# LOSSY PROJECTION, EXACT EXPANSION: RETROFITTING PRETRAINED GATED FEED-FORWARD BLOCKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Weight movement can limit autoregressive language-model inference, motivating gated feed-forward architectures that reuse weights across computational paths. Masked Gated Linear Units (MGLUs) construct gate and value projections using complementary masks over a shared matrix, with multiple masks providing additional routes without duplicating that matrix. Converting pretrained blocks into this representation can incur branch-reconstruction error, while initializing additional routes can further alter the projected function. We propose Projection-Aware Route Expansion (PARE), which separates dense-to-shared projection from function-preserving multi-mask expansion. Our analysis characterizes the reconstruction limits of complementary sharing and establishes an exact embedding of the projected single-mask parent through repeated masks and a rescaled down projection. Mask logits are initialized with matching signs but distinct magnitudes across routes, yielding identical binary masks at entry while permitting differentiation during recovery. In the controlled 159M experiments, four-mask PARE reaches 15.65 perplexity versus 16.06 for fixed single-mask recovery at the same recovery-token budget. Within the same four-mask architecture, PARE also achieves lower endpoint perplexity than quota-preserving and reconstruction-calibrated diverse initializations. These endpoint advantages persist when the frozen checkpoints are evaluated on source-excluded FineWeb-Edu and C4. Further experiments on TinyLlama-1.1B and Gemma-3-1B, PARE also achieves lower endpoint perplexity than fixed single-mask recovery. A matched four-mask comparison on TinyLlama additionally favors PARE over diverse initialization. These results support function-preserving route entry for improving conversion fidelity at fixed token budgets.

## 1 INTRODUCTION

Autoregressive language models face different inference bottlenecks. Prompt processing is often compute intensive, whereas low-batch token generation can be limited by weight movement through memory (Gholami et al., 2024; Patel et al., 2024). Gated feed-forward blocks amplify this pressure because SwiGLU and GeGLU apply separate dense gate and value projections before the down projection (Shazeer, 2020; Chowdhery et al., 2023). The Masked Gated Linear Unit (MGLU) addresses this duplication with complementary binary masks over one shared matrix $W$ (Tajima et al., 2025). Its multi-mask variant, Mixture of Element-wise Gating (MoEG), applies $n_m$ gate/value assignments to $W$, adding gated transformations without duplicating the matrix. Using MGLU with pretrained models requires transferring learned branch computations into shared weights under a finite recovery budget.

Pretrained networks can be expanded through function-preserving constructions (Chen et al., 2016; Wei et al., 2016) or approximate and learned Transformer mappings (Chen et al., 2022; Wang et al., 2023). Dense-to-mixture-of-experts (MoE) conversion and sparse upcycling partition or copy feed-forward weights for token-routed experts (Zhang et al., 2022; Komatsuzaki et al., 2023; Zhu et al., 2024). MGLU conversion additionally merges two learned branch matrices into complementary views of one shared matrix, discarding one source role per coordinate. Introducing diverse masks



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

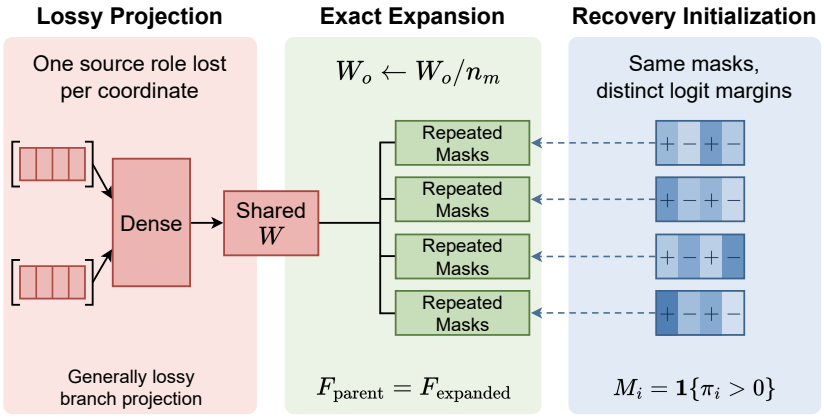


Figure 1: PARE's conversion path and recovery initialization. Left: dense gate and value maps generally overlap, so complementary ownership discards one branch coefficient per coordinate. Middle: repeating the parent mask across $n_m$ routes and scaling $W_o$ by $1/n_m$ preserves the projected single-mask function exactly. Right: mask logits have identical signs but distinct magnitudes, giving identical initial masks that can differentiate during recovery. $W$ holds the shared gate/value weights, while $W_o$ maps the summed route outputs back to the model dimension. Weight and logit strips illustrate the same four coordinates. Blue intensity encodes logit magnitudes, and dashed arrows indicate the logit-to-mask mapping.

can then alter this projected function again. Prior MGLU results mainly concern training from scratch, leaving these pretrained conversion effects unresolved.

We study the conversion of pretrained gated feed-forward blocks into complementary shared-weight MGLUs, using SwiGLU as the primary controlled setting. The target retains one shared input matrix and input-independent binary assignments, with adaptation under a fixed recovery-token budget. The conversion contains two distinct transitions. Dense-to-single-mask conversion projects two source branch matrices into complementary views of one matrix and is generically lossy when their supports overlap. In contrast, single-to-multi-mask expansion can preserve the projected parent function exactly. This distinction asks a sharper question than whether multiple masks increase capacity: can the target class expand without a second function discontinuity?

A single-mask comparison alone cannot attribute recovery gains to the expansion initialization. Comparing alternative entries within the same multi-route architecture separates the entry-path effect from the complete expansion procedure.

Projection-Aware Route Expansion (PARE) constructs a shared single-mask parent by coefficient copying, then repeats its mask and scales the down projection by $1/n_m$ to preserve the projected function (Figure 1). Independent mask-logit margins allow assignments to diverge during recovery. At 159M, PARE lowers endpoint perplexity (PPL) relative to fixed-mask recovery and the tested non-preserving entries at matched tokens. Their ordering persists on source-excluded FineWeb-Edu and Colossal Clean Crawled Corpus (C4) evaluations (Penedo et al., 2024; Raffel et al., 2020), and the TinyLlama comparison extends the entry-path result to a public checkpoint. With projection–epilogue fusion, recovered TinyLlama feed-forward networks (FFNs) run 1.21–1.22× faster than the dense source on an RTX 4090 under single-token rotation through 22 independent FFNs.

Our contributions are:

- **Branch-sharing analysis.** We formulate complementary parameter sharing as constrained branch reconstruction, characterize exact preservation of the source branch maps, and compare the reconstruction limits of nested assignment classes.

- **Projection-aware route expansion.** PARE exactly embeds a single-mask parent into multi-mask MGLU and uses untied per-mask latent parameters during recovery.

- **Controlled conversion evidence.** At 159M, PARE achieves lower endpoint perplexity than quota-preserving and reconstruction-calibrated diverse entries within the same four-route architecture and recovery-token budget. This ordering persists on source-excluded FineWeb-Edu and C4.
- **Execution of recovered blocks.** We implement projection–epilogue fusion for shared multi-mask FFNs and evaluate a separate CUTLASS-backed 2:4 structured-sparse path, identifying complete-FFN trade-offs across workloads.

## 2 Related Work

**Function-preserving model growth.** Net2Net and network morphism transformed trained networks while preserving their input–output function (Chen et al., 2016; Wei et al., 2016). Transformer variants used exact, approximate, or learned width/depth mappings (Chen et al., 2022; Wang et al., 2023; Gesmundo & Maile, 2023), including identity-initialized post-pretraining expansion (Wu et al., 2024). Width-progressive learning also studied the tension between signal preservation and symmetry breaking during expansion (Yu et al., 2026). Together, this literature motivates preserving a pretrained function while exposing additional capacity. We study the corresponding expansion step for repeated MGLU masks after a lossy dense-to-shared projection.

**Dense-to-MoE conversion and checkpoint upcycling.** Dense-to-MoE methods constructed token-routed experts from pretrained FFNs (Zhang et al., 2022; Zhu et al., 2024; Cai et al., 2024; Szatkowski et al., 2024; Zhao et al., 2026). Generalized MoEfication extended neuron-to-expert conversion beyond rectified linear unit (ReLU) FFNs (Lee et al., 2024). Other approaches derived experts from pretrained FFNs through pruning and reassembly (Feng et al., 2025) or learned activation masks (Liu et al., 2025b). Sparse upcycling instead copied dense FFNs (Komatsuzaki et al., 2023), with Drop-Upcycling partially reinitializing expert weights before specialization (Nakamura et al., 2025). Expert upcycling expanded an already sparse MoE by duplicating experts and extending its router (Dwivedi et al., 2026). MLPMoE gave a training-free, static reinterpretation of dense multilayer perceptrons (MLPs) through tensor slicing and summation (Novikov, 2025). Unlike PARE, it did not merge pretrained gate and value matrices through complementary ownership. Raw-sum multi-mask MGLU instead evaluates every route, shares one $W$, and varies input-independent gate/value decompositions rather than expert-specific weight matrices. The assignments may change during recovery but are fixed across tokens at inference.

**Constrained sharing, sparsity, and architectural uptraining.** One-shot and structured methods reconstructed, scored, or removed pretrained weights and subspaces (Frantar & Alistarh, 2022; 2023; Sun et al., 2024; Dong et al., 2024; Ma et al., 2023; An et al., 2024; Xia et al., 2024; Ashkboos et al., 2024; Wei et al., 2024). Strict $N{:}M$ work learned masks or expanded support through permutations (Zhou et al., 2021; Fang et al., 2024; Sun et al., 2026; Zou et al., 2025), while NVIDIA Sparse Tensor Cores provide native acceleration for the 2:4 pattern (Mishra et al., 2021). DaSS retained separate gated linear unit (GLU) projections (Guo et al., 2025). MGLU merged them and was studied primarily from scratch (Tajima et al., 2025), including alternative gating functions and an optional token-wise Top-$K$ router. Grouped-query attention provides an attention-side example of checkpoint uptraining into a new sharing structure (Ainslie et al., 2023), while contextual sparsity instead changed execution by input (Liu et al., 2023; 2025a).

