T-REX: Learning Tile-Reuse Indexes for Structured Model Compression

Haoyi Yang
AIML, TU Darmstadt
haoyi.yang@tu-darmstadt.de

Simon Kohaut
AIML, TU Darmstadt
simon.kohaut@cs.tu-darmstadt.de

Zihan Ye
AIML, TU Darmstadt
zihan.ye@tu-darmstadt.de

Devendra Singh Dhami
TU Eindhoven
d.s.dhami@tue.nl

Kristian Kersting
AIML, TU Darmstadt
kristian.kersting@tu-darmstadt.de

Abstract

The rapid proliferation of large-scale machine learning models, such as LLMs, has driven remarkable progress across application domains. Nevertheless, their scale comes with significant deployment challenges and inherent redundancies that remain open challenges to this day. In this work, we systematically exploit these redundancies to offer a principled path toward compression without sacrificing capability. To this end, we propose the Tile-Reuse Index (T-REX), a structured model compression method that identifies and indexes reusable, contiguous parameter groups (tiles) across a model's architecture. By learning a compact set of matching tiles, we compress the model's memory footprint while preserving its performance. Furthermore, by reconstructing dense layers on the fly at inference time, T-REX compressed models preserve inference compatibility without requiring architectural modifications. In our experiments, we show that T-REX outperforms existing post-training compression methods across benchmarks by most closely matching the original model's performance. We furthermore show how post-training on top of T-REX enables small footprint models to recover most of the performance loss introduced by compression.

1 Introduction

Today, large-scale models such as LLMs [OpenAI, 2023, Yang et al., 2025, DeepSeek-AI, 2026] are deeply embedded in a broad spectrum of everyday applications, reflecting their impressive performance and versatility across domains, from natural language to code [Paul et al., 2025]. However, their capabilities necessitate substantial memory and compute budgets, which prevent their deployment outside of large data centers, e.g., onboard autonomous systems that couple learned perception with symbolic reasoning Kohaut et al. [2025], Skryagin et al. [2026]. Thus, efficient compression techniques have garnered significant attention [Hoefler et al., 2021], but a technical gap in preserving the original model's performance remains.

Pruning weights [Han et al., 2015b, Frankle and Carbin, 2018] with small impact on activations provides a naive path to compression by zeroing out parameters at different levels of granularity, from individual weights [Sun et al., 2023, Frantar and Alistarh, 2023] to entire blocks [Zhong et al.,

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

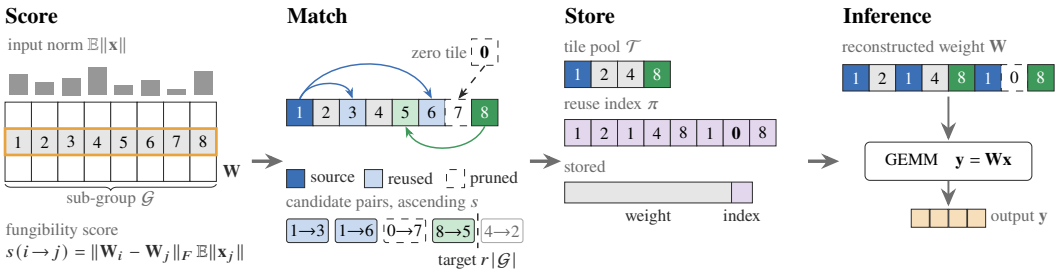


Figure 1: **T-REX model compression via tile-matching:** By systematically analyzing the fungibility of contiguous parameters in a pretrained model (tile) based on their similarity and output behavior, T-REX produces compressed models that stick close to the original's performance and allow for effective post-training. From left to right: tiles are scored (Eq. 3), greedily reused or pruned (Algorithm 1), stored as a tile pool with a reuse index, and rebuilt into the dense weight for a standard GEMM at inference (Sec. 3.3).

2025]. However, previously explored methods sit at undesirable extremes of granularity: on the one hand, coarse pruning incurs significant performance losses, whereas fine-granular approaches fail to yield meaningful compression rates. Beyond this, although the zero-only pruning paradigm can reduce computation and memory footprint in some scenarios, it does not exploit similarity among the surviving weights. By discarding parameter blocks rather than simply increasing the sparsity, some methods achieve an effective size reduction. Nevertheless, this approach necessarily leads to a loss of model capabilities and prohibits effective recovery via post-training.

In this work, we propose learning Tile-Reuse Indexes (T-REX), a novel structured model compression method that reduces the effective size of a model by reusing and pruning its weights at the tile level. To achieve this, we introduce a novel score that allows T-REX to jointly consider the values within and activations of contiguous groups of model parameters (tiles). Hereby, T-REX outperforms existing compression methods limited to raising sparsity or discarding parts of the model architecture.

T-REX operates by partitioning the weight tensors of a pretrained model into contiguous, non-overlapping parameter tiles and systematically identifying groups of tiles that are sufficiently similar to be treated as fungible. Rather than discarding or zeroing out redundant parameters, T-REX builds an index that couples matching tiles by replacing them with a reference to a shared source tile. T-REX constructs this reuse index via a calibrated, greedy matching algorithm that jointly scores tile similarity and activation magnitude, ensuring that replacements incur minimal disruption to the model's behavior. We illustrate the T-REX compression scheme in Figure 1.

The resulting compressed model retains the same architectural footprint but operates over a reduced set of unique parameter tiles, enabling control of memory savings and impact on inference latency via T-REX's tiling granularity. Notably, T-REX unifies pruning and reuse under a single framework: tiles that are close to zero are pruned outright, while those with a nearby surviving counterpart are redirected, allowing the method to gracefully interpolate between pure sparsity and full tile reuse depending on the structure of each layer's weight distribution.

In summary, we make the following contributions to structured model compression.

- We propose a novel fungibility scoring for contiguous parameter groups (tiles) that jointly considers similarity and activation impact to identify redundant substructures within large machine learning models.
- We present T-REX (Tile-Reuse Index), a loss-conserving model compression algorithm based on our fungibility score that searches for and fine-tunes contiguous, redundant parameter groups, leading to a tangible reduction in a model's memory footprint.
- We demonstrate that T-REX consistently outperforms state-of-the-art structured compression baselines across compression ratios and model sizes, and show that the tile-reuse structure is amenable to post-compression fine-tuning for further performance recovery.

For reproducibility, we make the source code[1] and our evaluation pipeline publicly available.

2 Related Work

Model compression has become a central topic in modern deep learning, as LLMs and other over-parameterized architectures impose substantial memory and inference costs that prohibit their application outside large-scale data centers and instead on, e.g., consumer-grade hardware.

Pruning. Pruning compresses neural networks by selecting parameters or structures to remove according to an estimated saliency score. In terms of scoring, simple magnitude-based criteria are inexpensive but often insufficient because they ignore activation statistics and parameter interactions. Classical second-order methods such as Optimal Brain Damage and Optimal Brain Surgeon provide a more principled estimate of removal-induced loss [LeCun et al., 1989, Hassibi et al., 1993], but accurate Hessian-based scoring is prohibitively expensive for modern LLMs. In terms of granularity, pruning may operate on individual weights[Sun et al., 2023, Frantar and Alistarh, 2023], contiguous blocks, rows or columns of weight matrices [Ashkboos et al., 2024], or higher-level structures such as neurons, channels [Liu et al., 2017, Molchanov et al., 2019], and attention heads [Michel et al., 2019]; recent LLM pruning methods often combine several of these axes under a unified structured compression framework [Ma et al., 2023, Muñoz et al., 2025]. Unstructured pruning can achieve high nominal sparsity, but its irregular patterns often incur indexing overhead and require specialized kernels to yield practical speedups. Block-sparse pruning addresses this limitation by removing contiguous weight regions, which reduces metadata overhead and better aligns the sparsity pattern with cache behavior, dense sub-kernels, and hardware-supported sparse execution [Narang et al., 2017, Mishra et al., 2021, Zhou et al., 2021]. However, these methods still fundamentally compress by deleting or zeroing parameters, and coarser granularities can cause substantial degradation. Unlike these pruning-based methods, T-REX treats contiguous tiles as reusable units: low-impact tiles may be pruned, while redundant tiles can be redirected to similar surviving tiles, enabling compression through both elimination and reuse.

Parameter Reuse. Another line of work obtains smaller models by reducing the number of distinct parameters through reuse or sharing. [Nowlan and Hinton, 2018, Chen et al., 2015] Weight tying reuses matrices across embedding spaces [Press and Wolf, 2017], while ALBERT and recurrent or looped Transformers share Transformer blocks across depth or iterations [Lan et al., 2019, Dehghani et al., 2018, Giannou et al., 2023, Zeitoun et al., 2026]. These approaches reduce the number of distinct trainable parameters, but the sharing pattern is typically built into the architecture and pre-training phase. Post-training methods often implement reuse through codebooks: scalar clustering in Deep Compression [Han et al., 2015a], product quantization [Stock et al., 2019], and recent vector-quantization-based LLM compressors [Tseng et al., 2024] replace many weights or weight vectors with shared representatives. Although these methods also use indices, their indices typically point to learned codebook entries or quantized representatives rather than to existing parameter blocks in the original model. In contrast, T-REX performs post-training reuse at the tile level: it forms the reusable set from surviving tiles already present in the model and redirects redundant contiguous tiles to similar surviving tiles, guided by a fungibility score that accounts for both weight similarity and activation impact.

