analytic solution and can be computationally expensive (Jaggi, 2013; Combettes & Pokutta, 2021). The FW algorithm is an alternative which is projection-free and often yields solutions with desirable structure. Instead of moving along the gradient direction and then requiring a projection step, FW moves towards the boundary point of the feasible region that is best aligned with the descent direction. Specifically, in each iteration t and at iterate M_t, FW calls a Linear Minimization Oracle (LMO) on the gradient $\nabla\mathcal{L}(M_t)$ of $\mathcal{L}$ at M_t to solve

$$V_t = \arg\min_{V \in \mathcal{C}} \langle V, \nabla\mathcal{L}(M_t)\rangle, \tag{8}$$

which is then used to update the parameters using the convex combination

$$M_{t+1} \leftarrow (1-\eta_t)M_t + \eta_t V_t, \tag{9}$$

where $\eta_t \in [0,1]$ is the step size. Throughout this work, we stick to the learning rate schedule given by $\eta_t = \frac{2}{t+2}$. If now $M_0 \in \mathcal{C}$, then the convex update rule ensures feasibility of all iterates. In practice, solving Equation (8) is often much cheaper than performing a projection step. If $\mathcal{C}$ is further given by the convex hull of a set of points, e.g., the vertices of a polytope, then the solution to Equation (8) is attained at one of these points. In each iteration, FW moves towards the vertices.

Relaxing the combinatorial constraints. The FW algorithm can only be applied to convex constraint sets, which is not the case for (MASK SELECTION). We make the problem tractable by relaxing the combinatorial constraints to their convex hull, i.e.,

$$\mathcal{C}_k = \left\{M \in [0,1]^{d_{\text{out}} \times d_{\text{in}}} : \|M\|_1 \leq k\right\}. \tag{10}$$

Given that the objective function of (MASK SELECTION) is a convex quadratic, this relaxation transforms the combinatorial mask selection problem into a convex optimization problem, which can be solved efficiently using the FW algorithm. We restate the reformulation of (RELAXED MASK SEL.) for completeness:

$$\min_{M \in \mathcal{C}_k} \|WX - (M \odot W)X\|_F^2. \tag{11}$$

This relaxation has the advantage that, unlike the previously discussed greedy approaches, it fully accounts for interactions between weights. However, the solution to the relaxed problem (RELAXED MASK SEL.) is not guaranteed to be feasible for the original problem (MASK SELECTION); in Section 4, we show that rounding the relaxed solution to integrality yields an approximate solution to the original problem.

The sparse Linear Minimization Oracle. We next discuss how to compute the LMO for the feasible set $\mathcal{C}_k$. Note that $\mathcal{C}_k$ is a polytope and can be described as the convex hull of its vertices, which are exactly the binary masks with at most k ones. At any vertex, all coordinates lie on box bounds 0 or 1, and the coupling constraint $\sum_{i,j} M_{ij} \leq k$ is either inactive (fewer than k ones) or tight (exactly k ones); see Figure 1. Minimizing a linear function over $\mathcal{C}_k$ therefore consists of selecting up to k entries with the most negative coefficients and setting them to one, leaving the rest at zero. Letting $\nabla\mathcal{L}(M_t) \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ denote the gradient of the objective at iterate M_t, the LMO solution at step t is hence given by

$$[\text{LMO}\left(\nabla\mathcal{L}(M_t)\right)]_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \texttt{Top-k}\left(-\nabla\mathcal{L}(M_t)\right), [\nabla\mathcal{L}(M_t)]_{ij} < 0 \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

where $\texttt{Top-k}(\nabla\mathcal{L}(M_t))$ denotes the set of indices corresponding to the k entries of $\nabla\mathcal{L}(M_t)$ with the smallest values. The LMO for $\mathcal{C}_k$ can be computed efficiently and naturally produces sparse updates: at most k out of $d_{\text{out}} \cdot d_{\text{in}}$ entries are nonzero. While the above corresponds to unstructured sparsity, the LMO can be adapted to per-row sparsity and n:m sparsity; see Appendix D.

2.3 THE SPARSEFW ALGORITHM

We present the full SparseFW algorithm in Algorithm 1. At a high level, for each layer we solve the relaxed optimization problem using the FW algorithm, starting from any binary mask that satisfies the sparsity constraints. After running for T iterations, we threshold the learned mask—whose entries lie in $[0, 1]$—to obtain a binary mask that meets the original sparsity constraints. The objective function and the gradient with respect to M_t are given by

$$\mathcal{L}(M_t) = \mathrm{Tr}(W(1 - M_t)XX^\top(1 - M_t)^\top W^\top)$$
$$\nabla\mathcal{L}(M_t) = -2 \cdot W \odot (WXX^\top - (W \odot M_t)XX^\top).$$