## 3 Problem Formulation and Branch-Sharing Constraints

### 3.1 Dense and Shared Gated Feed-Forward Blocks

Consider a pretrained gated feed-forward block with model width $d$, FFN intermediate width $h$, input $x \in \mathbb{R}^{1\times d}$, gate and value projections $W_g, W_v \in \mathbb{R}^{h\times d}$, and down projection $W_o \in \mathbb{R}^{h\times d}$. For an elementwise gate $\phi$, the block computes

$$F^{(\phi)}_{\text{dense}}(x) = \left[\phi(xW_g^\top) \odot xW_v^\top\right] W_o.$$

Following MGLU notation, $n_m$ denotes the number of mixture components (equivalently, masks), and $M_i$ is the $i$th binary mask. The sigmoid linear unit (SiLU) and Gaussian error linear unit (GELU) gates yield SwiGLU and GeGLU, respectively. We use MGLU for the gate-agnostic shared family

and SwiMGLU and GeMGLU for its corresponding gated instances. MGLU replaces $(W_g, W_v)$ with a shared $W \in \mathbb{R}^{h\times d}$. For a single mask $M = [m_{kj}] \in \{0,1\}^{h\times d}$, $m_{kj} = 1$ assigns coordinate $(k, j)$ to the gate branch. Its complement is defined elementwise by $(1 - M)_{kj} = 1 - m_{kj}$. The hidden output of one route is

$$q^{(\phi)}(M, W; x) = \phi\big(x(M \odot W)^\top\big) \odot x((1 - M) \odot W)^\top \in \mathbb{R}^{1\times h}. \tag{1}$$

Using the route output in Equation (1), the single-mask and raw-sum $n_m$-mask block outputs are

$$F_1^{(\phi)}(x; W, M, W_o) = q^{(\phi)}(M, W; x)W_o,$$

$$F_{n_m}^{(\phi)}(x; W, M_1, \ldots, M_{n_m}, W_o^{(n_m)}) = \left[\sum_{i=1}^{n_m} q^{(\phi)}(M_i, W; x)\right] W_o^{(n_m)}. \tag{2}$$

All routes execute for every input and share $W$, but may use different binary masks.

### 3.2 Constrained Branch Projection

The first conversion stage chooses $(M, W)$ to approximate the pretrained gate and value maps through complementary coefficient ownership. The source branch preactivations are $g = xW_g^\top$ and $v = xW_v^\top$. The deployed branch maps and their coefficient errors are

$$\widetilde{g} = x(M\odot W)^\top, \quad \widetilde{v} = x((1-M)\odot W)^\top, \quad E_g = W_g - M\odot W, \quad E_v = W_v - (1-M)\odot W. \tag{3}$$

We measure the best branch-preactivation reconstruction available to a fixed assignment by

$$\begin{aligned}\mathcal{L}(M, W) &= \mathbb{E}[(g - \widetilde{g})\Gamma_g(g - \widetilde{g})^\top] + \mathbb{E}[(v - \widetilde{v})\Gamma_v(v - \widetilde{v})^\top] \\ &= \mathrm{Tr}(\Gamma_g E_g \Sigma E_g^\top) + \mathrm{Tr}(\Gamma_v E_v \Sigma E_v^\top), \\ R(M) &= \min_W \mathcal{L}(M, W),\end{aligned} \tag{4}$$

where $\Sigma = \mathbb{E}[x^\top x]$ is the uncentered input second moment, and the positive diagonal matrices $\Gamma_g = \mathrm{diag}(\alpha_1, \ldots, \alpha_h)$ and $\Gamma_v = \mathrm{diag}(\beta_1, \ldots, \beta_h)$ weight the gate and value reconstruction errors, respectively. For a fixed assignment $M$, $R(M)$ is the minimum branch-reconstruction error over all shared weights $W$.

The next result concerns a strict $N{:}B$ mask, which assigns exactly $N$ gate entries in each aligned block of $B$ input coordinates.

**Proposition 1** (Exact branch preservation)**.** *Under $\Sigma \succ 0$, a branch-map-preserving strict mask and shared matrix exist if and only if*

$$S_{kb}^g \cap S_{kb}^v = \varnothing, \qquad |S_{kb}^g| \le N \le B - |S_{kb}^v|, \tag{5}$$

*for every $(k, b)$, where $S_{kb}^g$ and $S_{kb}^v$ are the nonzero supports of $W_g$ and $W_v$ in row $k$ and aligned block $b$. Necessity follows because the $N$ gate positions must contain $S_{kb}^g$ and exclude $S_{kb}^v$; sufficiency follows by filling the remaining gate quota with coordinates in neither support. Dense overlapping gate/value rows violate the first condition, so exact branch preservation is generically impossible at conversion time.*

Exact branch preservation requires disjoint source supports for any complementary assignment, while local quotas impose the additional cardinality conditions. Relaxing these quotas enlarges the reconstruction class but does not remove overlapping-support conflicts. $\mathcal{M}_{N:B}$ denotes the strict assignment class, whereas $\mathcal{M}_{\mathrm{bal},50\%}$ denotes layerwise balanced assignments: exactly half of the coordinates in each layer's mask matrix are assigned to the gate branch, with the remainder assigned to the value branch.

**Proposition 2** (Aligned nesting and projection-floor monotonicity)**.** *For partitions with a common origin, hierarchically aligned boundaries, and $8 \mid d$,*

$$\mathcal{M}_{1:2} \subseteq \mathcal{M}_{2:4} \subseteq \mathcal{M}_{4:8} \subseteq \mathcal{M}_{\mathrm{bal},50\%}. \tag{6}$$

*Writing $P(\mathcal{C}) = \min_{M\in\mathcal{C}} R(M)$ therefore gives a nonincreasing sequence of projection floors as the class expands.*

The class floor and selector error are distinct: enlarging the feasible class cannot increase $P(\mathcal{C})$, but a particular surrogate selector can still have non-monotone regret. Although instantiated by MGLU, the analysis applies whenever two maps share coefficients through complementary ownership. These results characterize the limits of the assignment classes. The initialization used in our experiments is specified in Section 4.2. Appendix A provides the detailed derivations and structural results.

# 4 PROJECTION-AWARE ROUTE EXPANSION

## 4.1 EXACT REPEATED-ROUTE EMBEDDING

The projection results characterize the branch-map reconstruction error incurred when constructing a single-mask parent. A separate question is whether that parent function can enter the raw-sum multi-mask MGLU model in Equation (2) without an additional function discrepancy.

For fixed $\phi$, the block-function classes realizable by the single-mask and $n_m$-mask parameterizations in Equation (2) are denoted by $\mathcal{F}_1^{(\phi)}$ and $\mathcal{F}_{n_m}^{(\phi)}$, respectively.

**Proposition 3** (Exact repeated-route embedding). *Assume $n_m \geq 1$, repeated masks are admissible, all routes share $W$, aggregation is the raw sum in Equation (2), and $W_o/n_m$ is an admissible $n_m$-mask down projection. Then, for every input,*

$$F_{n_m}^{(\phi)}(x; W, M, \ldots, M, W_o/n_m) = F_1^{(\phi)}(x; W, M, W_o). \tag{7}$$

*Consequently, for otherwise identical surrounding networks, $\mathcal{F}_1^{(\phi)} \subseteq \mathcal{F}_{n_m}^{(\phi)}$.*

The identity holds for any fixed elementwise gate $\phi$, including ReLU, GELU, and SiLU, without requiring continuity or differentiability. It preserves the projected parent function at expansion but does not remove the preceding dense-to-shared projection error. Whether this initialization improves finite-budget recovery is evaluated in Section 5.3. Appendix B provides the function-class risk corollary and symmetry analysis.

## 4.2 CONSTRUCTING AND TRAINING THE EXPANDED MODEL

**Single-mask parent.** For a chosen assignment $M$, we initialize the shared matrix by exact coefficient copying:

$$W_0(M) = M \odot W_g + (1 - M) \odot W_v. \tag{8}$$

This coefficient-copying rule supplies the practical initializer, while $R(M)$ in Equation (4) remains an analytical reference. When $\Sigma$ is diagonal, $W_0(M)$ also attains the fixed-mask optimum. Magnitude scores each coordinate by $|w_{kj}^g| - |w_{kj}^v|$. Activation-aware scoring instead uses the difference between squared gate and value weights, weighted by input and gate-sensitivity second moments estimated from calibration data. Both selectors assign the highest-scoring coordinates to the gate branch: $N$ per aligned block for strict $N{:}B$, or exactly half over each layer's full assignment matrix for layerwise balanced. The primary parent uses activation-aware layerwise balanced assignment, and the primary single-mask runs keep its mask fixed. The activation-aware formula and calibration estimators are in Appendix C.

**Projection-aware route expansion.** We first project the dense checkpoint into a layerwise balanced single-mask parent using Equation (8), then repeat its binary mask across a chosen number $n_m$ of routes and apply $W_o \leftarrow W_o/n_m$, instantiating Proposition 3. PARE combines these two stages. In the primary runs, expansion follows projection immediately, without intervening parent recovery. Fixed-mask recovery continues training this single-mask model while keeping its assignment fixed. Each route $i \in \{1, \ldots, n_m\}$ receives an independent mask-logit tensor $\pi_i \in \mathbb{R}^{h \times d}$ initialized with signed random margins:

$$\pi_i = (2M - 1) \odot |\epsilon_i|, \qquad M_i = \mathbf{1}\{\pi_i > 0\}, \tag{9}$$

where the entries are sampled independently across routes and coordinates:

$$(\epsilon_i)_{kj} \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2), \tag{10}$$

where $\sigma > 0$ is the mask-logit initialization scale. Absolute values and thresholding are elementwise. Thus the binary masks are identical across routes and the network function agrees with the projected parent at entry, while the mask-logit state is not exactly symmetric. An unconstrained elementwise straight-through estimator (STE) (Bengio et al., 2013) can subsequently update the untied mask logits. The layerwise 50% gate quota applies only at initialization: recovery retains complementarity without enforcing a quota. Recovery uses the binary masks in the forward pass and the identity straight-through derivative. The resulting masks vary across recovery steps but are input-independent and fixed at deployment. The weight matrix $W$ remains shared across routes, and all Transformer weights and the mask logits are optimized jointly. Algorithm 1 outlines PARE.