3 Structured Model Compression via Fungibility Scoring and Indexing

In this section, we present the two main ingredients making up T-REX. First, we present a tile-fungibility scoring function, allowing T-REX to weight the similarity between parameter groups to strategically assign them to a group. Second, we show T-REX tile pool and reuse index construction, which enables its structured model compression under loss preservation.

3.1 Layer Weight Partitioning

Let a linear layer be computed as $\mathbf{Y} = \mathbf{X}\mathbf{W}$, where $\mathbf{X} \in \mathbb{R}^{T \times d_{\text{in}}}$ is the input activation and $\mathbf{Y} \in \mathbb{R}^{T \times d_{\text{out}}}$ is the output with sequence length and input- and output-dimensions $T, d_{\text{in}}, d_{\text{out}} \in \mathbb{N}^+$.

[1] T-REX https://github.com/haoyi-yang/TREX

We partition the weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ of each linear layer into non-overlapping tiles of size $n \times n, n \in \mathbb{N}^+$. If a dimension is not divisible by n, the remaining strip of at most $n-1$ rows or columns is left uncompressed or zero-padded, with a negligible effect on the compression ratio. For each group of n consecutive output dimensions, the corresponding $\lfloor d_{\text{in}}/n \rfloor$ tiles form an **output tile group**. We further partition each output tile group into sub-groups $\mathcal{G}$ of size n_g, where each sub-group serves as the basic unit for reuse planning. Under this tiling, the layer output decomposes as

$$\mathbf{Y}_{:,m} = \sum_{\mathcal{G}} \sum_{k \in \mathcal{G}} \mathbf{X}_{:,k} \, \mathbf{W}_{k,m}. \tag{1}$$

In the following, we treat each position in the input sequence as an independent vector $\mathbf{x}$ in place of the complete matrix $\mathbf{X}$.

3.2 Tile Fungibility Scoring

Tile Fungibility. In order to setup T-REX tile reuse for structured model compression, we propose pairwise fungibility scores to indicate tiles that are *fungible*, i.e., one may be replaced with the other. We compute fungibility, and establish reuse relationships, over all tile pairs within each sub-group independently. For a candidate pair ($\mathbf{W}_i, \mathbf{W}_j \in \mathcal{G}$), the reuse score quantifies the cost of replacing **target** tile $\mathbf{W}_j$ with **source** tile $\mathbf{W}_i$. By the submultiplicativity of matrix norms [Bernstein, 2009], the induced change in output satisfies

$$\|(\mathbf{W}_i - \mathbf{W}_j)\mathbf{x}_j\| \leq \|\mathbf{W}_i - \mathbf{W}_j\|_F \cdot \|\mathbf{x}_j\|. \tag{2}$$

We use the right-hand side as a tractable upper bound on replacement cost, and define our fungibility score thus as

$$s(i \to j) = \|\mathbf{W}_i - \mathbf{W}_j\|_F \cdot \mathbb{E}[\|\mathbf{x}_j\|], \tag{3}$$

where the activation norm $\mathbb{E}[\|x\|]$ is estimated as the mean over a calibration dataset. Due to their pairwise independence, scores can be computed in parallel across sub-groups and layers.

Virtual Zero Tile. To unify pruning and reuse under the same framework, we additionally introduce a virtual zero tile $\mathbf{W}_0 = \mathbf{0}$ as a special candidate source participating in the pairwise fungibility scoring. Its reuse score follows directly as a special case of Eq. (3): since $\mathbf{W}_i = \mathbf{0}$, the score reduces to

$$s(\mathbf{0} \to j) = \|\mathbf{W}_j\|_F \cdot \mathbb{E}[\|\mathbf{x}_j\|], \tag{4}$$

recovering a calibration-weighted magnitude criterion. Consequently, tiles that are close to zero will naturally receive low pruning scores and are eliminated outright rather than redirected, without requiring a separate pruning criterion or threshold.

With the triangle inequality, Eq. (2) extends from single tiles to a whole output tile group: if $\mathcal{D}_m$ is the set of tiles deleted in group m and $\pi(j)$ the source of $j \in \mathcal{D}_m$, the expected per-token change of the group output satisfies $\mathbb{E}\|\Delta \mathbf{y}_m\| \leq \sum_{j \in \mathcal{D}_m} s(\pi(j) \to j)$. The reuse index construction below therefore greedily minimizes an upper bound on the output perturbation of each layer, with pruning as the special case $\mathbf{W}_{\pi(j)} = \mathbf{0}$; Appendix E discusses the tightness of this bound and the optimality of the greedy matching.

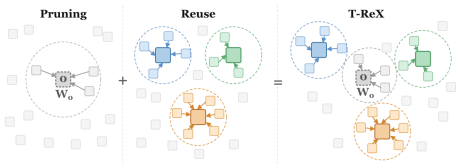


Figure 2: **T-REX unifies pruning and weight reuse:** While many pruning methods aim to map weights with small importance to the activations to zero, we incorporate weight reuse by identifying fungible parameter groups and mapping them to a shared center. T-REX combines these two approaches into a unified, structured model compression algorithm.

Geometric Interpretation. This gives T-REX a clean clustering view, illustrated in Figure 2: each tile is assigned to its nearest surviving centre, while a tile assigned to $\mathbf{W}_0$ is simply pruned. Structured pruning is the degenerate case where $\mathbf{W}_0$ is the *only* admissible centre; T-REX lifts this restriction by allowing multiple adaptive centres selected from the weight distribution itself.

Algorithm 1 Greedy Reuse Index Construction

Require: Weight tiles $\{\mathbf{W}_{k,m}\}$, calibration activation norms $\{\mathbb{E}[\|\mathbf{x}_k\|]\}$, target tile deletion ratio r
Ensure: Tile pool $\mathcal{T}$, reuse index $\pi : (j, m) \mapsto (i, m)$

$\pi \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$; define $\mathbf{W}_{\mathbf{0},m} \leftarrow \mathbf{0}$ for all m
for each output tile group m **do**
  for each sub-group $\mathcal{G}$ **do**
    $\mathcal{P}_\mathcal{G}, \mathcal{D}_\mathcal{G}, \mathcal{S}_\mathcal{G} \leftarrow \emptyset$ ▷ candidate pairs, deleted tiles, source tiles
    for each pair (i, j) in $(\mathcal{G} \cup \{\mathbf{0}\}) \times \mathcal{G}$, $i \neq j$ **do**
      $s \leftarrow \|\mathbf{W}_{i,m} - \mathbf{W}_{j,m}\|_F \cdot \mathbb{E}[\|\mathbf{x}_j\|]$
      $\mathcal{P}_\mathcal{G} \leftarrow \mathcal{P}_\mathcal{G} \cup \{(s, i, j)\}$
    Sort $\mathcal{P}_\mathcal{G}$ ascending by s; $N_{\text{target}} \leftarrow \lfloor r|\mathcal{G}| \rfloor$
    for each (s, i, j) in $\mathcal{P}_\mathcal{G}$ **do**
      if $|\mathcal{D}_\mathcal{G}| \geq N_{\text{target}}$ **then break**
      if $(i, m) \in \mathcal{D}_\mathcal{G}$ **or** $(j, m) \in \mathcal{D}_\mathcal{G}$ **or** $(j, m) \in \mathcal{S}_\mathcal{G}$ **then continue**
      $\mathcal{D}_\mathcal{G} \leftarrow \mathcal{D}_\mathcal{G} \cup \{(j, m)\}$; $\pi[j, m] \leftarrow (i, m)$
      if $i \neq \mathbf{0}$ **then** $\mathcal{S}_\mathcal{G} \leftarrow \mathcal{S}_\mathcal{G} \cup \{(i, m)\}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(k, m) \mid k \in \mathcal{G},\ (k, m) \notin \mathcal{D}_\mathcal{G}\}$
return $\mathcal{T}$, π

3.3 Tile Pool & Reuse Index Construction.

Reuse Index Construction. Given the fungibility scores defined in Eq. (3), we now describe how T-REX constructs a compressed representation of a weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. We show the T-REX index construction algorithm in Algorithm 1. Given all pairwise scores (including zero-tile scores) of a sub-group $\mathcal{G}$, we construct a reuse index via 1. All candidate pairs $(i \to j)$ within the same sub-group and layers are sorted globally by score in ascending order. We then iterate through the sorted list: when a pair $(i \to j)$ is selected, target tile $\mathbf{W}_j$ is assigned to source $\mathbf{W}_i$ and subsequently **discarded**. This continues until the target deletion ratio $r \in [0, 1]$ is reached.

A deleted tile can neither serve as a valid replacement nor be replaced again. Hence, during iteration, if either tile of a candidate pair has already been deleted in a previous step, the pair is skipped. In the resulting reuse index, each deleted tile $\mathbf{W}_j$ is mapped to a surviving source tile $\mathbf{W}_i$ (or the zero tile $\mathbf{W}_0$), and a single source tile may serve as the replacement for multiple targets.

Reuse Pool & Index. The compressed representation is stored as a pair $(\mathcal{T}, \pi)$, where the tile pool $\mathcal{T}$ contains the weights of all surviving tiles, and the reuse index π maps every tile position (k, m) to its assigned source within the corresponding sub-group. With weights and indexes stored in 16-bit precision (see Appendix D for the general case), under a uniform deletion ratio r, the tile pool retains $(1 - r)$ of the original parameters, and the reuse index contributes $1/n^2$ of the original footprint. The total compression ratio is thus

$$\rho = (1 - r) + \frac{1}{n^2}. \tag{5}$$

The tile size n governs a secondary trade-off: small n yield fine-granular reuse opportunities but inflate the index overhead, while large n reduces indexing cost at the expense of matching granularity.