Even for small calibration datasets, the activation matrix X can be very large. For example, the largest matrix in a LLaMA-2-7B transformer block (`up_proj`) has $d_{in} = 4096$. With $N = 128$ samples and sequence length $L = 4096$, X has dimensions $4096 \times 524{,}288$. Because both the objective and the gradient depend only on $G := XX^\top$ (which can be computed in batches), we precompute $G := XX^\top$ and $H := WG$ once to drastically reduce resource demands. Note that G has dimensions 4096×4096, in contrast to the $4096 \times 524{,}288$ dimensions of X; this independence of the sequence length L and number of samples N is crucial for efficiency. With G and H precomputed, the gradient requires only two elementwise multiplications, a matrix–matrix multiplication, and a matrix addition:

$$\nabla\mathcal{L}(M_t) = -2 \cdot W \odot (H - (W \odot M_t)G).$$

In practice, we have to navigate a caveat that we did not detail in Algorithm 1 for the sake of simplicity, exact details are in the appendix. Throughout the experiments, we noticed that while FW often substantially reduces pruning error relative to baselines like Wanda, it can still produce worse final perplexity, likely due to a mismatch between local and global objectives. Constraining Sparse Frank-Wolfe (SparseFW) by fixing a fraction of very high-saliency weights (e.g., those with highest Wanda scores) as unprunable consistently improves performance. This suggests that Wanda reliably identifies weights that should be preserved, even if a more thorough local optimization would prune them. We therefore fix these weights and apply FW to the remaining ones, optimizing over a smaller search space. We ablate the impact of this ratio in Table 2 in the appendix: Surprisingly, we observe the best consistent improvements when setting $\alpha = 0.9$, i.e., fixing 90% of the highest saliency weights and optimizing only over the remaining 10%. Even small α values (e.g., $\alpha = 0.1$) can yield significant perplexity improvements. On the other hand, setting $\alpha = 0.0$ (full FW without any fixed weights) consistently yields worse results than the baselines.

Algorithm 1 SparseFW

Require: Weight matrix W, input X, no. of nonzero entries k, iterations T, warm-start mask M_0

1: $G = XX^\top$, $H = WG$ ▷ Precompute buffers
2: **for** $t = 0$ to $T - 1$ **do**
3: $\quad\nabla\mathcal{L}(M_t) = -2 \cdot W \odot (H - (W \odot M_t)G)$ ▷ Compute gradient
4: $\quad V_t = \mathrm{LMO}\big(\nabla\mathcal{L}(M_t), \mathcal{C}_k\big)$ ▷ Compute LMO
5: $\quad\eta_t \leftarrow \frac{2}{t+2}$
6: $\quad M_{t+1} \leftarrow (1 - \eta_t)M_t + \eta_t V_t$ ▷ FW Update
7: $[M]_{ij} \leftarrow \begin{cases} 1 & \text{if } (i,j) \in \texttt{Top-k}(M_T) \\ 0 & \text{otherwise} \end{cases}$ ▷ Threshold
8: **return** M

3 EXPERIMENTAL RESULTS

We present our experimental methodology; our code will be made publicly available to ensure reproducibility. Our focus is on language modeling and we utilize pretrained models from HuggingFace (Wolf et al., 2020), including *LLaMA-3.1-8B* (Grattafiori et al., 2024), *Gemma-2-9B* (Riviere et al., 2024), *Yi-1.5-9B* (Young et al., 2025), *DeepSeek-7B-base* (Bi et al., 2024), and *Qwen2.5-7B* (Yang et al., 2025). For the calibration set, we randomly sample 2048-token sequences from the *C4* dataset (Raffel et al., 2020). For validation, we select 100 sequences from the validation split. We evaluate performance using perplexity on *WikiText* (Merity et al., 2016) and zero-shot accuracy on

Figure 2: LLaMA-3.1-8B pruned to 60% unstructured sparsity with SparseFW using Wanda warmstart with 256 samples. This figure shows the relative reduction in pruning error (y-axis) for each matrix type (see legend for colors) for all layers of the model (x-axis) compared to the warmstart mask.

the EleutherAI evaluation set (Gao et al., 2023). Following Sun et al. (2023), we prune all linear layers with a uniform sparsity allocation across layers, while keeping the embedding and final linear head dense. SparseFW is compared with Wanda and RIA, as these methods also aim to find a better pruning mask by solving (MASK SELECTION); we hence do not compare directly to methods that involve a reconstruction step, such as SparseGPT (Frantar & Alistarh, 2023). We report results for both unstructured and semi-structured sparsity (Mishra et al., 2021).