**Algorithm 1** Projection-Aware Route Expansion (PARE)

**Require:** Pretrained gated model; parent assignment $M$ for each converted block; mask count $n_m$; initialization scale $\sigma$; recovery data and number of recovery steps $T$
**Ensure:** Recovered shared-weight model with $n_m$ binary masks per converted block

**for** each converted block $(W_g, W_v, W_o)$ **do**
  $W \leftarrow M \odot W_g + (1 - M) \odot W_v$ ▷ Lossy branch projection
  $W_o \leftarrow W_o / n_m$ ▷ Exact expansion of the projected parent
  **for** $i = 1, \ldots, n_m$ **do**
    Sample $\epsilon_i$ according to Equation (10)
    Initialize $\pi_i$ and $M_i$ using Equation (9)
  **end for**
  Replace the dense block by raw-sum multi-mask MGLU with shared $W$ and scaled $W_o$
**end for**
Retain all other pretrained weights and initialize fresh optimizer state
**for** $t = 1, \ldots, T$ **do**
  Sample a minibatch from the recovery data
  Form hard masks $M_i = \mathbf{1}\{\pi_i > 0\}$ in every converted block
  Evaluate the language-model loss using raw-sum multi-mask MGLU forward passes
  Backpropagate with the identity STE for the mask thresholds
  Update all Transformer weights and the independent mask logits
**end for**
Freeze the final binary masks for deployment

# 5 Experiments

## 5.1 Experimental Setup

**Models, data, and budgets.** Unless noted, experiments use FineWeb-Edu `sample-10BT` (Penedo et al., 2024), the TinyLlama tokenizer (Zhang et al., 2024), length 1,024, global batch 512, bfloat16 (BF16), and AdamW (Loshchilov & Hutter, 2019). Each recovery step processes 524,288 tokens. The fixed 524,800-token buffer drawn from the FineWeb-Edu training stream was repeatedly consulted during method selection and is treated as a development buffer. Calibration and recovery streams are document-separated from it. After fixing the methods and checkpoints, we evaluate the same models on a source-corpus-excluded FineWeb-Edu shard and on C4.

We retain the reference "159M" configuration label (Tajima et al., 2025), although its exact dense count with our tokenizer and tied embeddings is 140.987M. The dense source was trained for 10B tokens and has development-buffer PPL 14.8047. Conversion uses a cosine schedule from $3 \times 10^{-5}$ to $3 \times 10^{-6}$ with 20 warmup steps. The 200, 2k, and 19,074-step budgets contain 104.86M, 1.049B, and 10B continuation tokens. Unless stated otherwise, $x \pm y$ denotes the mean and sample standard deviation across run seeds, with paired quantities formed within seed before aggregation. Model dimensions, checkpoint sources, data roles, and exact parameter counts are provided in Appendix D.1.

**Optimization and evaluation.** Unless otherwise stated, PARE uses $n_m = 4$, following the quality–resource trade-off identified in the original MGLU experiments, which report diminishing returns from additional masks (Tajima et al., 2025, Section 5.3 and Appendix B.4). Mask-logit initialization uses $\sigma = 0.01$ in Equation (10). The primary 159M comparisons match the source checkpoint, recovery-token order, development buffer, optimizer family, and 2k schedule. All primary conditions update the same non-mask Transformer weights. The $n_m = 4$ entry-path conditions additionally share the same trainable mask-logit parameterization, while fixed-mask recovery retains the parent's single mask. All conditions start with fresh optimizer and scheduler state, and expansion follows projection immediately. We present negative log-likelihood (NLL) and PPL endpoints, paired NLL trajectories on common grids, entry logit/NLL equality, binary-mask differentiation, measured optimizer-step time, and the area under each NLL recovery curve (NLL-AUC). Lower NLL, PPL, and NLL-AUC indicate better performance. Appendix D provides full hyperparameters, seed construction, numerical and metric conventions, parameter storage, and recovery timing. On the same RTX 4090, median step time is 11.63 s for fixed-mask recovery and 29.86 s for PARE, excluding evaluation and checkpointing. Projection-kernel and complete-FFN measurements are presented in Appendix G.

### 5.2 Single-Mask Assignment Frontier

We first compare the assignment classes used to construct single-mask parents, before evaluating expansion from the layerwise balanced parent. The feasible assignment classes satisfy $\mathcal{M}_{1:2} \subseteq \mathcal{M}_{2:4} \subseteq \mathcal{M}_{4:8} \subseteq \mathcal{M}_{\mathrm{bal},50\%}$, so the branch-reconstruction surrogate floor cannot increase as freedom grows. Separately, at 159M both magnitude and activation-aware assignment yield the observed PPL order 1:2 $>$ 2:4 $>$ 4:8 $>$ layerwise balanced at both step zero and 2k (Figure 2). At 2k, activation-aware improves magnitude by 0.0502 PPL for 1:2, 0.0382 for 2:4, and 0.0226 for 4:8, while the two methods are effectively tied for layerwise balanced assignment. By 2k, these within-class differences have contracted and are substantially smaller than the gaps between assignment classes. We therefore use the layerwise balanced parent for the primary expansion comparisons and retain strict 2:4 for separate execution studies. Appendix F provides the full trajectories.

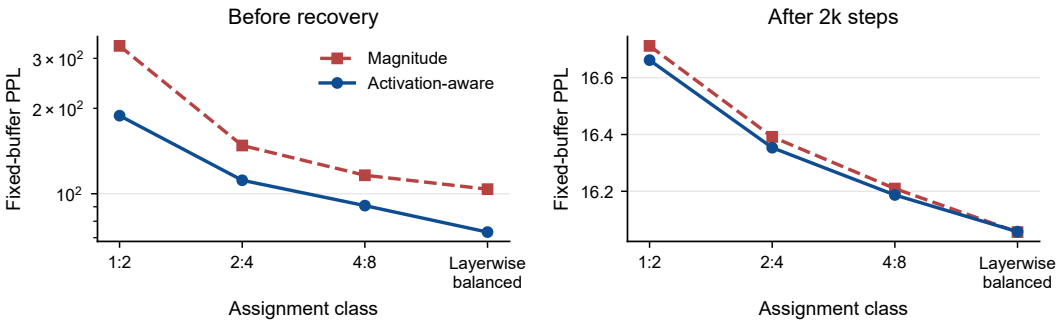


Figure 2: First-stage structure–fidelity frontier at 159M. Step-zero and 2k PPL improve monotonically across the tested nested assignment classes for both initializers (one seed per configuration).

### 5.3 Recovery with Projection-Aware Route Expansion

Table 1 compares PARE with fixed single-mask recovery and with alternative initializations of the same four-route model. The quota-preserving diverse entry retains one parent mask and perturbs the other three by exchanging 10% of gate coordinates with an equal number of value coordinates in each layer. Its reconstruction-calibrated variant additionally fits shared weights to the projected parent's outputs before recovery.

**Expansion versus fixed-mask recovery.** For both strict 2:4 and layerwise balanced parents, PARE's step-zero full-model 32-bit floating-point (FP32) logits and BF16 fixed-buffer NLL agree exactly with those of the projected single-mask parent. At matched tokens, PARE from the layerwise balanced parent reaches $15.6511 \pm 0.0024$ PPL versus $16.0559 \pm 0.0023$ for fixed-mask recovery. Four-mask PARE also achieves a modest improvement in average zero-shot performance over fixed-mask recovery across six downstream tasks (Appendix E.6).

Table 1: Primary 159M development-buffer results. PARE is rerun for each comparison under the corresponding matched protocol. Comparisons are paired within seed against the PARE reference in the same group. All conditions except fixed-mask recovery use $n_m = 4$. PARE and quota-preserving diverse entries ran as same-GPU RTX 3090 pairs, while the seed-aligned reconstruction-calibrated diverse condition ran on RTX 4090.

| Comparison | Condition | Entry NLL | PPL@2k |
|---|---|---|---|
| Parent comparison | Fixed single-mask | 4.2940 | $16.0559 \pm 0.0023$ |
| | PARE | 4.2940 | $\mathbf{15.6511 \pm 0.0024}$ |
| Entry-path comparison | PARE | 4.2940 | $\mathbf{15.6513 \pm 0.0004}$ |
| | Quota-preserving diverse entry | $5.1549 \pm 0.0726$ | $15.8627 \pm 0.0042$ |
| | Reconstruction-calibrated diverse entry | $4.3601 \pm 0.0101$ | $15.8887 \pm 0.0036$ |

**Entry-path comparison.** To test whether preserving the parent function improves recovery, we compare PARE with the quota-preserving diverse entry. Both use layerwise balanced initialization and unconstrained elementwise mask recovery, with the same shared $W$, absolute mask-logit margins, optimizer, data, and recovery tokens. PARE reaches $15.6513 \pm 0.0004$ PPL versus $15.8627 \pm 0.0042$ for the diverse initialization, with lower PPL in every paired seed. The diverse initialization still outperforms fixed-mask recovery. Even within the same four-route architecture and recovery budget, endpoint quality depends on how expansion is initialized.