Forward Pass. The reuse index π is frozen after compression and requires no modification to the model architecture. At inference time, T-REX reconstructs the full weight matrix on-the-fly by expanding each tile position (k, m) to its assigned source $\mathbf{W}_{\pi(k,m)}$ from the tile pool $\mathcal{T}$, yielding a dense matrix of the original shape, after which standard matrix multiplication proceeds without modification. This reconstruction is performed one layer at a time and the reconstructed matrix is released after its GEMM, so the full dense model is never materialized.

4 Experiment

In our experiments, we evaluate T-REX's compression capabilities compared to state-of-the-art structured model compression methods. More specifically, we aim to answer the following questions:

Q1: How does T-REX compare to other structured pruning methods regarding loss preservation at the same target compression ratios?

Q2: How does tile-reuse improve T-REX loss preservation over pruning alone?

Q3: How much performance is restored when post-training a T-REX compressed model?

Q4: How sensitive is a T-REX compressed model's performance to tile and sub-group size?

Q5: How much latency is incurred on a T-REX compressed model's forward pass via the tile-index?

4.1 Experimental Setup

We evaluate T-REX on Qwen3-8B and Qwen3-14B [Yang et al., 2025]. For this, we sample calibration data from the C4 dataset [Raffel et al., 2020] using 128 samples. All experiments are conducted on a single NVIDIA A100 (80 GB) GPU. Compression index construction is performed in a single forward pass over the calibration set. For the fine-tuned variants (T-REX(FT)), we apply LoRA fine-tuning on C4 with the reuse index held fixed after compression. Unless otherwise stated, we use a default tile size of 4×4 and sub-group size 1024; with dedicated experiments regarding T-REX's sensitivity to these hyperparameters presented in Section 4.5.

We employ the LM Evaluation Harness [Gao et al., 2024] to assess downstream task performance on five benchmarks: PIQA [Bisk et al., 2020], WinoGrande [Sakaguchi et al., 2021], HellaSwag [Zellers et al., 2019], ARC-Easy and ARC Challenge [Clark et al., 2018]. Additionally, we report Perplexity (PPL) on the WikiText-2 dataset [Merity et al., 2016] to further quantify general language modeling quality after compression.

4.2 Loss Preservation Comparison Study

We compare T-REX against four structured compression baselines at target compression ratios of 20% and 30%. **LLM-Pruner** [Ma et al., 2023] identifies and removes coupled structures such as neurons and attention heads via gradient-based dependency analysis. **SliceGPT** [Ashkboos et al., 2024] compresses each weight matrix by deleting rows and columns through PCA-based orthogonal transformations. **BlockPruner** [Zhong et al., 2025] removes contiguous blocks of weights at a coarser granularity based on block-level importance scores. **MultiPruner** [Muñoz et al., 2025] integrates multiple pruning strategies, including width and depth pruning, under a unified importance-scoring framework. All baselines are applied under identical compression budgets; T-REX's reported ratio accounts for the full memory footprint—including tile pool and reuse index—ensuring a fair comparison.

Results are reported per downstream task for both Qwen3-8B and Qwen3-14B in Table 1. The results show that tile-granular reuse consistently outperforms other structured pruning methods on most tasks and across model sizes, particularly at higher compression ratios where less granular approaches suffer from significant accuracy degradation. Notably, T-REX achieves this while retaining *fewer* effective parameters than competing methods at the same target ratio: for instance, at 20% compression on Qwen3-8B, T-REX operates with only 6.12B effective weights compared to 6.48–6.55B for baselines, yet delivers superior downstream accuracy and substantially lower perplexity.

This highlights a key advantage of tile reuse: redirecting redundant tiles to surviving counterparts preserves more functional capacity per retained parameter than outright deletion. The performance gap widens at the 30% compression ratio, where baseline methods exhibit sharp perplexity increases and notable benchmark degradation, while T-REX maintains comparatively stable performance across all evaluated tasks on both model scales.

4.3 Tile-Reuse Ablation Study

To isolate the contribution of the reuse mechanism, we introduce **T-REX-P**, a restricted variant of T-REX in which the zero tile is the only admissible source with Eq. 4 and no reuse relationships are established between surviving and removed tiles. T-REX-P can therefore be interpreted as a structured magnitude pruning baseline within our framework.

Both variants use the same greedy selection procedure at identical compression ratios. The improvement of T-REX over T-REX-P demonstrates that redirecting tiles to nearby survivors preserves more

Table 1: **T-REX outperforms state-of-the-art structured compression methods in loss preservation:** Results on Qwen3-8B and Qwen3-14B; T-REX's lead over baselines grows as the original model size increases. We show absolute losses for target compression rates of 20% (above) and 30% (below) within each model, respectively. Best result per compression rate in **bold**, second best underlined.

Model	Method	Compression		PPL ↓	Benchmark ↑					
		Ratio%	Eff.Wt		ARC-c	ARC-e	HellaS	PIQA	WinoG	Average
Qwen3-8B	*Original*	*0*	8.19B	9.90	56.48	80.85	74.93	77.69	67.72	71.53
	LLM-Pruner	20.00	6.55B	16.78	33.28	56.65	57.86	72.25	59.59	55.93
	SliceGPT	20.35	6.52B	31.63	34.04	57.74	57.80	71.71	**64.72**	57.20
	BlockPruner	20.89	6.48B	24.81	**43.60**	**68.73**	56.09	72.03	57.77	59.64
	MultiPruner	20.02	6.55B	17.55	42.58	64.31	57.50	72.31	57.70	58.88
	T-REX *(ours)*	20.04	6.12B	**14.39**	43.00	68.31	**61.26**	**73.94**	62.75	**61.85**
	LLM-Pruner	30.00	5.73B	36.56	24.40	42.59	43.88	65.34	53.20	45.88
	SliceGPT	29.26	5.79B	40.40	28.16	44.61	49.08	68.50	**63.38**	50.75
	BlockPruner	31.14	5.64B	53.67	33.02	48.06	46.53	64.69	54.06	49.27
	MultiPruner	30.02	5.73B	44.67	**33.87**	53.66	45.12	64.36	52.17	49.84
	T-REX *(ours)*	30.06	5.29B	**26.28**	33.31	**56.91**	**53.22**	**70.62**	58.59	**54.53**
Qwen3-14B	*Original*	*0*	14.77B	8.93	60.24	82.83	78.85	80.14	72.85	74.98
	LLM-Pruner	19.98	11.82B	14.52	42.06	67.93	67.05	76.06	61.09	62.84
	SliceGPT	21.24	11.63B	24.12	37.63	61.28	63.18	74.76	**71.03**	61.58
	BlockPruner	20.88	11.68B	18.11	43.60	68.64	62.77	73.45	59.04	61.50
	MultiPruner	20.02	11.81B	16.94	44.20	63.01	61.64	71.33	61.56	60.35
	T-REX *(ours)*	19.99	10.99B	**11.84**	**48.55**	**73.82**	**68.96**	**76.44**	67.72	**67.10**
	LLM-Pruner	30.03	10.33B	26.30	30.29	50.80	52.78	68.72	55.96	51.71
	SliceGPT	30.41	10.28B	36.36	31.40	50.63	54.95	70.84	**67.09**	54.98
	BlockPruner	31.21	10.16B	30.11	34.30	54.00	51.78	68.77	55.25	52.82
	MultiPruner	30.03	10.33B	32.85	31.06	48.44	53.22	66.32	56.43	51.09
	T-REX *(ours)*	30.05	9.50B	**15.59**	**41.13**	**67.26**	**59.84**	**73.07**	64.40	**61.14**

performance at no additional parameter cost—a recovery that pure pruning cannot achieve. T-REX-P uses the same tile pool, reuse index and per-layer reconstruction as T-REX (Section 3.3), so both have the same memory footprint and inference cost; they differ only in where the index entries of the modified tiles point (see Appendix C.3 for statistics of the reuse index).

4.4 T-REX Post-Training for Loss Recovery

We investigate whether fine-tuning can recover the performance degradation introduced by T-REX. Fine-tuning is conducted on the C4 dataset with the standard auto-regressive objective; the reuse index is kept fixed throughout, incurring no additional compression overhead. We apply LoRA [Hu et al., 2022] with rank r=16 on all linear sublayers. We train for one epoch on a subset of C4 on a single A100-80GB GPU; the full configuration and training cost are given in Table 5. As shown in Table 3, T-REX(FT) consistently recovers a substantial portion of the compression-induced accuracy loss across both model scales and compression ratios: on Qwen3-8B, average accuracy improves from 61.85% to 67.62% at 20% compression, and from 54.53% to 63.43% at 30%. Comparable gains are observed on Qwen3-14B. Furthermore, T-REX(FT) consistently outperforms T-REX-P(FT) across all settings, suggesting that tile reuse provides a better initialization for post-compression adaptation than pruning alone. A same-budget post-training baseline is reported in Appendix A.4.

Table 2: **Unifying pruning and reuse boosts structured compression performance:** Excluding reuse from T-REX (T-REX-P) results in lower performance, as shown by WikiText-2 perplexity (PPL) and across benchmark scores.