SparseFW outperforms state-of-the-art mask selection methods. In Table 1, we compare SparseFW (warm-started from Wanda or RIA) to the respective baselines across five state-of-the-art GPTs and multiple sparsity regimes (50%, 60%, and 2:4). SparseFW generally performs on par with or better than the baselines in terms of perplexity; for zero-shot accuracy, SparseFW consistently outperforms competing methods. We generally observe much more consistent and bigger improvements in the higher sparsity regimes than for 50% sparsity.

SparseFW successfully optimizes the matrix-wise pruning objective. We observe consistent improvement in terms of the local pruning objective over both Wanda and RIA warmstarts. Figure 2 shows the per-layer reductions relative to a Wanda Warmstart, where we observe reductions of up to 80%. In general, we found the average relative reduction over the layers to range between 20% and 40% across the different models, sparsity regimes and warmstarts.

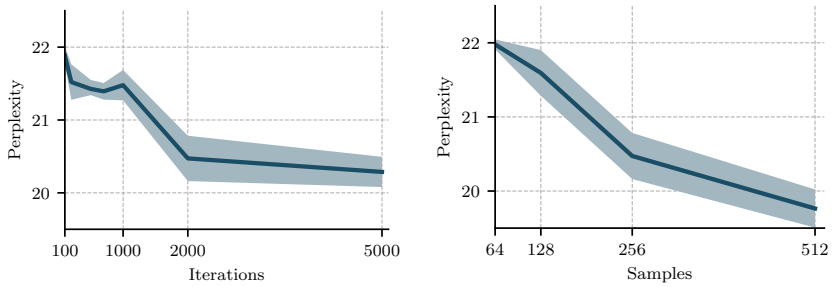

Figure 3: LLaMA-3.1-8B pruned to 2:4 sparsity using SparseFW. Left: Perplexity over the number of SparseFW iterations per layer with 256 samples. Right: Perplexity over the number of calibration samples with 2000 SparseFW iterations per layer. The solid curve represents the mean over multiple random seeds, the shaded regions represent the min-max range.

Sample and iteration efficiency. Figure 3 ablates the impact of the number of SparseFW iterations (left) and the number of calibration samples (right). Fixing the amount of samples at 256, perplexity decreases up to around 2000 iterations and then flattens. We therefore use 2000 iterations throughout. In contrast, at a fixed 2000 iterations, increasing the number of calibration samples from 64 to 512 brings substantial additional perplexity gains. This trend contrasts with Wanda, whose performance

Table 1: Perplexity (↓, lower is better) and zero-shot accuracy (↑, higher is better) comparison. We report SparseFW performance with Wanda and RIA warmstart for unstructured 50% and 60% sparsity and semi-structured 2:4 sparsity after 2000 iterations using 256 samples compared to the baseline warmstarts. We indicate the SparseFW warmstart method in parentheses. Best values are highlighted in bold. We omit standard deviations for legibility.

Perplexity (↓)		GEMMA-2	YI-1.5	DEEPSEEK-7	QWEN2.5		LLAMA-3
Method	Sparsity	9B	9B	7B	7B	14B	8B
Wanda	50%	11.19	6.58	**7.79**	8.45	7.11	10.09
RIA		11.19	6.71	7.90	8.54	7.01	**9.88**
SparseFW (Wanda)		**10.67**	6.58	7.89	8.35	7.10	10.21
SparseFW (RIA)		10.77	**6.53**	7.93	**8.22**	**6.98**	9.95
Wanda	60%	16.46	11.38	**11.44**	13.47	10.87	21.53
RIA		17.17	14.37	11.87	12.86	9.78	19.14
SparseFW (Wanda)		**14.83**	**10.56**	11.99	12.44	10.28	**17.97**
SparseFW (RIA)		15.07	10.67	12.41	**11.66**	**9.65**	18.16
Wanda	2:4	17.41	11.58	11.76	14.40	11.37	24.82
RIA		16.78	11.27	12.04	13.46	**10.98**	23.7
SparseFW (Wanda)		**15.81**	10.61	**11.73**	14.16	11.82	**20.45**
SparseFW (RIA)		15.83	**10.35**	11.91	**13.42**	11.20	21.31
Accuracy in % (↑)		GEMMA-2	YI-1.5	DEEPSEEK-7	QWEN2.5		LLAMA-3
Method	Sparsity	9B	9B	7B	7B	14B	8B
Wanda	50%	68.44	61.04	56.67	63.72	67.94	58.78
RIA		**68.71**	61.22	55.76	64.03	67.83	58.94
SparseFW (Wanda)		68.42	62.49	**56.8**	64.97	**69.44**	**60.17**
SparseFW (RIA)		68.67	**62.53**	56.24	**65.34**	69.19	59.63
Wanda	60%	63.19	53.7	50.51	59.44	63.58	48.08
RIA		63.19	53.7	50.51	59.44	63.58	48.08
SparseFW (Wanda)		64.46	**54.90**	50.56	61.13	65.59	**51.92**
SparseFW (RIA)		**65.35**	55.41	**50.65**	**61.52**	**65.80**	52.15
Wanda	2:4	63.75	52.92	50.65	59.11	63.39	47.13
RIA		63.83	52.41	51.08	58.48	63.85	47.77
SparseFW (Wanda)		63.81	**53.78**	**51.12**	**60.15**	64.12	48.43
SparseFW (RIA)		**63.90**	52.54	50.69	**60.15**	**64.35**	**48.54**