These ownership exchanges can also make source-aligned coefficients serve the opposite branch. To test this alternative explanation, a reconstruction control freezes the diverse binary masks, mask logits, and surrounding Transformer, and reconstructs only shared $W$ and $W_o$ against projected-parent layer outputs on a separate calibration stream. It discards the calibration optimizer state before the same 2k recovery schedule and therefore receives extra computation outside the matched recovery budget.

Reconstruction removes 91.05–94.14% of the initial NLL excess and reaches $15.8887 \pm 0.0036$ PPL. PARE achieves lower PPL by $0.2374 \pm 0.0039$, with lower full and post-200 NLL-AUC in every seed. The endpoint gap persists after calibration substantially reduces entry NLL excess and partially reduces held-out FFN-output error. Appendices D.6 and E.2 provide the reconstruction protocol and per-seed results, respectively. Figure 3 summarizes the paired recovery trajectories and matched entry-path gaps on their respective common grids.

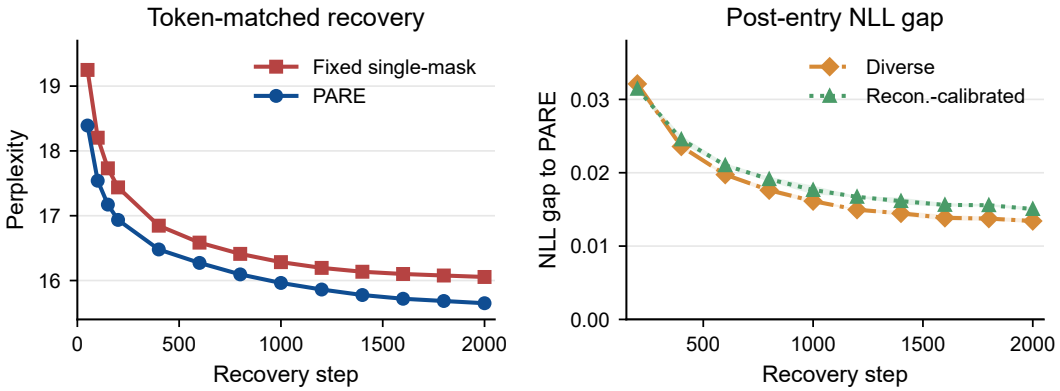


Figure 3: Recovery after projection. Left: post-entry PPL on the common trajectory grid for two paired seeds, with the step-zero point omitted. PARE remains below fixed-mask recovery. Right: quota-preserving and reconstruction-calibrated diverse entries retain positive NLL gaps to PARE through 2k across three seeds. Lines and shading show means and one sample standard deviation. Endpoint statistics in Table 1 use three seeds per comparison.

PARE's initially identical masks become distinct during recovery: their mean pairwise Hamming distance reaches 0.06017 at 2k, with no duplicate mask pair. Auxiliary controls learn the layerwise balanced single mask or tie route logits after entry from a strict 2:4 parent (Table 8).

### 5.4 Post-Selection Evaluation Across Corpora

Using checkpoints fixed before these evaluations, we test the development-buffer ordering on a source-corpus-excluded FineWeb-Edu shard and the external C4 validation corpus (Table 2).

Table 2: Fixed-checkpoint 159M PPL on source-excluded FineWeb-Edu and C4.

| Evaluation set | Fixed single-mask | PARE | Quota-preserving diverse | Recon.-calibrated diverse |
|---|---|---|---|---|
| FineWeb-Edu | $17.6866 \pm 0.0010$ | $\mathbf{17.2535 \pm 0.0007}$ | $17.4956 \pm 0.0014$ | $17.5238 \pm 0.0028$ |
| C4 validation | $24.8852 \pm 0.0019$ | $\mathbf{24.2904 \pm 0.0015}$ | $24.6089 \pm 0.0022$ | $24.6456 \pm 0.0044$ |

PARE retains its advantage over fixed-mask recovery and both diverse initializations on both corpora, preserving the ordering observed on the development buffer. Appendix D.2 provides the evaluation protocol.

### 5.5 Transfer Across Public Checkpoints

We compare PARE with fixed-mask recovery on TinyLlama-1.1B and Gemma-3-1B-PT, using 1k recovery steps under a 2k-step learning-rate schedule. Table 3 presents the PPL comparisons within each checkpoint.

Table 3: Public-checkpoint conversion PPL at 1k recovery steps (one seed, 2k learning-rate horizon). The diverse condition uses quota-preserving ownership exchanges. †Gemma retains its native `gelu_pytorch_tanh` gate. Comparisons within each row match recovery tokens and steps.

| Checkpoint | Gate | Fixed single-mask | PARE | Diverse ($n_m = 4$) |
|---|---|---|---|---|
| TinyLlama-1.1B | SiLU | 9.7291 | **9.6010** | 9.8102 |
| Gemma-3-1B-PT | GELU† | 13.8599 | **13.8145** | — |

On TinyLlama, PARE lowers PPL by 0.1281 relative to fixed-mask recovery at 1k steps and leads at every recorded post-entry step. It also lowers endpoint PPL by 0.2092 relative to the matched four-mask diverse entry, extending the entry-path comparison beyond the controlled checkpoint. On Gemma, PARE lowers endpoint PPL by 0.0454 and leads early, while fixed-mask recovery achieves lower PPL at steps 400–800 and lower post-200 NLL-AUC. Appendix E.5 provides the trajectories and mask diagnostics.

### 5.6 Complete-FFN Execution

We evaluate complete FFNs, including the down projection, on an RTX 4090 in 16-bit floating-point (FP16). For shared multi-mask blocks, each value projection is the unmasked dot product minus its masked gate projection. Our implementation fuses these projections with SiLU, gate–value multiplication, and route summation before the down projection. For recovered TinyLlama PARE, single-token execution rotating through 22 independently evaluated FFNs is $1.21$–$1.22\times$ faster than the dense source FFNs. Repeated same-layer calls yield $1.99\times$ over the dense source and $1.69$–$1.70\times$ over the public FlashMGLU projection with an eager epilogue (Appendix G.3).

For the first FFN of the strict $2{:}4$ single-mask 159M checkpoint, a CUTLASS implementation of $2{:}4$ structured-sparse general matrix multiplication (GEMM) uses NVIDIA Sparse Tensor Cores to compute two separately compressed gate and value projections. At 512–2048 tokens, it achieves $1.10$–$1.34\times$ speedup over dense GEMM of the materialized masked branches and $2.20$–$2.61\times$ over jointly fused batched MGLU. All implementations execute the same recovered checkpoint. At one token, local token-wise fusion is fastest among the tested implementations. The two paths address different execution costs: fusion reduces intermediate memory traffic and kernel launches, while structured-sparse GEMM exploits the fixed $2{:}4$ pattern to reduce projection arithmetic. Appendix G.4 provides the full comparisons and timing protocol.

## 6 Conclusion

PARE separates pretrained MGLU conversion into a generically lossy dense-to-shared branch projection and an exact repeated-mask expansion of the projected parent. Under the tested recovery-token budgets, repeating the parent mask and allowing masks to diverge improves endpoint perplexity over fixed-mask recovery and diverse entry into the same four-route architecture. These gains persist on both post-selection evaluation sets, and the TinyLlama entry-path comparison also favors PARE. The benefits of fused shared-weight and structured-sparse FFNs depend on token count and weight-access pattern. Future work will study $2{:}4$-constrained PARE with kernels that retain weight sharing across routes, connecting route expansion to structured-sparse execution.

## REPRODUCIBILITY STATEMENT

The experimental protocol, model configurations, fixed-buffer definition, and conversion rules are specified in the main text and appendices. Hardware records retain CUDA-event samples, input seeds, checkpoint provenance, mask diagnostics, and the fixed evaluation-buffer hash. The single-seed 10B dense continuation trajectory is a common reference for the three conversion seeds. Code, configurations, reference results, and ordered evaluation-document manifests for the main method and primary quality comparisons are available at https://anonymous.4open.science/r/PARE-6C5C/. An anonymized dense source checkpoint, export manifest, and validation record are available through https://osf.io/hwsny/overview?view_only=e4a51c23e1c9457598ebe14471357f87.

## AI USE STATEMENT

Generative AI tools assisted in proof auditing, experimental design and interpretation, code implementation and debugging, and manuscript polishing. Numerical results were generated using simulation scripts. The authors are solely responsible for all mathematical statements, interpretations, source code, experimental outcomes, and the final manuscript.

## REFERENCES


Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.298.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10865–10873, 2024. doi: 10.1609/aaai.v38i10.28960.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vXxardq6db.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. URL https://arxiv.org/abs/1308.3432.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, Charles Lovering, Niklas Muennighoff, Ellie Pavlick, Jason Phang, Aviya Skowron, Samson Tan, Xiangru Tang, Kevin A. Wang, Genta Indra Winata, François Yvon, and Andy Zou. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024. URL https://arxiv.org/abs/2405.14782v2.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.

Ruisi Cai, Yeonju Ro, Geon-Woo Kim, Peihao Wang, Babak Ehteshami Bejnordi, Aditya Akella, and Zhangyang Wang. Read-ME: Refactorizing LLMs as router-decoupled mixture of experts with system co-design. In *Advances in Neural Information Processing Systems*, volume 37, pp. 116126–116148, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/d298cf34e4539f9134db7f38b42f69fe-Abstract-Conference.html.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. bert2BERT: Towards reusable pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2134–2148. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.151.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.05641.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern,

Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with Pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. URL https://jmlr.org/papers/v24/22-1144.html.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. Pruner-Zero: Evolving symbolic pruning metric from scratch for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11346–11374. PMLR, 2024. URL https://proceedings.mlr.press/v235/dong24b.html.