Model	Method	Compression		PPL↓	Benchmark ↑					
		Ratio%	Eff.Wt		ARC-c	ARC-e	HellaS	PIQA	WinoG	Average
Qwen3-8B	*Original*	*0*	8.19B	9.90	56.48	80.85	74.93	77.69	67.72	71.53
	T-REX-P	20	6.12B	54.87	32.59	58.96	45.58	68.28	57.46	52.57
	T-REX	20	6.12B	**14.39**	**43.00**	**68.31**	**61.26**	**73.94**	**62.75**	**61.85**
	T-REX-P	30	5.29B	607.34	21.50	35.10	29.88	55.98	52.25	38.94
	T-REX	30	5.29B	**26.28**	**33.31**	**56.91**	**53.22**	**70.62**	**58.59**	**54.53**
Qwen3-14B	*Original*	*0*	14.77B	8.93	60.24	82.83	78.85	80.14	72.85	74.98
	T-REX-P	20	10.99B	20.07	44.11	73.53	56.59	71.98	65.59	62.36
	T-REX	20	10.99B	**11.84**	**48.55**	**73.82**	**68.96**	**76.44**	**67.72**	**67.10**
	T-REX-P	30	9.50B	108.94	24.83	46.17	37.31	63.71	55.01	45.41
	T-REX	30	9.50B	**15.59**	**41.13**	**67.26**	**59.84**	**73.07**	**64.40**	**61.14**

Table 3: **Tile reuse is a better starting point for fine-tuning than pruning:** T-REX(FT) consistently surpasses T-REX-P(FT) across all model scales and compression ratios, nearly closing the gap to the original model.

Model	Method	Compression		PPL↓	Benchmark ↑					
		Ratio%	Eff.Wt		ARC-c	ARC-e	HellaS	PIQA	WinoG	Average
Qwen3-8B	*Original*	*0*	8.19B	9.90	56.48	80.85	74.93	77.69	67.72	71.53
	T-REX-P(FT)	20	6.16B	11.88	43.69	72.05	63.80	75.14	63.85	63.71
	T-REX(FT)	20	6.16B	**9.85**	**49.49**	**75.76**	**68.92**	**76.55**	**67.40**	**67.62**
	T-REX-P(FT)	30	5.33B	16.58	35.07	60.98	53.66	70.67	60.22	56.12
	T-REX(FT)	30	5.33B	**11.70**	**44.11**	**70.33**	**63.61**	**74.54**	**64.56**	**63.43**
Qwen3-14B	*Original*	*0*	14.77B	8.93	60.24	82.83	78.85	80.14	72.85	74.98
	T-REX-P(FT)	20	11.05B	10.11	51.02	74.58	69.93	77.48	**68.98**	68.40
	T-REX(FT)	20	11.05B	**8.56**	**53.58**	**77.27**	**73.74**	**79.27**	68.51	**70.47**
	T-REX-P(FT)	30	9.56B	13.01	41.04	66.79	62.61	73.99	61.40	61.17
	T-REX(FT)	30	9.56B	**9.85**	**47.70**	**72.22**	**70.14**	**77.58**	**67.01**	**66.93**

4.5 Tile and Sub-Group Sensitivity Analysis

We analyze the effect of two key hyperparameters: the tile size $n \times n$ and the sub-group size g. For each hyperparameter, we vary its value while keeping all other settings fixed, and report perplexity on WikiText-2 as a function of compression ratio.

Tile Size: Figure 3a shows the effect of varying the tile size $n \times n$. Finer tiles yield more granular reuse and better quality, while coarser tiles reduce candidate pairs and may cause suboptimal assignments; very small tiles also increase score computation and storage overhead.

Sub-group Size: Figure 3b shows the effect of varying the sub-group size g. Larger sub-groups expose more candidate pairs within each group, potentially yielding better reuse matches, but at the cost of increased score computation. Smaller sub-groups are faster to process but may miss beneficial reuse opportunities across tiles that are farther apart in the weight matrix.

4.6 T-REX Forward-Pass Runtime Analysis

We measure the runtime overhead introduced by restoring a tiled weight back to a dense matrix before a cuBLAS GEMM on Qwen3-8B. We report t_{restore} and t_{gemm} (ms) across projection groups (`gate/up`, `down`, `q/o`, and `k/v`) at batch sizes of 8 and 16 with a sequence length of 2048. As shown in Table 4(a), the per-weight restoration step contributes only a modest absolute overhead

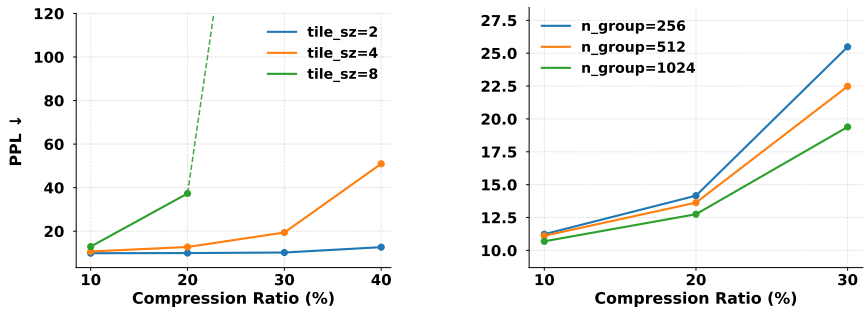
Table 4: **Inference cost of T-REX on Qwen3-8B.** (a) Restoring a tiled weight from its tile pool versus the cuBLAS GEMM on the restored weight at 20% compression on a single A100-80GB, in ms, averaged over the linear layers of a projection group and summed over all 252 linear layers, for 8 and 16 sequences of 2048 tokens. (b) End-to-end inference at 20% compression on the same GPU with batch size 1 and a 4096-token prompt; every linear layer is restored with the same kernel before its GEMM, in prefill and decode alike.

(a) Per linear layer

	Restore	GEMM 8×2048	16×2048
q/o	0.056	2.47	4.92
k/v	0.018	0.61	1.26
gate/up	0.161	7.41	14.73
down	0.148	7.30	14.46
Total	22.3	1018	2026

(b) End to end

	Resident	Prefill	Decode	Peak
Dense	16.38 GB	340.0 ms	32.81 ms/tok	17.43 GB
T-REX	13.10 GB	360.4 ms	46.81 ms/tok	14.44 GB



(a) Tile Size (b) Subgroup Size

Figure 3: **We show hyperparameter impact on T-REX: (a)** Smaller tiles yield lower perplexity across compression ratios, at the cost of increased index overhead. **(b)** Larger sub-groups expose more reuse candidates and improve quality.

compared to the GEMM computation across all evaluated configurations. End-to-end, the cumulative restoration overhead across all layers amounts to ∼22 ms, a cost that could further be reduced or fully masked by a dedicated kernel fusing tile reconstruction directly with the GEMM operation.

End-to-End Inference. Table 4(b) compares end-to-end inference of the dense model and T-REX on Qwen3-8B at 20% compression. Since reconstruction is performed one layer at a time, as described in Section 3.3, peak GPU memory is the compressed footprint and reuse index plus one transient layer, so the memory saving holds throughout inference rather than only at checkpoint load: T-REX reduces resident weights by 20% and peak GPU memory by 17%. Reconstruction adds 6% latency to prefill and 43% to decode, where it is amortized less in the memory-bound phase. T-REX therefore trades latency for memory and does not accelerate inference as dimension-reducing structured pruning can.

5 Conclusion

Summary. We have presented T-REX, a structured model compression framework that leverages the inherent redundancy in the weight distributions of large-scale neural networks. By identifying reusable contiguous parameter tiles within weight tensors, T-REX eliminates redundant memory usage without sacrificing model behavior. Empirically, T-REX achieves competitive memory reduction while preserving loss more faithfully than structured compression baselines. We further demonstrate that

the tile block size is a practically important hyperparameter, allowing to trade off compression rate and fidelity to the original model's loss surface.

Limitations and Future Work. T-REX currently uses a uniform block size; a natural extension is locally adaptive tile granularities, applying finer compression in loss-critical layers and coarser compression elsewhere. Beyond architectural adaptivity, T-REX is orthogonal to sparsity- and pruning-based methods, and combining these paradigms could unlock compression ratios unattainable by either alone. On the system side, realizing the full inference benefits of tile reuse will require dedicated kernels [Dao et al., 2022] and hardware architectures that explicitly account for tile fungibility. Finally, the optimal tile structure is likely task-dependent: workloads such as image generation may tolerate substantially different compression profiles than complex reasoning tasks, suggesting that task-aware compression objectives and subsequent fine-tuning are a promising direction for maximizing downstream utility under a given resource budget. Lastly, the gains of T-REX depend on how similar the tiles of a model are, which varies across model families (Appendix C.4).

Acknowledgments and Disclosure of Funding

This work was further supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy through the Excellence Clusters "Reasonable AI" (EXC 3057) and "The Adaptive Mind" (EXC 3066). Simon Kohaut gratefully acknowledges the financial support from Honda Research Institute Europe (HRI-EU). The Eindhoven University of Technology authors received support from their Department of Mathematics and Computer Science and the Eindhoven Artificial Intelligence Systems Institute.