does not seem to increase significantly with additional calibration data: increasing the sample count from 64 to 512 leads to a perplexity decrease from 25.1 to only 24.6 for Wanda. Overall, SparseFW is clearly more compute-intensive than Wanda and RIA, but we argue that spending more resources once to improve the performance of pruned models is, given that deployed LLMs now serve millions of users and inference costs scale with the number of requests, worthwhile. That being said, the results of Figure 3 indicate clear benefits of increasing the number of samples while keeping the number of iterations fixed and relatively low. While more samples require slightly more compute to build the matrix $G = XX^\top$, the cost of a single FW iteration is independent of the sample count.

4 THEORETICAL RESULTS

In this section, we state a data-dependent error guarantee for the mask produced by SparseFW with respect to the original pruning objective (MASK SELECTION). This is a key benefit of SparseFW over greedy heuristics, which can yield suboptimal solutions even though the objective function is convex. We state our main result informally here, deferring full statements and proofs to the appendix.

Lemma 1 (Informal)**.** *After T iterations of SparseFW, the resulting mask M satisfies*

$$\mathcal{L}(M) - \mathcal{L}(M^*) \leq \lambda_{\max}(Q)\left(\frac{k}{T} + 2\left(k + \sqrt{2d_{in}d_{out}k}\right)\right)$$

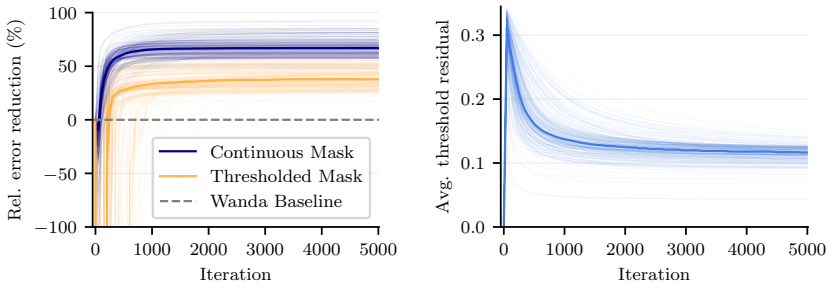

Figure 4: LLaMA-3.1-8B optimized towards 60% unstructured sparsity with SparseFW using 256 calibration samples. Lightly colored curves show the results individual matrices; the solid curve is their median. Left: Relative pruning error reduction versus FW iterations for continuous and thresholded masks. Right: Average threshold residual (mean ℓ_1 distance between continuous and thresholded masks) versus iterations.

where M^* is an optimal mask for (MASK SELECTION), k is the maximum number of nonzeros in the mask, Q represents the Hessian of the objective function and $\lambda_{\max}(Q)$ its largest eigenvalue.

Note that Q is not equal to $G = XX^\top$, the latter being the Hessian of the objective w.r.t. reconstruction of the weights, not w.r.t. the mask. The bound captures two sources of error: (i) the *optimization error* from solving the relaxed problem (RELAXED MASK SEL.), and (ii) the *thresholding error* from converting a relaxed solution to a binary mask (Line 7 in Algorithm 1).

Optimization error. After T iterations of the FW algorithm, the resulting (continuous, not-yet-thresholded) mask M_T satisfies

$$\mathcal{L}(M_T) - \mathcal{L}(\hat{M}) \leq k\lambda_{\max}(Q)/T,$$

where $\hat{M}$ is an optimal solution to the relaxed problem (RELAXED MASK SEL.). In other words, by increasing the number of iterations T, FW can guarantee an arbitrarily small optimization error.

Thresholding error. The error due to thresholding can be controlled by the curvature of the objective (captured by $\lambda_{\max}(Q)$) and the distance between the fractional iterate and its thresholded version, which in turn can be upper bounded in terms of k and the dimension of the input space $d_{\text{in}}d_{\text{out}}$.

These insights explain the empirical behavior in Figure 4. The left panel reports the relative pruning error reduction (higher is better) versus FW iterations for the continuous and thresholded masks. After a short initial drop, due to the large stepsize, the continuous iterate improves consistently, as predicted by the FW convergence guarantee. In contrast, the thresholded mask first degrades as the thresholding error grows while the iterate moves through the interior of $\mathcal{C}_k$. This is reflected in the right panel, which shows the average threshold residual (the $\|\cdot\|_1$ distance between the continuous and thresholded masks): It first rises steeply, then decreases and eventually plateaus above zero. As long as the relaxed solution is not at a vertex, the thresholding error remains nonzero, so the thresholded curve does not fully catch up to the continuous one.