Chaitanya Dwivedi, Binxuan Huang, Himanshu Gupta, Pratik Jayarao, Neeraj Varshney, and Bing Yin. Expert upcycling: Shifting the compute-efficient frontier of mixture-of-experts. *arXiv preprint arXiv:2604.19835*, 2026. URL https://arxiv.org/abs/2604.19835.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. MaskLLM: Learnable semi-structured sparsity for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 7736–7758, 2024. doi: 10.52202/079017-0248.

Yuchen Feng, Bowen Shen, Naibin Gu, Jiaxuan Zhao, Peng Fu, Zheng Lin, and Weiping Wang. DIVE into MoE: Diversity-enhanced reconstruction of large language models from dense into mixture-of-experts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19375–19394. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.951.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4475–4488, 2022. doi: 10.52202/068431-0323.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

Andrea Gesmundo and Kaitlin Maile. Composable function-preserving expansions for transformer architectures. *arXiv preprint arXiv:2308.06103*, 2023. URL https://arxiv.org/abs/2308.06103.

Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. AI and memory wall. *IEEE Micro*, 44(3):33–39, 2024. doi: 10.1109/MM.2024.3373763.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Dependency-aware semi-structured sparsity of GLU variants in large language models. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=T5OuTgPxHS.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=T5nUQDrM4u.

Jaeseong Lee, Seung-won Hwang, Wonpyo Park, and Mingi Ji. Breaking ReLU barrier: Generalized MoEfication for dense pretrained models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 10097–10107. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.563.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=dGVZwyq5tV.

Peiyu Liu, Tianwen Wei, Bo Zhu, Wayne Xin Zhao, and Shuicheng Yan. Masks can be learned as an alternative to experts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15800–15811. Association for Computational Linguistics, 2025b. doi: 10.18653/v1/2025.acl-long.768.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. Deja Vu: Contextual sparsity for efficient LLMs at inference time. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22137–22176. PMLR, 2023. URL https://proceedings.mlr.press/v202/liu23am.html.

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

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21702–21720, 2023. doi: 10.52202/075280-0950.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021. URL https://arxiv.org/abs/2104.08378.

Taishi Nakamura, Takuya Akiba, Kazuki Fujii, Yusuke Oda, Rio Yokota, and Jun Suzuki. Drop-Upcycling: Training sparse mixture of experts with partial re-initialization. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=gx1wHnf5Vp.

Ivan Novikov. MLPMoE: Zero-shot architectural metamorphosis of dense LLM MLPs into static mixture-of-experts. *arXiv preprint arXiv:2511.21089*, 2025. URL https://arxiv.org/abs/2511.21089.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *Proceedings of the 51st Annual International Symposium on Computer Architecture*, pp. 118–132, 2024. doi: 10.1109/ISCA59077.2024.00019.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849, 2024. doi: 10.52202/079017-0970.

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. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Yan Sun, Qixin Zhang, Zhiyuan Yu, Xikun Zhang, Li Shen, and Dacheng Tao. MaskPro: Linear-space probabilistic learning for strict (N:M)-sparsity on LLMs. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=0R06BghLJX.

Filip Szatkowski, Bartosz Wójcik, Mikołaj Piórczyński, and Simone Scardapane. Exploiting activation sparsity with dense to dynamic-$k$ mixture-of-experts conversion. In *Advances in Neural Information Processing Systems*, volume 37, pp. 43245–43273, 2024. doi: 10.52202/079017-1370.

Yukito Tajima, Nakamasa Inoue, Yusuke Sekikawa, Ikuro Sato, and Rio Yokota. Masked gated linear unit. In *Advances in Neural Information Processing Systems*, volume 38, pp. 150735–150763, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/dd73f39426a03131c38c8d943153d44b-Abstract-Conference.html.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

Vijay Thakkar, Pradeep Ramani, Cris Cecka, Aniket Shivam, Honghao Lu, Ethan Yan, Jack Kosaian, Mark Hoemmen, Haicheng Wu, Andrew Kerr, Matt Nicely, Duane Merrill, Dustyn Blasig, Aditya Atluri, Fengqi Qiao, Piotr Majcher, Paul Springer, Markus Hohnerbach, Jin Wang, and Manish Gupta. CUTLASS, 2023. URL https://github.com/NVIDIA/cutlass. Software.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David D. Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cDYRS5iZ16f.

Jiateng Wei, Quan Lu, Ning Jiang, Siqi Li, Jingyang Xiang, Jun Chen, and Yong Liu. Structured optimal brain pruning for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13991–14007, 2024. doi: 10.18653/v1/2024.emnlp-main.775.

Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 564–572. PMLR, 2016. URL https://proceedings.mlr.press/v48/wei16.html.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106. Association for Computational Linguistics, 2017. doi: 10.18653/v1/W17-4413.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ying Shan, and Ping Luo. LLaMA Pro: Progressive LLaMA with block expansion. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6518–6537. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.352.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=09iOdaeOzp.

Qifan Yu, Xinyu Ma, Zhijian Zhuo, Minrui Wang, Deyi Liu, Shiyi Zhan, Yiyuan Ma, Liang Xiang, Xingyan Bin, and Di He. SPARKLING: Balancing signal preservation and symmetry breaking for width-progressive learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=SXJBFREkWi.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1472.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. TinyLlama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024. URL https://arxiv.org/abs/2401.02385.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. MoEfication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 877–890. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.71.

Ziyu Zhao, Tong Zhu, Xin Yu, Zhi Zhang, Tiantian Fan, Jinluan Yang, Kun Kuang, Zhongyu Wei, Fei Wu, and Yu Cheng. ExpertWeaver: Unlocking the inherent MoE in dense LLMs with GLU activation patterns. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=cRs6HUUUay.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning $N$:$M$ fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=K9bw7vqp_s.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15913–15923. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.890.

Lancheng Zou, Shuo Yin, Zehua Pei, Tsung-Yi Ho, Farzan Farnia, and Bei Yu. PermLLM: Learnable channel permutation for $N$:$M$ sparse large language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0682.

# APPENDIX CONTENTS

**A Branch Projection: Proofs and Structural Consequences** **17**
A.1 Reconstruction Objective and Trace Identity . . . 17
A.2 Fixed-Mask Optimal Reconstruction . . . 17
A.3 Exact Branch Preservation . . . 18
A.4 Branch-Reconstruction Lower Bounds . . . 18
A.5 Diagonal Selection as a Special Case . . . 19
A.6 Nested Assignment Classes . . . 19

**B Exact Route Expansion and Optimization Boundaries** **19**
B.1 Exact Embedding and Function-Class Inclusion . . . 19
B.2 Symmetry and Mask-Logit Initialization . . . 20

**C Fixed-Class Initialization** **20**
C.1 First-Order Motivation and Activation-Aware Score . . . 20
C.2 Calibration-Moment Estimation . . . 21

**D Experimental Protocols and Reproducibility** **21**
D.1 Models, Stored State, and Checkpoint Sources . . . 21
D.2 Data Roles and Evaluation Sets . . . 21
D.3 Recovery Implementation and Run Pairing . . . 22
D.4 Metrics and Statistical Estimands . . . 23
D.5 Downstream Evaluation Protocol . . . 24
D.6 Reconstruction Calibration Protocol . . . 24

**E Route-Expansion Results and Controls** **24**
E.1 Parent and Diverse-Initialization Comparisons . . . 24
E.2 Reconstruction-Calibrated Diverse Entry . . . 24
E.3 Mask-Adaptation and Alternative-Entry Controls . . . 25
E.4 Fixed-Checkpoint Results Across Corpora . . . 27
E.5 Public-Checkpoint Evaluations . . . 27
E.6 Downstream Evaluation of PARE . . . 28

**F Single-Mask Assignment and Recovery Results** **29**
F.1 Single-Mask Assignment Structure . . . 29
F.2 Fixed-Class Initializer Recovery . . . 29
F.3 Public-Checkpoint Initializer Transfer . . . 29

**G Deployment Measurements** **30**
G.1 Implementation Details and Measurement Protocols . . . 30
G.2 Four-Mask Projection Kernels . . . 32
G.3 Fused FFN Measurements . . . 33
G.4 Structured Single-Mask FFN Execution . . . 34

**H Limitations and Discussion** **35**

# A BRANCH PROJECTION: PROOFS AND STRUCTURAL CONSEQUENCES

This appendix derives the branch-reconstruction results used in the main text, including the trace identity, fixed-mask projection, exact-preservation condition, lower bound, and class nesting. All claims use the row-vector convention and the dimensions stated around Equation (3). The symbols $m_k, w_k^g, w_k^v, w_k$ denote row $k$ and are represented as column vectors when used in quadratic forms.