References

S. Ashkboos, M. L. Croci, M. G. d. Nascimento, T. Hoefler, and J. Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

D. S. Bernstein. *Matrix mathematics: theory, facts, and formulas*. Princeton university press, 2009.

Y. Bisk, R. Zellers, J. Gao, Y. Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, 2020.

W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *International conference on machine learning*, pages 2285–2294. PMLR, 2015.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

E. Frantar and D. Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*. PMLR, 2023.

L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

A. Giannou, S. Rajput, J.-y. Sohn, K. Lee, J. D. Lee, and D. Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pages 11398–11442. PMLR, 2023.

S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 2015b.

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

T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

S. Kohaut, F. Divo, B. Flade, D. S. Dhami, J. Eggert, and K. Kersting. The constitutional filter: Bayesian estimation of compliant agents. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3092–3099, 2025. doi: 10.1109/IROS60139.2025.11246215.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Y. LeCun, J. Denker, and S. Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017.

X. Ma, G. Fang, and X. Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 2023.

S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models, 2016.

P. Michel, O. Levy, and G. Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

A. Mishra, J. A. Latorre, J. Pool, D. Stosic, D. Stosic, G. Venkatesh, C. Yu, and P. Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272, 2019.

J. P. Muñoz, J. Yuan, and N. Jain. Multipruner: Balanced structure removal in foundation models. *arXiv preprint arXiv:2501.09949*, 2025.

S. Narang, E. Undersander, and G. Diamos. Block-sparse recurrent neural networks. *arXiv preprint arXiv:1711.02782*, 2017.

S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight sharing. In *The mathematics of generalization*, pages 373–394. CRC Press, 2018.

R. OpenAI. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5):1, 2023.

I. Paul, H. Yang, G. Glavaš, K. Kersting, and I. Gurevych. ObscuraCoder: Powering Efficient Code LM Pre-Training via Obfuscation Grounding. In *International Conference on Learning Representations*, 2025.

O. Press and L. Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, 2017.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 2021.

A. Skryagin, D. Ochs, P. Deibert, S. Kohaut, D. S. Dhami, and K. Kersting. Answer set networks: Casting answer set programming into deep learning, 2026. URL https://arxiv.org/abs/2412.14814.

P. Stock, A. Joulin, R. Gribonval, B. Graham, and H. Jégou. And the bit goes down: Revisiting the quantization of neural networks. *arXiv preprint arXiv:1907.05686*, 2019.

M. Sun, Z. Liu, A. Bair, and J. Z. Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

V. Thakkar, P. Ramani, C. Cecka, A. Shivam, H. Lu, E. Yan, J. Kosaian, M. Hoemmen, H. Wu, A. Kerr, M. Nicely, D. Merrill, D. Blasig, A. Atluri, F. Qiao, P. Majcher, P. Springer, M. Hohnerbach, J. Wang, and M. Gupta. CUTLASS, Jan. 2023. URL https://github.com/NVIDIA/cutlass.

A. Tseng, J. Chee, Q. Sun, V. Kuleshov, and C. De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *Proceedings of machine learning research*, 235: 48630, 2024.

A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

A. Zeitoun, L. Torroba-Hennigen, and Y. Kim. Hyperloop transformers. *arXiv preprint arXiv:2604.21254*, 2026.

R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019.

L. Zhong, F. Wan, R. Chen, X. Quan, and L. Li. Blockpruner: Fine-grained pruning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

A. Zhou, Y. Ma, J. Zhu, J. Liu, Z. Zhang, K. Yuan, W. Sun, and H. Li. Learning n: M fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations*, 2021.

A Details of Machine Learning Setup

A.1 Calibration Dataset

C4 Raffel et al. [2020] is a large-scale general-purpose English text corpus derived from Common Crawl web data. The dataset is constructed by applying heuristic filtering to remove low-quality content, resulting in approximately 750 GB of clean natural language text. We randomly sampled 128 sequences from the dataset and utilized them as calibration data for T-REX.

A.2 Evaluation Dataset

ARC Clark et al. [2018] is a multiple-choice question-answering benchmark consisting of 7,787 grade-school-level science questions. The dataset is partitioned into two subsets: *Easy*, which contains questions answerable by retrieval-based methods, and *Challenge*, which exclusively contains questions that such methods fail to answer correctly. The Challenge subset places higher demands on a model's multi-step reasoning capabilities.

HellaSwag Zellers et al. [2019] is a commonsense natural language inference benchmark that requires a model to select the most plausible continuation of a given situational description from four candidate options. The dataset is constructed via adversarial filtering, making it particularly challenging for language models despite its apparent simplicity.

PIQA Bisk et al. [2020] targets physical commonsense reasoning. Each example presents a goal and two candidate solutions, and the model must identify the more physically plausible option. The benchmark probes a model's understanding of everyday physical interactions and object affordances.

WinoGrande Sakaguchi et al. [2021] is a large-scale adversarial benchmark based on the Winograd Schema Challenge. Each instance requires resolving the referent of a pronoun in a sentence, demanding a combination of linguistic understanding and commonsense knowledge. The adversarial construction of the dataset minimizes dataset-level artifacts, making it a robust probe of model reasoning.

WikiText-2 Merity et al. [2016] is a large-scale language modeling benchmark built from Wikipedia articles that have been editorially verified for quality. It contains over 100 million tokens and is widely adopted as a standard testbed for evaluating language model perplexity.

A.3 Baseline Implementation Details

All baselines are run using their official code repositories, adapted to support Qwen3 architectures, and evaluated in a zero-shot setting without any post-training or recovery fine-tuning. Where calibration data is required, we use 128 samples from C4, following the same setup as T-REX, except for **BlockPruner**, which uses Alpaca by default. For **LLM-Pruner** we leave the first and last four layers unpruned. **SliceGPT** and **MultiPruner** are applied with default settings; for **MultiPruner** we use the width-to-depth pruning ratio of 44:52:4 as recommended by the authors.

A.4 Post-Training Details

Same-Budget Baseline. To separate the contribution of the T-REX representation from that of additional adaptation, we post-train MultiPruner on Qwen3-8B at 30% compression under the same budget. Post-training helps both: MultiPruner improves from 44.67 to 13.00 PPL and from 49.84% to 59.98% average accuracy, and T-REX from 26.28 to 11.70 PPL and from 54.53% to 63.43%.

B Sensitivity Analysis

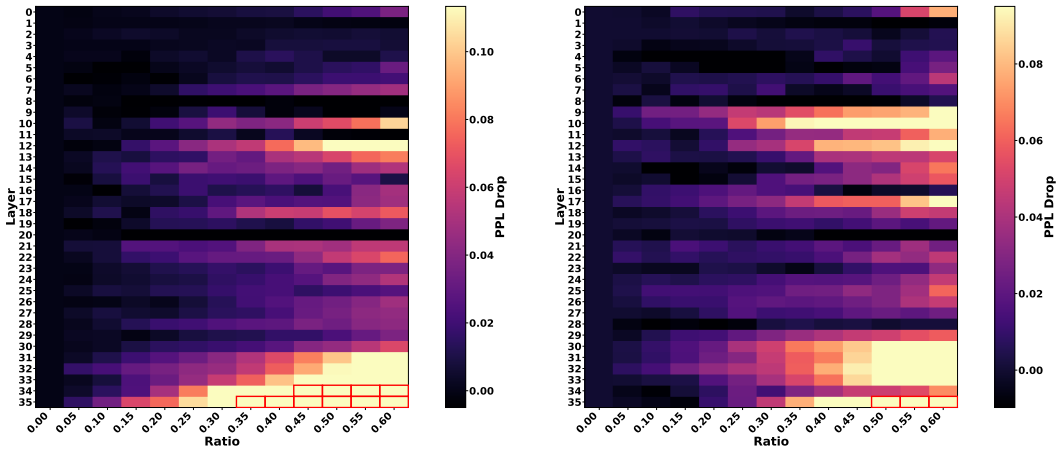
We present a comprehensive per-layer, per-projection sensitivity analysis of T-REX on Qwen3-8B. For each projection type and each layer, we independently apply T-REX at varying compression ratios and measure the resulting change in perplexity (PPL Drop) on WikiText-2. A positive PPL Drop indicates degradation; a negative value indicates a marginal improvement after compression.

Attribute	Configuration
Maximum sequence length	1024
Epochs	1
Training data	C4 subset, ∼41M tokens
Training steps	2500
Batch size	16
Learning rate	2e−4
Warmup ratio	0.03
Weight decay	0.0
Rank / alpha	16 / 32
Dropout	0.05
LR schedule	cosine
Hardware / cost	single A100-80GB; ∼7.5 h (Qwen3-8B), ∼20 h (Qwen3-14B)

Table 5: **Post-Training Attributes**

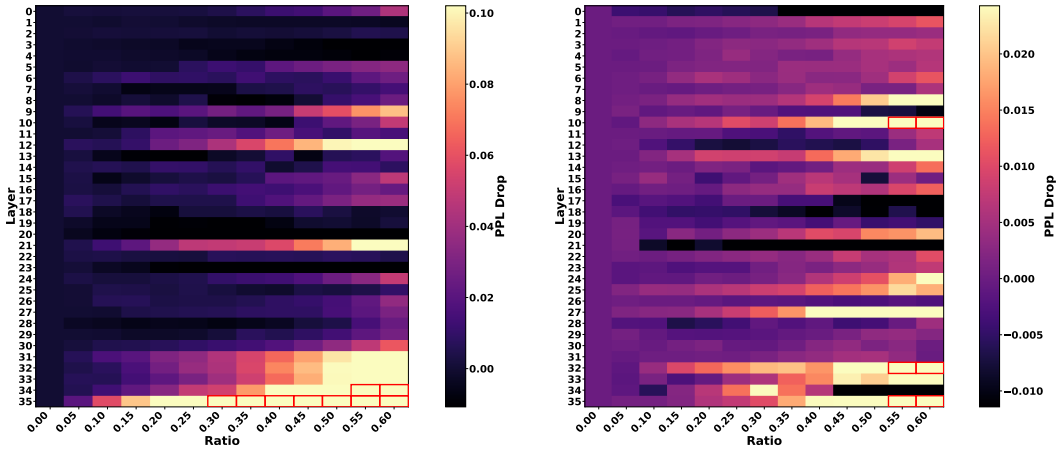
B.1 Per-Projection Sensitivity Across Layers and Ratios

Figures 4–6 present heatmaps of PPL Drop as a function of compression ratio (x-axis) and layer index (y-axis) for each projection type. The up_proj exhibits the highest sensitivity with PPL Drop up to ∼0.12, while down_proj remains comparatively moderate (∼0.10). The gate_proj shows relatively high sensitivity (up to ∼0.12), whereas q_proj remains largely insensitive (0.03). all k_proj, v_proj and o_proj exhibit low overall sensitivity, with several entries showing slightly negative PPL Drop, indicating that moderate compression can marginally improve perplexity in these components.