5 CONCLUSION

Solving the pruning mask selection problem for LLMs is a hard combinatorial problem. In this work, we relax the binary constraints to their convex hull and solve the resulting convex problem with the FW algorithm; we call this approach SparseFW, a simple and memory-efficient layerwise method that explicitly accounts for weight interactions and supports both unstructured and semi-structured sparsity. Across modern GPT architectures, SparseFW drastically reduces the per-layer reconstruction error and improves perplexity and zero-shot accuracy over state-of-the-art LLM pruning approaches. Our work demonstrates that classical constrained optimization is a scalable and effective alternative to greedy heuristics for LLM pruning.

However, our work is not without limitations. Although vanilla FW substantially reduces per-layer pruning error, this does not reliably yield lower perplexity. Without fixing part of the mask, it tends

to prune weights crucial for overall performance. SparseFW successfully mitigates this by preserving a fraction of high-saliency weights from the warmstart, but the local–global objective mismatch persists; inductive biases still appear necessary for improved perplexity.

REFERENCES

Leonard Berrada, Andrew Zisserman, and M. Pawan Kumar. Deep frank-wolfe for neural network optimization. *International Conference on Learning Representations 2019*, November 2018.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism, January 2024. URL http://arxiv.org/abs/2401.02954.

Gábor Braun, Alejandro Carderera, Cyrille W Combettes, Hamed Hassani, Amin Karbasi, Aryan Mokhtari, and Sebastian Pokutta. Conditional gradient methods. November 2022. URL https://conditional-gradients.org/.

Alejandro Carderera, Sebastian Pokutta, Christof Schütte, and Martin Weiser. Cindy: Conditional gradient-based identification of non-linear dynamics – noise-robust recovery. January 2021.

Lin Chen, Christopher Harshaw, Hamed Hassani, and Amin Karbasi. Projection-free online optimization with stochastic gradient: From convexity to submodularity. In *International Conference on Machine Learning*, pp. 814–823. PMLR, 2018.

Cyrille W. Combettes and Sebastian Pokutta. Complexity of linear minimization and projection on some sets. January 2021.

Cyrille W. Combettes, Christoph Spiegel, and Sebastian Pokutta. Projection-free adaptive gradients for large-scale optimization. September 2020.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. August 2022.

Emanuele Frandi, Ricardo Nanculef, Stefano Lodi, Claudio Sartori, and Johan A. K. Suykens. Fast and scalable lasso via stochastic frank-wolfe methods with a convergence guarantee. October 2015.

Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. August 2022.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai

Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 Herd of Models, November 2024. URL http://arxiv.org/abs/2407.21783.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. Hanson, J. Cowan, and C. Giles (eds.), *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1993. URL https://proceedings.neurips.cc/paper/1992/file/303ed4c69846ab36c2904d3ba8573050-Paper.pdf.

Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pp. 1263–1271. PMLR, 2016.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint arXiv:2102.00554*, January 2021.

Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th international conference on machine learning*, pp. 427–435, 2013.

Steven A. Janowsky. Pruning versus clipping in neural networks. *Phys. Rev. A*, 39:6600–6603, Jun 1989. doi: 10.1103/PhysRevA.39.6600.

Woosuk Kwon, Sehoon Kim, Michael W. Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. March 2022.

Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. July 2016.

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *International Conference on Machine Learning*, pp. 53–61. PMLR, 2013.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In David S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pp. 598–605. Morgan Kaufmann, 1989. URL http://papers.nips.cc/paper/250-optimal-brain-damage.

Evgeny S Levitin and Boris T Polyak. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6(5):1–50, 1966.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. September 2016.

Lu Miao, Xiaolong Luo, Tianlong Chen, Wuyang Chen, Dong Liu, and Zhangyang Wang. Learning pruning-friendly networks via frank-wolfe: One-shot, any-sparsity, and no retraining. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=O1DEtITim__.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. April 2021.

Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Conditional gradient method for stochastic submodular maximization: Closing the gap. In *International Conference on Artificial Intelligence and Statistics*, pp. 1886–1895. PMLR, 2018.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. November 2016.

Geoffrey Négiar, Gideon Dresdner, Alicia Tsai, Laurent El Ghaoui, Francesco Locatello, Robert Freund, and Fabian Pedregosa. Stochastic frank-wolfe for constrained finite-sum minimization. In *International Conference on Machine Learning*, pp. 7253–7262. PMLR, 2020.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. In *Forty-Second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2Oqm2IzTy9.