## A.1 RECONSTRUCTION OBJECTIVE AND TRACE IDENTITY

**Lemma 1** (Exact branch-error decomposition)**.** *The weighted branch-preactivation objective satisfies the exact trace identity in Equation* (4)*.*

*Proof.* From Equation (3),

$$g - \widetilde{g} = xE_g^\top, \qquad v - \widetilde{v} = xE_v^\top.$$

The gate contribution is a scalar, so cyclic invariance of the trace gives

$$\begin{aligned}\mathbb{E}[xE_g^\top \Gamma_g E_g x^\top] &= \mathbb{E}\,\mathrm{Tr}(xE_g^\top \Gamma_g E_g x^\top) \\ &= \mathrm{Tr}\big(\Gamma_g E_g\, \mathbb{E}[x^\top x]\, E_g^\top\big) \\ &= \mathrm{Tr}(\Gamma_g E_g \Sigma E_g^\top).\end{aligned}$$

The value contribution is identical after replacing $E_g, \Gamma_g$ by $E_v, \Gamma_v$, which proves Equation (4). □

The exact-copy initializer in Equation (8) satisfies

$$M \odot W_0(M) = M \odot W_g, \qquad (1-M) \odot W_0(M) = (1-M) \odot W_v,$$

because $M$ is binary and complementary to $1 - M$. At $W = W_0(M)$, the errors reduce to $E_g = (1-M) \odot W_g$ and $E_v = M \odot W_v$. Since $W_0(M)$ is feasible for the minimization over $W$,

$$R(M) = \min_W \mathcal{L}(M, W) \le \mathcal{L}(M, W_0(M)).$$

## A.2 FIXED-MASK OPTIMAL RECONSTRUCTION

**Proposition 4** (Fixed-mask Schur projection)**.** *Assume $\Sigma \succ 0$ and that $\Gamma_g, \Gamma_v$ are positive diagonal. For each row $k$, completing the square gives*

$$w_I^* = w_I^g + \Sigma_{II}^{-1}\Sigma_{IJ} w_J^g, \qquad w_J^* = w_J^v + \Sigma_{JJ}^{-1}\Sigma_{JI} w_I^v, \tag{11}$$

*where $I = \{j : m_{kj} = 1\}$ and $J = I^c$, and $w_k^g, w_k^v, w_k$ are regarded as coefficient column vectors. With $\Sigma_{J|I} = \Sigma_{JJ} - \Sigma_{JI}\Sigma_{II}^{-1}\Sigma_{IJ}$ and $\Sigma_{I|J} = \Sigma_{II} - \Sigma_{IJ}\Sigma_{JJ}^{-1}\Sigma_{JI}$,*

$$R(M) = \sum_k \left[\alpha_k (w_J^g)^\top \Sigma_{J|I} w_J^g + \beta_k (w_I^v)^\top \Sigma_{I|J} w_I^v\right]. \tag{12}$$

*Full $\Sigma$ couples input coordinates and potentially different blocks, so it does not in general admit blockwise selection. Equations* (11)*–*(12) *assume that $I$ and $J$ are nonempty. If $I = \varnothing$, the optimal row is $w_k^v$, and the gate branch retains its full quadratic error. The case $J = \varnothing$ is symmetric.*

*Proof.* Fix a row $k$ and use the index sets $I$ and $J$ from the proposition. Treat the corresponding rows of $W_g, W_v, W$ as column vectors $w^g, w^v, w$. Because $\Gamma_g$ and $\Gamma_v$ are positive diagonal, the objective separates across rows and their positive scalar row weights do not change the fixed-row minimizers. For the gate branch, $e_I = w_I^g - w_I$ is the coefficient residual. Its unweighted quadratic term can be completed as

$$\begin{aligned}\begin{bmatrix} e_I \\ w_J^g \end{bmatrix}^\top \begin{bmatrix} \Sigma_{II} & \Sigma_{IJ} \\ \Sigma_{JI} & \Sigma_{JJ} \end{bmatrix} \begin{bmatrix} e_I \\ w_J^g \end{bmatrix} &= (e_I + \Sigma_{II}^{-1}\Sigma_{IJ} w_J^g)^\top \Sigma_{II} (e_I + \Sigma_{II}^{-1}\Sigma_{IJ} w_J^g) \\ &\quad + (w_J^g)^\top (\Sigma_{JJ} - \Sigma_{JI}\Sigma_{II}^{-1}\Sigma_{IJ}) w_J^g.\end{aligned} \tag{13}$$

Because $\Sigma \succ 0$, both $\Sigma_{II}$ and its Schur complement are positive definite. The first term in Equation (13) is uniquely minimized at

$$w_I^* = w_I^g + \Sigma_{II}^{-1}\Sigma_{IJ}w_J^g.$$

For the value branch, $e_J = w_J^v - w_J$ is the coefficient residual. The symmetric calculation gives

$$\begin{bmatrix} w_I^v \\ e_J \end{bmatrix}^\top \begin{bmatrix} \Sigma_{II} & \Sigma_{IJ} \\ \Sigma_{JI} & \Sigma_{JJ} \end{bmatrix} \begin{bmatrix} w_I^v \\ e_J \end{bmatrix} = (e_J + \Sigma_{JJ}^{-1}\Sigma_{JI}w_I^v)^\top \Sigma_{JJ}(e_J + \Sigma_{JJ}^{-1}\Sigma_{JI}w_I^v) + (w_I^v)^\top(\Sigma_{II} - \Sigma_{IJ}\Sigma_{JJ}^{-1}\Sigma_{JI})w_I^v.$$

Thus

$$w_J^* = w_J^v + \Sigma_{JJ}^{-1}\Sigma_{JI}w_I^v.$$

The coordinates $w_I$ occur only in the gate term and $w_J$ only in the value term, so the two minimizers are compatible. Multiplying by $\alpha_k, \beta_k$ and summing over rows proves Equations (11) and (12). □

### A.3 Exact Branch Preservation

*Proof of Proposition 1.* Positive definiteness lets each trace term be written as a squared Frobenius norm:

$$\mathrm{Tr}(\Gamma_g E_g \Sigma E_g^\top) = \|\Gamma_g^{1/2} E_g \Sigma^{1/2}\|_F^2,$$

and analogously for the value branch. Since $\Gamma_g, \Gamma_v, \Sigma$ are positive definite, for fixed $(M, W)$, $\mathcal{L}(M, W) = 0$ if and only if

$$W_g = M \odot W, \qquad W_v = (1 - M) \odot W.$$

Consequently, for a fixed mask $M$, there exists a shared matrix $W$ with $\mathcal{L}(M, W) = 0$ if and only if $(1-M)\odot W_g = \mathbf{0}_{h\times d}$ and $M \odot W_v = \mathbf{0}_{h\times d}$. When these conditions hold, $W_0(M)$ in Equation (8) realizes zero reconstruction error.

Within row-block $(k, b)$, $\mathcal{B}_b$ denotes its aligned set of $B$ input coordinates and $I_{kb} \subseteq \mathcal{B}_b$ its $N$ gate positions. Exact preservation requires

$$S_{kb}^g \subseteq I_{kb} \subseteq \mathcal{B}_b \setminus S_{kb}^v, \qquad |I_{kb}| = N.$$

This inclusion implies $S_{kb}^g \cap S_{kb}^v = \varnothing$, $|S_{kb}^g| \le N$, and $|S_{kb}^v| \le B - N$. Conversely, if these three conditions hold, include all of $S_{kb}^g$ in $I_{kb}$ and fill the remaining positions from coordinates in neither support. Copying $W_g$ on $I_{kb}$ and $W_v$ on its complement constructs a shared matrix with zero error. This proves Equation (5). □

### A.4 Branch-Reconstruction Lower Bounds

**Proposition 5** (Branch-reconstruction lower bound). *If $\Sigma \succeq \mu I_d$ for $\mu > 0$, with $I_d$ the $d \times d$ identity matrix, the optimized shared weights obey*

$$R(M) \ge \mu \sum_k \left[\alpha_k \|(1 - m_k) \odot w_k^g\|_2^2 + \beta_k \|m_k \odot w_k^v\|_2^2\right]. \tag{14}$$

*For any finite mask class without a support-compatible mask, its minimum branch-reconstruction error is strictly positive.*

*Proof.* For the lower bound, $\Sigma \succeq \mu I_d$ gives, row by row,

$$E_{g,k:}\Sigma E_{g,k:}^\top \ge \mu \|E_{g,k:}\|_2^2.$$

At coordinates where $m_{kj} = 0$, we have $(E_g)_{kj} = w_{kj}^g$, independently of the shared weights $W$. Therefore

$$\|E_{g,k:}\|_2^2 \ge \|(1 - m_k) \odot w_k^g\|_2^2.$$

The value branch gives the symmetric inequality on positions with $m_{kj} = 1$. After weighting and summing, this proves Equation (14) for every $W$, and hence for $R(M) = \min_W \mathcal{L}(M, W)$. If a finite mask class contains no support-compatible mask, every mask omits at least one nonzero coefficient. Then its minimum lower bound is strictly positive. □

### A.5 Diagonal Selection as a Special Case

**Proposition 6** (Diagonal Top-$N$ optimum)**.** *If $\Sigma = \mathrm{diag}(a_1, \ldots, a_d)$ with $a_j > 0$, then $W_0(M)$ is a fixed-mask optimizer and*

$$R(M) = \sum_{k,j} a_j \left[\alpha_k (1 - m_{kj})(w^g_{kj})^2 + \beta_k m_{kj} (w^v_{kj})^2\right]. \tag{15}$$

*With*

$$s_{kj} = a_j \left[\alpha_k (w^g_{kj})^2 - \beta_k (w^v_{kj})^2\right], \tag{16}$$

*the $N$ largest scores in each aligned row-block minimize this objective over $\mathcal{M}_{N:B}$. Ties may produce multiple optima.*

*Proof.* If $\Sigma = \mathrm{diag}(a_1, \ldots, a_d)$, then $\Sigma_{IJ} = \mathbf{0}_{|I|\times|J|}$ and $\Sigma_{JI} = \mathbf{0}_{|J|\times|I|}$. The Schur solution reduces to $w^*_I = w^g_I$ and $w^*_J = w^v_J$, so $W_0(M)$ is a fixed-mask optimizer and Equation (15) follows. For row $k$ and aligned block $\mathcal{B}_b$, the mask-dependent part of the objective is $-\sum_{j\in\mathcal{B}_b} m_{kj} s_{kj}$, where the omitted term is independent of $M$. The strict quota requires exactly $N$ selected entries in the block, so the objective is minimized by choosing the $N$ largest $s_{kj}$. The row-block variables are disjoint, so these local choices form a global minimizer over $\mathcal{M}_{N:B}$. Ties at the selection boundary may be broken arbitrarily without affecting optimality. □

### A.6 Nested Assignment Classes

*Proof of Proposition 2.* For aligned partitions with a common origin, each 1:2 mask has two selected entries in every 4-block, so it is also a 2:4 mask. Each 2:4 mask has four selected entries in every 8-block, which consists of two aligned 4-blocks, so it is also a 4:8 mask. Every structured half-density mask also has $hd/2$ selected entries over the full matrix. This proves the inclusions in Equation (6). Minimizing the same $R(M)$ over successively larger feasible sets gives

$$P(\mathcal{M}_{1:2}) \geq P(\mathcal{M}_{2:4}) \geq P(\mathcal{M}_{4:8}) \geq P(\mathcal{M}_{\mathrm{bal},50\%}).$$

□

**Selector and structural projection gaps.** The following decomposition separates assignment-class restrictions from the suboptimality of a particular selection rule. For feasible mask classes $\varnothing \neq \mathcal{C} \subseteq \mathcal{C}_0$, $\widehat{M}_{\mathcal{C}} \in \mathcal{C}$ denotes the output of an arbitrary selection rule. Adding and subtracting $P(\mathcal{C})$ gives

$$R(\widehat{M}_{\mathcal{C}}) - P(\mathcal{C}_0) = \underbrace{R(\widehat{M}_{\mathcal{C}}) - P(\mathcal{C})}_{\Delta_{\mathrm{sel}}\geq 0} + \underbrace{P(\mathcal{C}) - P(\mathcal{C}_0)}_{\Delta_{\mathrm{struct}}\geq 0}. \tag{17}$$

The first bracket is nonnegative because $P(\mathcal{C})$ minimizes $R$ on $\mathcal{C}$; the second is nonnegative because $\mathcal{C} \subseteq \mathcal{C}_0$. In Equation (17), nestedness controls only the structural term. No comparable monotonicity follows for the surrogate-selection term.