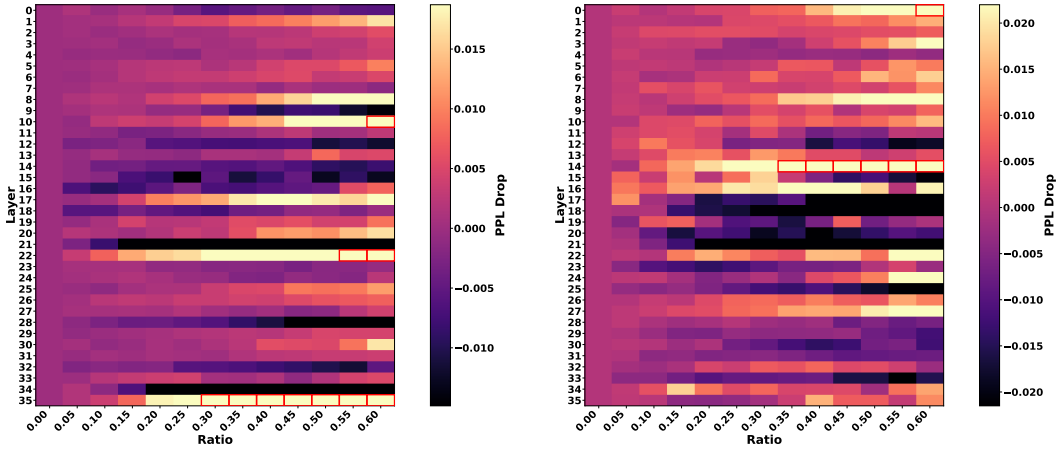
(a) Up Proj (b) Down Proj

Figure 4: Per-layer sensitivity heatmaps for up_proj and down_proj. Red cells indicate PPL Drop values that exceed the colorbar maximum by at least 2×.



(a) Gate Proj (b) Q Proj

Figure 5: Per-layer sensitivity heatmaps for `gate_proj` and `q_proj`. Red cells indicate PPL Drop values that exceed the colorbar maximum by at least $2\times$.



(a) K Proj (b) V Proj

Figure 6: Per-layer sensitivity heatmaps for `k_proj` and `v_proj`. Red cells indicate PPL Drop values that exceed the colorbar maximum by at least $2\times$.

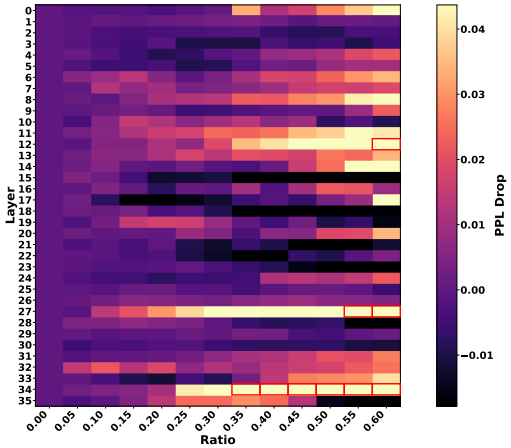


Figure 7: Per-layer sensitivity heatmaps for `o_proj`.

B.2 Sensitivity Along the Depth Dimension

Figure 8 aggregates the sensitivity of all projection types across layers, revealing how compression tolerance evolves with network depth. Early layers (0–5) exhibit the highest sensitivity in `down_proj` and `gate_proj`, while attention projections (`k`, `v`, `q`) remain stable throughout. The final layer (35) shows a sharp increase in PPL Drop across FFN projections, making it the most compression-sensitive region of the model.

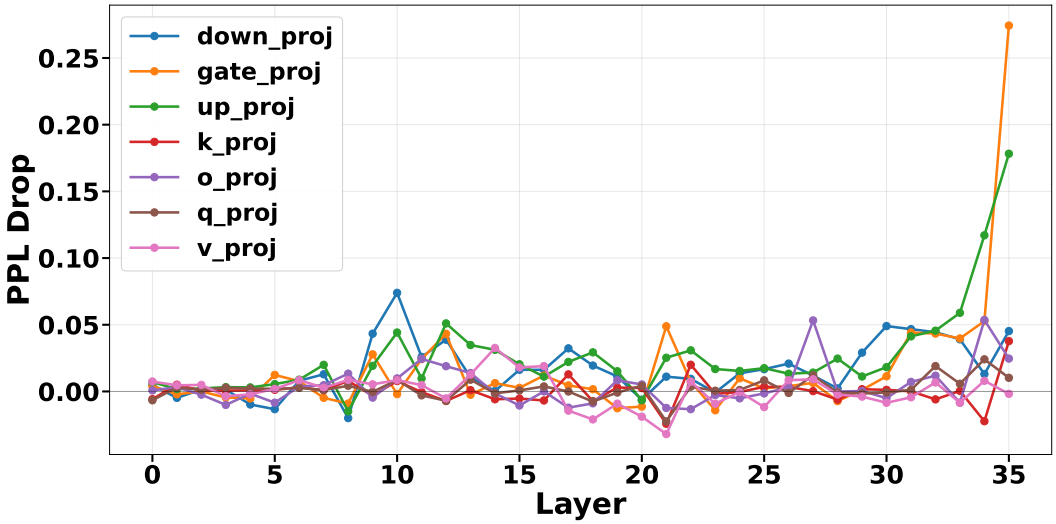


Figure 8: Layer-wise sensitivity of all projection types under T-REX at deletion ratio $r = 0.3$ FFN projections (up, down, gate) are consistently more sensitive than attention projections (q, k, v, o), with the terminal layer showing the sharpest degradation.

B.3 Per-Layer Cross-Projection Comparison

Figures 9–11 compare the sensitivity of all projection types within a single representative layer as a function of compression ratio, providing a fine-grained view of intra-layer compression tolerance. Layer 1 (Figure 9) exhibits near-zero PPL Drop (± 0.017) across all projection types and ratios, confirming that early layers are highly tolerant to compression. Layer 18 (Figure 10) shows moderate sensitivity, with `up_proj` and `gate_proj` slightly elevated at higher ratios. Layer 35 (Figure 11) is the most sensitive, with `up_proj` and `gate_proj` reaching PPL Drop values of up to ~ 1.0 at a compression ratio of 0.6, underscoring the importance of conservative compression budgets for terminal layers.

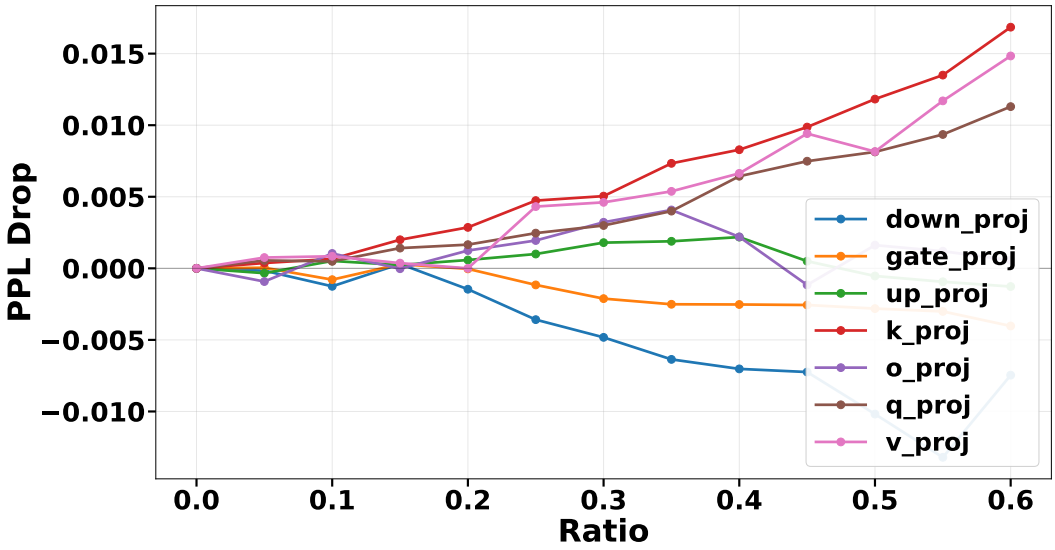


Figure 9: Layer 1: all projection types remain near-zero PPL Drop across all compression ratios.