Sebastian Pokutta, Christoph Spiegel, and Max Zimmer. Deep neural network training with frank-wolfe. *arXiv preprint arXiv:2010.07243*, 2020. URL https://arxiv.org/abs/2010.07243.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Sathya N. Ravi, Tuan Dinh, Vishnu Lokhande, and Vikas Singh. Constrained deep learning using conditional gradient and applications in computer vision. March 2018.

Sashank J Reddi, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic frank-wolfe methods for nonconvex optimization. In *2016 54th annual Allerton conference on communication, control, and computing (Allerton)*, pp. 1244–1251. IEEE, 2016.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian,

Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving Open Language Models at a Practical Size, October 2024. URL http://arxiv.org/abs/2408.00118.

Zebang Shen, Cong Fang, Peilin Zhao, Junzhou Huang, and Hui Qian. Complexities in projection-free stochastic non-convex minimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2868–2876. PMLR, 2019.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. June 2023.

Theodoros Tsiligkaridis and Jay Roberts. On frank-wolfe optimization for adversarial robustness and interpretability. December 2020.

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

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, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Jiahao Xie, Zebang Shen, Chao Zhang, Boyu Wang, and Hui Qian. Efficient projection-free online methods with stochastic recursive gradient. October 2019.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report, January 2025. URL http://arxiv.org/abs/2412.15115.

Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Pruning by explaining: A novel criterion for deep neural network pruning. December 2019.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Mykola Pechenizkiy, Yi Liang, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. October 2023.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yanpeng Li, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. Yi: Open Foundation Models by 01.AI, January 2025. URL http://arxiv.org/abs/2403.04652.

Mengxia Yu, De Wang, Qi Shan, Colorado J. Reed, and Alvin Wan. The Super Weight in Large Language Models, July 2025. URL http://arxiv.org/abs/2411.07191.

Alp Yurtsever, Suvrit Sra, and Volkan Cevher. Conditional gradient methods via stochastic path-integrated differential estimator. In *International Conference on Machine Learning*, pp. 7282–7291. PMLR, 2019.

Xiangrong Zeng and Mário A. T. Figueiredo. The ordered weighted ℓ_1 norm: Atomic formulation, projections, and algorithms. September 2014.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. Plug-and-play: An efficient post-training pruning method for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Tr0lPx9woF.

Max Zimmer, Christoph Spiegel, and Sebastian Pokutta. How I Learned To Stop Worrying And Love Retraining. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=_nF5imFKQI.

Max Zimmer, Christoph Spiegel, and Sebastian Pokutta. *Compression-aware training of neural networks using Frank–Wolfe*, pp. 137–168. De Gruyter, Berlin, Boston, 2025. ISBN 9783111376776. doi: doi:10.1515/9783111376776-010. URL https://doi.org/10.1515/9783111376776-010.

A USE OF LARGE LANGUAGE MODELS

Large language models were used to aid in writing (polishing text) as well as to help with the implementation of code components, including both the methods and the generation of plots. They also served as a tool for brainstorming research ideas and refining development approaches to address the challenges explored in this paper.

B THE SPARSEFW ALGORITHM

We state the full SparseFW algorithm in Algorithm 2, which includes the details about how the fraction α of weights fixed to one is implemented. Before running FW, we compute the number of weights to keep based on saliency $k_{\text{keep}} = \lfloor k \cdot \alpha \rfloor$ and compute the mask of the weights to keep $\overline{M}$ by setting the k_{keep} weights with the highest Wanda saliency scores S to one and the remaining weights to zero. Then we apply FW to the remaining weights with the adjusted sparsity budget $k_{\text{new}} = k(1-\alpha)$. Finally, we threshold the resulting mask M_T by keeping its k_{new} largest entries to obtain a binary mask M^*, and return $M^* + \overline{M}$, which preserves the salient weights and yields exactly k nonzeros.

Algorithm 2 The SparseFW algorithm

Require: Weight matrix W, input data X, nonzero entries k, maximum iterations T, warm-start saliency matrix S, fraction of weights to keep from saliency α

$k_{\text{keep}} \leftarrow \lfloor k \cdot \alpha \rfloor$ ▷ Number of weights retained based on saliency
$k_{\text{new}} \leftarrow \lfloor k(1-\alpha) \rfloor$ ▷ Remaining budget
$\overline{M}_{ij} \leftarrow 1$ for $(i,j) \in \texttt{Top-k}_{\text{keep}}(S)$, 0 otherwise ▷ Fixed (preserved) mask
$G = XX^\top$, $H = WG$ ▷ Precompute caches
for $t = 0$ to $T-1$ **do**
  $\nabla f(M_t) = -2 \cdot W \odot (H - (W \odot M_t)G)$ ▷ Compute gradient
  $V_t = \text{LMO}\big(\nabla f(M_t) \odot (1-\overline{M}), \mathcal{C}_{k_{\text{new}}}\big)$ ▷ LMO on unfixed coordinates
  $\eta_t \leftarrow \frac{2}{t+2}$
  $M_{t+1} \leftarrow (1-\eta_t)M_t + \eta_t V_t$ ▷ FW Update
$M^*_{ij} \leftarrow 1$ if $(i,j) \in \texttt{Top-k}_{\text{new}}(M_T)$ else 0 ▷ Threshold
return $M^* + \overline{M}$