## B Exact Route Expansion and Optimization Boundaries

We first prove the exact embedding and its function-class consequence, then separate exact binary-mask symmetry from the untied mask-logit initialization used in the experiments.

### B.1 Exact Embedding and Function-Class Inclusion

*Proof of Proposition 3.* Fix $\phi$. When $M_1 = \cdots = M_{n_m} = M$, every route response equals $q^{(\phi)}(M, W; x)$. Therefore

$$\left[\sum_{i=1}^{n_m} q^{(\phi)}(M, W; x)\right] \frac{W_o}{n_m} = n_m q^{(\phi)}(M, W; x) \frac{W_o}{n_m} = q^{(\phi)}(M, W; x) W_o,$$

which proves Equation (7) pointwise in $x$. Adding the same residual preserves equality, and composition with the same deterministic surrounding maps preserves it as well.

The construction maps every function in $\mathcal{F}_1^{(\phi)}$ to an equal function in $\mathcal{F}_{n_m}^{(\phi)}$, proving $\mathcal{F}_1^{(\phi)} \subseteq \mathcal{F}_{n_m}^{(\phi)}$. □

**Risk corollary.** For any output-dependent population or empirical loss $\mathcal{R}$, the inclusion above gives

$$\inf_{f \in \mathcal{F}_{n_m}^{(\phi)}} \mathcal{R}(f) \leq \inf_{f \in \mathcal{F}_1^{(\phi)}} \mathcal{R}(f).$$

### B.2 Symmetry and Mask-Logit Initialization

**Lemma 2** (Symmetry preservation under equivariant updates)**.** *$\mathcal{S}_t$ comprises all route parameters, route-specific optimizer states, and shared state, while $\mathsf{P}_\rho$ permutes every route-indexed component according to a route permutation $\rho$. If a deterministic update $\mathcal{U}_t$ is permutation equivariant, $\mathcal{U}_t(\mathsf{P}_\rho \mathcal{S}) = \mathsf{P}_\rho \mathcal{U}_t(\mathcal{S})$, and $\mathsf{P}_\rho \mathcal{S}_0 = \mathcal{S}_0$ for every $\rho$, then $\mathsf{P}_\rho \mathcal{S}_t = \mathcal{S}_t$ for every finite $t$.*

*Proof.* Assume $\mathsf{P}_\rho \mathcal{S}_t = \mathcal{S}_t$. Permutation equivariance gives

$$\mathsf{P}_\rho \mathcal{S}_{t+1} = \mathsf{P}_\rho \mathcal{U}_t(\mathcal{S}_t) = \mathcal{U}_t(\mathsf{P}_\rho \mathcal{S}_t) = \mathcal{U}_t(\mathcal{S}_t) = \mathcal{S}_{t+1}.$$

The premise holds at $t = 0$, so induction proves it for every finite step. □

This symmetry condition applies to the full update state, including optimizer moments and any realized route-specific randomness.

**Mask-logit threshold-crossing analysis.** A hard-assignment entry is $m_{ij}^{(t)} = \mathbf{1}\{\pi_{ij}^{(t)} > 0\}$, where $i$ indexes a route and $j$ a flattened mask coordinate. To isolate the effect of unequal initial margins, fix a gradient component $d_j$ and consider the comparison process

$$\pi_{ij}^{(t)} = \pi_{ij}^{(0)} - \eta t d_j, \qquad \eta > 0, \quad \pi_{ij}^{(0)} d_j > 0.$$

The first binary-mask change in this comparison process is

$$\tau_{ij} = \begin{cases} \left\lceil |\pi_{ij}^{(0)}| / (\eta |d_j|) \right\rceil, & \pi_{ij}^{(0)} > 0, \\ \left\lceil |\pi_{ij}^{(0)}| / (\eta |d_j|) \right\rceil + 1, & \pi_{ij}^{(0)} < 0. \end{cases}$$

PARE initializes identical binary masks with different logit margins, so its initial parameter state does not satisfy the symmetry condition in Lemma 2. The fixed-gradient calculation above illustrates how these margins can produce different mask-switching times even under identical updates. During recovery, the switching times also depend on the evolving gradients and optimizer states.

## C Fixed-Class Initialization

### C.1 First-Order Motivation and Activation-Aware Score

For input row vector $x$, hidden channel $k$, and a differentiable gate $\phi$ (SiLU in the primary experiments), the dense preactivations and output are

$$g_k = \sum_j w_{kj}^g x_j, \qquad v_k = \sum_j w_{kj}^v x_j, \qquad z_k = \phi(g_k) v_k.$$

Under the coefficient-copy initializer in Equation (8), the branch perturbations are

$$\delta g_k = -\sum_j (1 - m_{kj}) w_{kj}^g x_j, \qquad \delta v_k = -\sum_j m_{kj} w_{kj}^v x_j.$$

The first-order hidden response at the teacher activation is

$$\Delta z_k \approx \underbrace{\phi'(g_k) v_k}_{\text{gate sensitivity}} \delta g_k + \underbrace{\phi(g_k)}_{\text{value sensitivity}} \delta v_k.$$

Approximating $\sum_k \mathbb{E}[(\Delta z_k)^2]$ by dropping branch and coordinate cross terms and factorizing mixed moments gives the separable objective

$$\sum_{k,j} a_j \left[\alpha_k(1-m_{kj})(w^g_{kj})^2 + \beta_k m_{kj}(w^v_{kj})^2\right],$$

where $a_j$, $\alpha_k$, and $\beta_k$ are the corresponding factorized second moments. This has the same separable form as Equation (15), with the score in Equation (16). Its empirical estimate is

$$\widehat{s}_{kj} = \widehat{a}_j \left[(w^g_{kj})^2\widehat{\alpha}_k - (w^v_{kj})^2\widehat{\beta}_k\right]. \tag{18}$$

The mask selects the Top-$N$ scores within each aligned $B$-element block for strict $N{:}B$, or the Top-$hd/2$ scores over the full layer for layerwise balanced. Magnitude uses the same selection rules with score $|w^g_{kj}| - |w^v_{kj}|$, without activation statistics.

The approximation omits the Taylor remainder, full input correlation, input–sensitivity dependence, and down-projection coupling.

### C.2 Calibration-Moment Estimation

For calibration examples $\{x_t\}_{t=1}^{n_{\text{cal}}}$, the factors used in Equation (18) are estimated as

$$\widehat{a}_j = \frac{1}{n_{\text{cal}}}\sum_t x^2_{tj}, \qquad \widehat{\alpha}_k = \frac{1}{n_{\text{cal}}}\sum_t [\phi'(g_{tk})v_{tk}]^2, \qquad \widehat{\beta}_k = \frac{1}{n_{\text{cal}}}\sum_t \phi(g_{tk})^2,$$

where $g_{tk} = w^g_k x_t^\top$ and $v_{tk} = w^v_k x_t^\top$ are computed with the dense teacher before any conversion.

## D Experimental Protocols and Reproducibility

This appendix consolidates the shared model, data, recovery, metric, and numerical conventions used by the experiments.

### D.1 Models, Stored State, and Checkpoint Sources

Table 4 lists the controlled model dimensions and floating-point parameter counts. Public experiments use the `TinyLlama-1.1B` intermediate checkpoint `step-1431k-3T` (Zhang et al., 2024) and Gemma-3-1B-PT (Team et al., 2025), with their native tokenizers.

Table 4: Model configurations and floating-point weight-parameter counts. The 159M row retains the reference configuration label, and counts use our tokenizer and tied embeddings. Shared MGLU counts exclude binary masks and training-time mask logits, which are reported separately below.

| Scale | Layers | Hidden | FFN | Heads | Dense | Shared MGLU |
|---|---|---|---|---|---|---|
| 159M | 12 | 768 | 3,072 | 24 | 140.987M | 112.676M |

For the 159M configuration, one mask contains 28,311,552 binary assignment coordinates. Table 5 separates the shared floating-point model weights from mask state used only during training and from packed deployment metadata. This distinction matters because the exactly expanded model shares $W$ but trains $n_m = 4$ dense FP32 mask-logit tensors.

This advantage is precision dependent. With $b$-bit weights and ideal 1-bit masks, the complete-block MGLU-to-dense ratio is $(2b + n_m)/(3b)$. At $b = 4$ and $n_m = 4$, it equals one before quantization scales, packing alignment, and workspace.