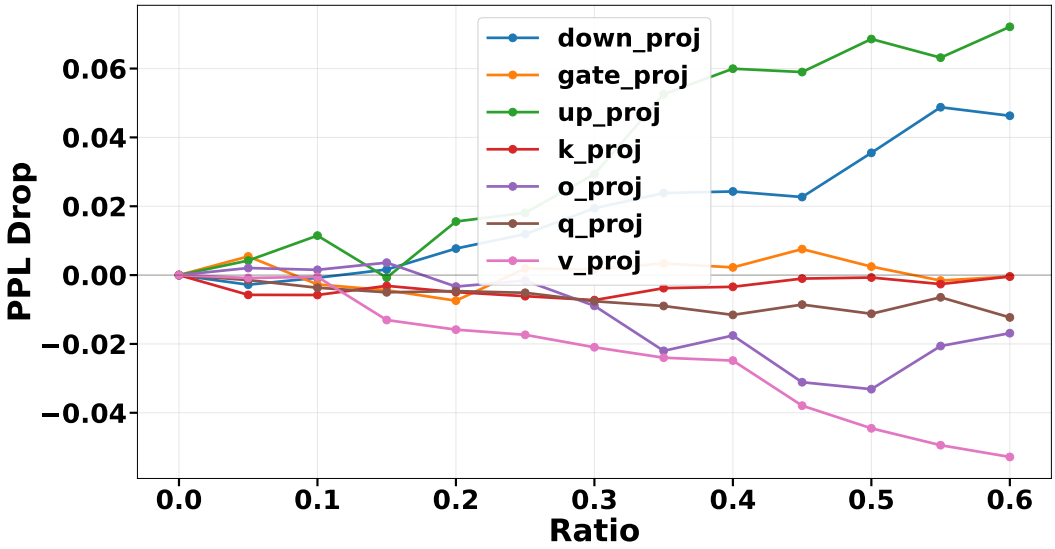


Figure 10: Layer 18: moderate sensitivity, with FFN projections slightly elevated at higher ratios.

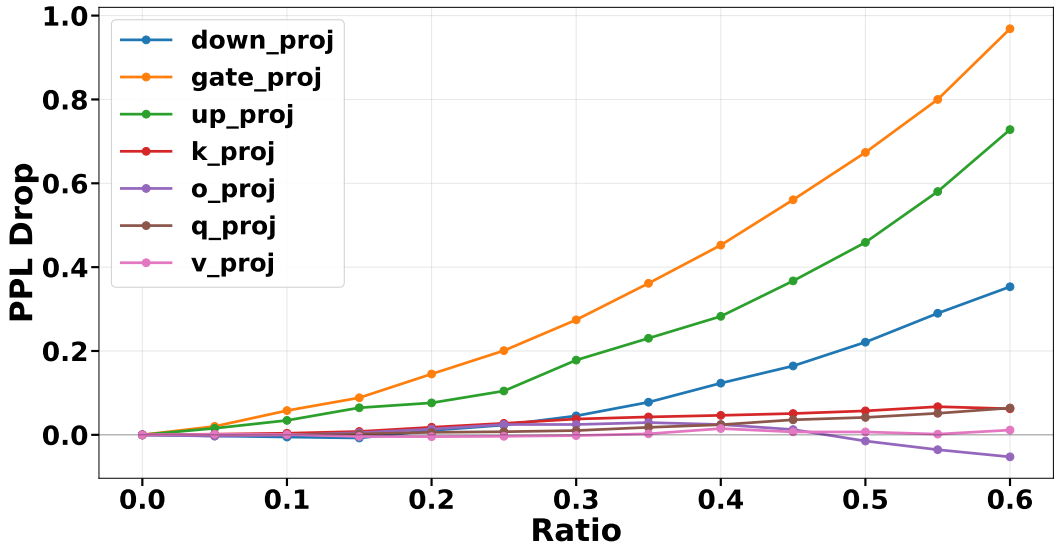


Figure 11: Layer 35: the most sensitive layer, with `up_proj` and `gate_proj` reaching PPL Drop $\sim$1.0 at ratio 0.6.

B.4 Discussion

The sensitivity analysis reveals several consistent patterns across layers and projection types:

- **FFN projections are more sensitive than attention projections.** The `up_proj`, `down_proj`, and `gate_proj` components consistently exhibit higher PPL Drop than `q,k,v,o_proj` at equivalent compression ratios.
- **Terminal layers are disproportionately sensitive.** Layer 35 shows PPL Drop values an order of magnitude larger than those of early or middle layers, suggesting that the final layers encode task-critical representations that are less redundant.
- **Attention key and value projections are compression-friendly.** The `k_proj` and `v_proj` components tolerate aggressive compression with negligible or even slightly negative PPL Drop, making them prime candidates for higher compression budgets.
- **Implications for adaptive compression.** These findings motivate a heterogeneous compression strategy in which the compression ratio is allocated non-uniformly across layers and projection types—applying finer tile granularity and lower deletion ratios to sensitive components (e.g., terminal FFN layers) while permitting coarser compression in robust components (e.g., `k_proj` and `v_proj` in early layers). We leave the design of such an adaptive allocation scheme to future work.

C Other Detailed Results

C.1 Ablation Smaller Model - Qwen3 0.6B

Model	Compression Ratio%	PPL↓	Benchmark ↑					
			ARC-c	ARC-e	HellaS	PIQA	WinoG	Average
Qwen3-0.6B	*0*	20.96	34.04	55.64	47.39	67.36	56.20	52.13
	10	30.00	26.37	47.81	39.53	61.81	53.28	45.76
	20	63.85	23.29	37.33	33.38	58.98	55.01	41.60
	30	647.32	22.35	33.08	29.66	55.28	50.59	38.19

Table 6: **Zero-shot performance under weight deletion (Qwen3-0.6B).**

C.2 Ablation Higher Reuse Ratio

Model	Compression Ratio%	PPL↓	Benchmark ↑					
			ARC-c	ARC-e	HellaS	PIQA	WinoG	Average
Qwen3-8B	40	50.79	28.67	52.86	44.77	66.87	53.43	49.32
	50	406.15	25.94	28.11	31.66	50.82	50.20	37.35

Table 7: **Zero-shot performance**

C.3 Reuse Index Statistics

Table 8 characterises the reuse index. The split between reuse and pruning follows from Algorithm 1: a modified tile is pruned only when the zero tile is selected as its source. The first two columns are shares of the modified tiles; the fan-out of a source tile is the number of targets redirected to it, measured over 28 weight matrices per model (all projection types at four depths). Nearly all of the budget goes to reuse rather than pruning, the freedom that T-REX-P lacks. Reuse is largely one-to-one: over two thirds of sources serve exactly one target, the maximum fan-out is 14, and $\sim$65% of tiles at 20% ($\sim$50% at 30%) are neither redirected nor referenced. The tile pool is thus not a small codebook onto which many tiles collapse: unlike vector quantization, where compression comes from high fan-out onto a few learned representatives, T-REX redirects tiles pairwise between weights already present in the model.

Model	Ratio	Reused (%)	Pruned to zero (%)	Mean fan-out	Never matched (%)
Qwen3-8B	20%	97.81	2.19	1.32	65.3
Qwen3-8B	30%	98.30	1.70	1.48	50.1
Qwen3-14B	20%	96.60	3.40	1.29	65.1
Qwen3-14B	30%	97.14	2.86	1.45	50.0

Table 8: **Reuse index statistics.** Reused and pruned are shares of the modified tiles; never matched are tiles that are neither redirected nor used as a source.

C.4 Generalization to Llama-3-8B

Table 9 compares T-REX with all four baselines on Llama-3-8B at 20% and 30% compression. At 20%, T-REX remains competitive with the structured pruning baselines, with the second-best average accuracy and a perplexity within the range of the baselines. At 30%, the advantage observed on Qwen3 does not fully transfer, and T-REX degrades faster than the strongest baselines. We measure lower inter-tile similarity in Llama-3-8B than in Qwen3, so at higher compression ratios T-REX has to merge less similar tiles, which amplifies the accuracy loss. The tile redundancy that T-REX exploits thus depends on the weight distribution and training dynamics of a model rather than on its layer dimensions.

Method	Ratio	PPL↓	Average↑
Dense	0%	6.14	72.87
LLM-Pruner	20%	**14.66**	55.70
SliceGPT	20%	72.50	57.10
BlockPruner	20%	19.77	57.08
MultiPruner	20%	<u>16.03</u>	**62.10**
T-REX	20%	16.76	<u>57.36</u>
LLM-Pruner	30%	**26.59**	48.30
SliceGPT	30%	140.62	<u>48.41</u>
BlockPruner	30%	<u>40.90</u>	48.13
MultiPruner	30%	45.52	**48.99**
T-REX	30%	42.69	46.52

Table 9: **Llama-3-8B** at 20% and 30% compression. Best result per compression rate in **bold**, second best <u>underlined</u>.

D Compression Ratio: General Expression

We derive the general compression ratio for a single linear layer with weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, tiled with tile size $n \times n$, sub-group size n_g, deletion ratio r, and b bits per weight element.