C RATIO OF FIXED WEIGHTS ABLATION

Table 2 illustrates how the ratio α of fixed weights impacts SparseFW performance. Optimal results occur mostly at $\alpha = 0.9$, though even a small α (e.g., $\alpha = 0.1$) significantly enhance perplexity. Conversely, $\alpha = 0.0$ (full FW with no fixed weights) consistently underperforms compared to the baselines.

Table 2: Perplexity (↓, lower is better) comparison on WikiText. We report SparseFW performance with after 2000 iterations using 256 samples with Wanda warmstart for unstructured 60% sparsity and semi-structured 2:4 sparsity for different ratios α of mask entries fixed to one (see Algorithm 2). Here, $\alpha = 1.0$ corresponds to the Wanda baseline, as no further mask entries can be optimized. Best values per row are highlighted in bold. The Wanda column provides a baseline for comparison.

		α-ratio of fixed weights						
Model	Sparsity	0.0	0.1	0.25	0.5	0.75	0.9	1.0 (Wanda)
Gemma-2-9B	2:4	17.70	16.69	16.78	16.48	15.99	**15.81**	17.41
Yi-1.5-9B	2:4	12.26	11.50	11.49	11.25	10.83	**10.61**	11.58
DeepSeek-7B	2:4	13.25	12.77	13.13	12.99	12.32	**11.73**	11.76
Qwen2.5-7B	2:4	16.16	14.96	15.06	15.21	14.59	**14.16**	14.40
Qwen2.5-14B	2:4	13.70	12.62	13.34	12.99	12.79	11.82	**11.37**
Llama-3.1-8B	2:4	21.95	20.47	**20.45**	21.77	21.73	21.49	24.82
Gemma-2-9B	60%	18.25	16.41	15.78	15.46	14.92	**14.83**	16.46
Yi-1.5-9B	60%	11.19	**10.56**	10.66	10.81	11.06	11.31	11.38
DeepSeek-7B	60%	12.49	11.99	12.06	12.19	12.20	12.21	**11.44**
Qwen-7B	60%	14.28	13.13	13.12	12.73	**12.44**	12.54	13.47
Qwen-14B	60%	11.59	10.52	10.48	10.61	10.29	**10.28**	10.87
Llama-3.1-8B	60%	22.47	18.96	**17.97**	18.04	18.27	19.07	21.53

D LMOS FOR SEMI-STRUCTURED SPARSITY

Recall the definition of the constraint set $\mathcal{C}_k$ from Equation (10) for the unstructured sparsity case:

$$\mathcal{C}_k = \left\{ M \in [0,1]^{d_{\text{out}} \times d_{\text{in}}} : \|M\|_1 \leq k \right\}.$$

For the $n{:}m$ sparsity case, which corresponds to keeping at most m nonzeros in every group of n consecutive entries of each row, and assuming d_{in} is divisible by n, we can write the constraint set as

$$\mathcal{C}_{n:m} = \left\{ M \in [0,1]^{d_{\text{out}} \times d_{\text{in}}} \mid \sum_{j=qn+1}^{(q+1)n} M_{i,j} \leq m,\ \forall i,\ \forall q \in \{0, \ldots, d_{\text{in}}/n - 1\} \right\}.$$

Notice that this constraint set is simply the cartesian product of the constraint set for each block of n consecutive entries of each row, which can be written as

$$\mathcal{C}' = \left\{ M' \in [0,1]^n : \|M'\|_1 \leq m \right\},$$

which is a special case of the polytope $\mathcal{C}_k$ when $d_{\text{out}} = n$, $d_{\text{in}} = 1$ and $k = m$. Since we know the LMO for $\mathcal{C}_k$ and the LMO problem is fully separable between the $\mathcal{C}'$ sets, we can simply apply the LMO for $\mathcal{C}_k$ to each set $\mathcal{C}'$ individually to obtain the LMO for $\mathcal{C}_{n:m}$.