Tile pool. Under a uniform deletion ratio r, the tile pool $\mathcal{T}$ retains $(1 - r)$ of all tiles. Each tile contains n^2 scalar weights stored at b bits, giving a total tile pool size of

$$|\mathcal{T}|_{\text{bits}} = (1 - r) \cdot d_{\text{out}} \cdot d_{\text{in}} \cdot b. \tag{6}$$

Reuse index. The reuse index π stores one pointer per tile position, mapping each of the $\frac{d_{\text{out}}}{n} \cdot \frac{d_{\text{in}}}{n}$ tile positions to one of at most n_g candidates within its sub-group. Each pointer therefore requires $\lceil \log_2 n_g \rceil$ bits, giving a total index size of

$$|\pi|_{\text{bits}} = \frac{d_{\text{out}}}{n} \cdot \frac{d_{\text{in}}}{n} \cdot \lceil \log_2 n_g \rceil. \tag{7}$$

Compression ratio. Normalising by the original footprint $d_{\text{out}} \cdot d_{\text{in}} \cdot b$, the total compression ratio is

$$\rho = (1 - r) + \frac{\lceil \log_2 n_g \rceil}{n^2 \cdot b}. \tag{8}$$

Specialisation to 16-bit precision. With $b = 16$ and index pointers also stored as 16-bit integers (i.e. $\lceil \log_2 n_g \rceil = 16$), Eq. (8) simplifies to

$$\rho = (1 - r) + \frac{1}{n^2}, \tag{9}$$

which is the expression used in the main text.

Concrete example. Consider compressing a linear layer of Qwen3-8B to a target compression ratio of $\rho = 0.80$ (i.e. 20% memory reduction) with tile size $n = 4$ and sub-group size $n_g = 1024$. The index overhead is fixed at $1/n^2 = 1/16 \approx 6.25\%$. Substituting into Eq. (8), the required deletion ratio is

$$r = 1 - \left(\rho - \frac{1}{n^2} \right) = 1 - (0.80 - 0.0625) = 0.2625, \tag{10}$$

meaning that 26.25% of tiles must be deleted to offset the index overhead and achieve the target footprint of this layer. For the full Qwen3-8B model (8.19B parameters), the embedding and output layers (1.24B parameters) are not tiled, so the linear layers are compressed further to reach a 20% reduction of the whole model: a deletion ratio of $r = 0.2994$ yields 6.12B unique weights (tile pool and untiled layers) and a reuse index of $\sim$0.43B INT16 entries, for a total footprint of $\sim$6.55B FP16-equivalent weights, matching the 20% memory budget of the baselines in Section 4.2.

Choice of tile size. The index is the main cost of smaller tiles: with 16-bit pointers it occupies $1/n^2$ of the original footprint, i.e. 25% at $n = 2$ versus 6.25% at $n = 4$. Since the index counts toward the reported ratio, smaller tiles need a much larger deletion ratio for the same budget (Table 10). At $n = 2$, 45% of tiles must be deleted or redirected for a 20% budget, versus 26.25% at $n = 4$, while the number of tiles, index entries, scores and gathers all grow fourfold. With narrower pointers of $\lceil \log_2 1024 \rceil = 10$ bits instead of 16 (Eq. (8)), the index overhead at $n = 2$ drops from 25% to 15.6%, which materially changes this trade-off.

Tile size n	Index overhead	r for 20%	r for 30%	Tiles to score / gather
2	25.00%	0.4500	0.5500	4×
4	6.25%	0.2625	0.3625	1×
8	1.56%	0.2156	0.3156	0.25×

Table 10: **Cost of the tile size** with 16-bit pointers: index overhead, deletion ratio r required for a 20% and 30% budget, and number of tiles relative to $n = 4$.

E Theoretical Analysis

Layer-level bound. Replacing every deleted tile $j \in \mathcal{D}_m$ by its source changes the output of group m by $\Delta \mathbf{y}_m = \sum_{j \in \mathcal{D}_m} (\mathbf{W}_{\pi(j)} - \mathbf{W}_j)\, \mathbf{x}_j$ (Eq. (1)). By the triangle inequality and Eq. (2), $\|\Delta \mathbf{y}_m\| \le \sum_{j \in \mathcal{D}_m} \|\mathbf{W}_{\pi(j)} - \mathbf{W}_j\|_F \, \|\mathbf{x}_j\|$, and taking the expectation over tokens gives the bound of Section 3.2 with s from Eq. (3).

Tightness. The slack in this bound has two sources, neither of them unbounded. First, the operator norm is replaced by the Frobenius norm; since $\|\mathbf{A}\|_2 \le \|\mathbf{A}\|_F \le \sqrt{n}\, \|\mathbf{A}\|_2$ for an $n \times n$ tile, this costs at most a factor $\sqrt{n}$, i.e. 2 for our 4×4 tiles. Second, $\mathbb{E}\|\mathbf{A}\mathbf{x}_j\|$ is factorized into $\|\mathbf{A}\|_F\, \mathbb{E}\|\mathbf{x}_j\|$, which is exact up to the factor $\mathbb{E}\|\mathbf{A}\hat{\mathbf{x}}_j\|/\|\mathbf{A}\|_F$ with $\hat{\mathbf{x}}_j = \mathbf{x}_j/\|\mathbf{x}_j\|$; under isotropic input directions this factor is close to the constant $1/\sqrt{n}$. Both are near-uniform multiplicative constants within a sub-group. This matters because Algorithm 1 only sorts candidate pairs by s and never uses its magnitude, so it is invariant to any order-preserving rescaling of the score. The relevant notion of tightness is therefore rank fidelity rather than the absolute gap in Eq. (2).

Optimality of the greedy matching. Choosing which tiles of a sub-group survive and routing the others to a surviving tile or to the zero tile is an instance of the k-median problem with an additional virtual zero facility, which is NP-hard. Constant-factor approximation schemes require repeated passes over all candidate pairs, which is intractable for $n_g = 1024$ over the 4.3×10^8 tiles of Qwen3-8B, so we do not claim a constant-factor guarantee. The sub-optimality of the greedy can nevertheless be localized, since it is exactly optimal in two regimes: (i) when the zero tile is the only admissible source, the structured pruning case of T-REX-P, where it reduces to selecting the $\lfloor r|\mathcal{G}| \rfloor$ tiles with the smallest $s(\mathbf{0} \to j)$; and (ii) without the survivorship constraints, where each target independently takes its cheapest source. All remaining slack is thus due to the constraint that a deleted tile can neither serve as a source nor be deleted again. Moreover, the feasible set of T-REX strictly contains that of T-REX-P, so under the same surrogate its optimum is never worse, consistent with Table 2.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the three main contributions: (1) the fungibility scoring function, (2) the T-REX compression algorithm, and (3) empirical comparisons against state-of-the-art baselines. Experimental results in Tables 1**??**23 directly support these claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 explicitly discusses limitations, including the use of a uniform global tile size, the lack of adaptive/heterogeneous compression across layers, the inference overhead of tile reconstruction, and task-dependent compression profiles.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper's theoretical component is the fungibility score derivation 534, which relies on the submultiplicativity of the Frobenius norm. The bound is clearly stated as a tractable upper bound rather than a tight guarantee. No formal theorems requiring full proofs are claimed.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4.2 details the experimental setup including model names, calibration data, hardware, evaluation benchmarks, tile size, and sub-group size. Post-training hyperparameters are listed in A. Source code is made publicly available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The source code and evaluation pipeline are publicly released. All datasets used (C4, WikiText-2, ARC, HellaSwag, PIQA, WinoGrande) are publicly available benchmarks with established access procedures.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and evaluation details are provided in Section 4 and Appendix A. This includes optimizer settings, LoRA rank/alpha, learning rate schedule, sequence length, number of epochs, and hardware specifications. Baseline methods are applied under identical compression budgets as described in Section 4.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: The paper reports point estimates without error bars or confidence intervals. The evaluation benchmarks (ARC, HellaSwag, PIQA, WinoGrande) are deterministic given fixed model weights, and compression index construction is also deterministic.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments are conducted on a single NVIDIA A100 (80GB) GPU.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work presents a model compression method for large language models. The research does not involve human subjects, sensitive data collection, or applications with direct potential for harm. The authors have reviewed the NeurIPS Code of Ethics and confirm compliance.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: T-REX reduces the memory footprint of large language models, enabling deployment on resource-constrained hardware and lowering the energy cost of inference. This democratizes access to capable models. The method itself does not introduce new capabilities or risks beyond those of the base models it compresses; any societal risks are inherited from the underlying LLMs rather than introduced by the compression technique.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release new pretrained models or datasets. T-REX is a compression algorithm applied to existing publicly available models (Qwen3). No new high-risk assets are introduced.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and models used are properly cited. Qwen3 models are used under their respective open-source licenses. Evaluation datasets (ARC, HellaSwag, PIQA, WinoGrande, WikiText-2, C4) are publicly available research benchmarks. The LM Evaluation Harness is cited [Gao et al., 2024]. CUTLASS is cited [Thakkar et al., 2023].

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper releases the T-REX source code and evaluation pipeline at `https://github.com/haoyi-yang/TREX`. Documentation is provided in the repository. No new datasets or model checkpoints are released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or human subjects. All experiments are conducted on publicly available datasets and pretrained models.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve human subjects research and therefore does not require IRB approval or equivalent review.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs were used solely for writing assistance and editing. They do not constitute any part of the core method, experimental design, or scientific contributions of this work.

    Guidelines:

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