E THEORETICAL GUARANTEE FOR SPARSEFW

For simplicity, we work in the row-wise formulation; the proof for the full-matrix case follows by the same arguments. Let us introduce the relevant notation and definitions for the row-wise formulation. We first fix $w \in \mathbb{R}^d_{\text{in}}$ (a row of W) and $X \in \mathbb{R}^{d_{\text{in}} \times B}$. For $m \in \mathcal{C}_k$ as defined in Equation (10), the objective function is

$$f(m) := \|w^\top X - (w \odot m)^\top X\|_2^2 = (1-m)^\top Q (1-m),$$

where $Q := \mathrm{Diag}(w)\,(XX^\top)\,\mathrm{Diag}(w) \succeq 0$. Let $\lambda_{\max}(Q)$ denote the top eigenvalue of Q. We denote the combinatorial constraint of the original problem Equation (MASK SELECTION) as

$$\mathcal{C}_{\text{int}} := \Big\{ m \in \{0,1\}^{d_{\text{in}}} \;\Big|\; \sum_j m_j = k \Big\}.$$

Now we denote by m^* the solution to the relaxed problem (RELAXED MASK SEL.) and by m^{int} the solution to the integral problem (MASK SELECTION).

Lemma 2 (Formal statement of Lemma 1)**.** *Let $m^\varepsilon \in \mathcal{C}_k$ satisfy $\sum_j m_j^\varepsilon = k$ and $f(m^\varepsilon) \le f(m^*)+\varepsilon$. Let $\widehat{m} := \mathbf{1}\{\, j \in \texttt{Top-}k(m^\varepsilon)\,\}$ be its top-k rounding. Then, with $r := d_{in} - k$,*

$$f(\widehat{m}) - f(m^{\text{int}}) \le \varepsilon \;+\; 2\,\lambda_{\max}(Q)\,\Big(\min\{k,r\} + \sqrt{2r\,\min\{k,r\}}\Big). \tag{13}$$

Note that for sparsity 50% or more, we have $2k \le d_{in}$ and hence $\min\{k,r\} = k$, it follows that

$$f(\widehat{m}) - f(m^{\text{int}}) \le \varepsilon + 2\lambda_{\max}(Q)(k + \sqrt{2d_{in}\cdot k}). \tag{14}$$

Proof of Lemma 2. Our goal is to bound $f(\widehat{m}) - f(m^{\text{int}})$. To that end, first note that

$$f(m^\varepsilon) \le f(m^*) + \varepsilon \le f(m^{\text{int}}) + \varepsilon, \tag{15}$$

where the first inequality follows by assumption on m^ε and the second inequality follows since by the optimality of m^* we have $f(m^*) \le f(m^{\text{int}})$ (restricting to the $\mathcal{C}_{\text{int}}$ can only make the objective worse). Therefore it suffices to bound $f(\widehat{m}) - f(m^\varepsilon)$.

Set $v := \widehat{m} - m^\varepsilon$ and $z^\varepsilon := \mathbf{1} - m^\varepsilon$. By construction, $\sum_j \widehat{m}_j = \sum_j m_j^\varepsilon = k$, hence $\mathbf{1}^\top v = 0$. Let

$$\tau := \sum_{j \notin \texttt{Top-k}(m^\varepsilon)} m_j^\varepsilon = k - \sum_{j \in \texttt{Top-k}(m^\varepsilon)} m_j^\varepsilon.$$

Then we have that

$$\begin{aligned} f(\widehat{m}) - f(m^\varepsilon) &= (z^\varepsilon - v)^\top Q (z^\varepsilon - v) - (z^\varepsilon)^\top Q z^\varepsilon \\ &= v^\top Q v \;-\; 2\, z^{\varepsilon\top} Q v \\ &\le \lambda_{\max}(Q)\,\|v\|_2^2 + 2\lambda_{\max}(Q)\,\|z^\varepsilon\|_2\,\|v\|_2 \\ &\le \lambda_{\max}(Q)\big(2\tau\big) + 2\lambda_{\max}(Q)\sqrt{r}\sqrt{2\tau}, \end{aligned}$$

where the equalities follow by defintion of f and the first inequality follows by Cauchy-Schwarz. For the second inequality, consider that we have that $\|v\|_1 = 2\tau$ and $|v_j| \le 1$, hence $\|v\|_2^2 \le \|v\|_1 = 2\tau$. Further, we have that $\|z^\varepsilon\|_2^2 \le \|z^\varepsilon\|_1 = \sum_j (1 - m_j^\varepsilon) = d_{\text{in}} - k = r$.

Lastly, we note that $\tau \le \min\{k,r\}$. This holds since we have that $\tau \le \sum_j m_j = k$ and

$$\tau = \sum_{j \notin \texttt{Top-k}(m^\varepsilon)} m_j^\varepsilon \le \sum_{j \notin \texttt{Top-k}(m^\varepsilon)} 1 \le d_{\text{in}} - k$$

where the inequality follows since each $m_j \le 1$, and there are at most $d_{\text{in}} - k$ terms in that sum. This concludes the proof for the Equation (13) and the proof of the Equation (14) follows by simple computations.

□