### D.2 Data Roles and Evaluation Sets

FineWeb-Edu `sample-10BT` supplies source-pretraining candidates and the recovery stream. Calibration tokens, the repeatedly consulted development buffer, the source-excluded FineWeb-Edu

Table 5: Model weights, training-only mask logits, and deployment metadata are distinct quantities. Deployment totals assume BF16 weights and bit-packed binary masks. Sizes exclude gradients, optimizer moments, activations, allocator overhead, and checkpoint serialization overhead. A strict 2:4 mask instead uses 2.531 MiB of pattern metadata in this model.

| Stored state (159M) | Dense | Fixed single-mask | PARE ($n_m = 4$) |
|---|---|---|---|
| Floating-point model weights | 140.987M | 112.676M | 112.676M |
| BF16 weight bytes | 268.91 MiB | 214.91 MiB | 214.91 MiB |
| Trainable FP32 mask logits | 0 MiB | 0 MiB | 432 MiB |
| Packed binary masks at deployment | 0 MiB | 3.375 MiB | 13.500 MiB |
| Analytical deployment total | 268.91 MiB | 218.29 MiB | 228.41 MiB |
| Reduction from dense | N/A | 18.8% | 15.1% |

evaluation, and C4 have distinct roles and artifacts. Calibration and recovery documents are separated from the development buffer. The development buffer's exact overlap with the documents consumed during source pretraining remains unknown. The two post-selection evaluation sets were evaluated only after methods and checkpoints were fixed.

We use FineWeb-Edu shard `004_00002.parquet` from `sample/100BT`, starting at row zero. We remove 534 documents whose raw text exactly matches any document in the 14 `sample-10BT` shards available for source pretraining. The English C4 validation split (Raffel et al., 2020) uses dataset revision `1588ec4`, starting at row zero. The same exclusion index removes one document. We also skip C4 documents with fewer than two tokens after end-of-sequence handling and retain the remaining documents in their original order until reaching the three-million-token target.

For both evaluation sets, each document receives one end-of-sequence token and is evaluated independently with a 1,024-token context cap. Long documents are windowed with the final context token carried into the next window, so every next-token target is counted once and no prediction crosses a document boundary. C4 evaluation uses BF16.

### D.3 Recovery Implementation and Run Pairing

The primary 159M recovery runs use sequence length 1,024 and a global batch of 512 sequences, or 524,288 tokens per optimizer step. AdamW uses momentum coefficients $(0.9, 0.99)$, a numerical-stability term of $10^{-8}$, weight decay 0.1 for matrix weights, and gradient clipping at 1. The conversion schedule warms up for 20 steps and decays from $3 \times 10^{-5}$ to $3 \times 10^{-6}$ over the declared horizon. Primary mask logits use the base learning rate and zero weight decay. Fixed-mask recovery keeps assignments frozen.

The 159M $n_m = 4$ runs on RTX 4090 and RTX 3090 servers use the same source checkpoint, tokenizer, recovery data, and evaluation buffer. For run seed $s$ and zero-indexed layer $\ell$, mask-logit margin sampling uses seed $1{,}000{,}033s + 1{,}013\ell + 17$.

The 10% quota-preserving control keeps $M_1$ unchanged and, for $M_2, M_3, M_4$, swaps $\mathrm{round}(0.10hd/2)$ gate positions with the same number of value positions for a layer mask of size $hd$. For mask index $i \in \{2, 3, 4\}$, the permutation seed is $1{,}000{,}003s + 1{,}009\ell + (i-1)$. PARE and diverse conditions share the resulting absolute mask-logit margins.

TinyLlama experiments use 65,536 calibration tokens and the fixed 524,800-token buffer. The strict 2:4 initializer evaluation uses one seed and a 200-step learning-rate schedule. The $n_m = 1$ and $n_m = 4$ evaluation from a layerwise balanced parent uses a 2k learning-rate horizon, 1k recovery steps, 524,288 tokens per optimizer step, and the same source, assignment, data order, and evaluation buffer. The fixed and learned $n_m = 1$ conditions use one RTX 4090, while PARE and the matched diverse entry use two-GPU per-layer Fully Sharded Data Parallel (FSDP). The diverse entry retains the projected shared weights and the $1/4$ down-projection scaling, keeps the first parent mask, and applies 10% quota-preserving ownership exchanges to the other three masks. Both four-mask conditions use the same parameterization, optimizer settings, recovery-token order, and evaluation buffer.

Gemma-3-1B-PT retains its native `gelu_pytorch_tanh` gate. Calibration uses this tanh-approximated GELU and its corresponding derivative. Its three conditions share the source checkpoint, projected layerwise balanced mask, and data order, with 524,288 tokens per step and 1k recovery steps under a 2k learning-rate horizon. PARE uses two-GPU FSDP. Trajectory NLL-AUC uses uncapped NLL.

In the public learned single-mask control, a layerwise Top-$K$ operation retains the 50% gate quota throughout recovery, with an identity straight-through derivative. Its initial absolute mask-logit margins are sampled uniformly from $[0.005, 0.01]$. Public PARE runs instead use independent half-normal margins with scale 0.01 and unconstrained sign-threshold updates, while the TinyLlama diverse entry uses the same margin distribution and update rule. All public conditions with trainable masks use the base learning rate and AdamW decay 0.1 for mask logits. All non-mask Transformer parameters remain trainable in both the 159M and public protocols.

The 159M four-mask experiments use vectorized summation of BF16 route outputs, whereas the public-checkpoint experiments use sequential accumulation, following the released MGLU implementation (Tajima et al., 2025).[1] On a representative 159M checkpoint, the two reduction orders yield a fixed-buffer PPL difference below $10^{-4}$.

**Alternative initialization controls.** The controls in Table 9 start from the strict 2:4 parent. The two perturbed variants keep its first mask and swap 10% of coordinates in each remaining mask using the quota-preserving rule above. These perturbed masks retain the layerwise quota but need not retain local 2:4 structure. They differ only in their initial absolute logit margins: constant 0.01 or independent half-normal samples with scale 0.01. The identical-mask variant repeats the parent mask with the same half-normal margin rule.

All three retain the parent's down projection and scale shared $W$ by a single scalar chosen from 25 evenly spaced values in $[0.25, 0.85]$ to minimize squared FFN-output error at the dense source's layer inputs over 65,536 calibration tokens. The selected scale is 0.65 for the perturbed variants and 0.625 for identical masks. Recovery then uses unconstrained elementwise mask updates.

On the same RTX 4090, median recovery-step time is 11.63 s for fixed-mask recovery and 29.86 s for four-mask PARE, excluding evaluation and checkpointing.

### D.4 Metrics and Statistical Estimands

Perplexity is the exponential of token-weighted mean next-token NLL. Endpoint contrasts are formed within recovery seed before aggregation. For route-expansion comparisons, recovery NLL-AUC is

$$\mathcal{U}_c(a, b) = \int_a^b \mathrm{NLL}_c(t)\, dt,$$

computed by trapezoidal integration on the common recorded step grid for each comparison. Full and post-200 areas start at $a = 0$ and $a = 200$, respectively. Paired area gaps subtract PARE's area within seed, so positive gaps favor PARE. The tables specify the grids and report areas in NLL-steps.

Mean pairwise mask Hamming distance is the normalized binary-assignment disagreement averaged over unordered mask pairs and layers, and the duplicate fraction records the fraction of identical mask pairs. Coordinate-role entropy is the normalized binary entropy of each coordinate's gate frequency across masks, averaged over coordinates and layers. Local-$N$:$B$ compliance is the fraction of mask row-blocks that retain the stated quota.

For the fixed-class initializer comparison, piecewise-linear interpolation gives the excess NLL-AUC

$$\mathcal{A}_c(H) = \int_0^H [\mathrm{NLL}_c(t) - \mathrm{NLL}_{\mathrm{dense}}(t)]\, dt,$$

where $\mathrm{NLL}_c(t)$ is the fixed-buffer NLL for conversion condition $c$, and $\mathrm{NLL}_{\mathrm{dense}}(t)$ is that of dense continuation. We present its relative reduction from magnitude to activation-aware as

$$\frac{\mathcal{A}_{\mathrm{mag}}(H) - \mathcal{A}_{\mathrm{act}}(H)}{\mathcal{A}_{\mathrm{mag}}(H)}.$$

[1] Official repository: https://github.com/tjy-dev/masked-gated-linear-unit.

We also present the dense-independent paired area $\int_0^H [\mathrm{NLL}_{\mathrm{mag}}(t) - \mathrm{NLL}_{\mathrm{act}}(t)]\, dt$. These areas summarize recovery NLL over optimizer steps at a fixed token budget per step.

### D.5 Downstream Evaluation Protocol

Downstream evaluation uses the Language Model Evaluation Harness (v0.4.12) (Biderman et al., 2024) in the zero-shot setting, with the TinyLlama tokenizer, a maximum context length of 1,024 tokens, BF16 precision, and a batch size of four. The tasks, scoring conventions, and results are presented in Appendix E.6.

### D.6 Reconstruction Calibration Protocol

The reconstruction control freezes each diverse entry's binary masks, mask-logit margins, and surrounding Transformer parameters, and fits only the shared $W$ and down projection $W_o$ to the projected single-mask parent. Across $n_{\mathrm{layer}}$ layers, let $x_{\ell t}^{\mathrm{parent}}$ and $y_{\ell t}^{\mathrm{parent}}$ be the projected parent's FFN inputs and outputs at calibration token $t$. Each AdamW update minimizes

$$\frac{1}{n_{\mathrm{layer}}} \sum_{\ell=1}^{n_{\mathrm{layer}}} \frac{\sum_t \| F_\ell^{\mathrm{recon}}(x_{\ell t}^{\mathrm{parent}}) - y_{\ell t}^{\mathrm{parent}} \|_2^2}{\max\{\sum_t \| y_{\ell t}^{\mathrm{parent}} \|_2^2, 10^{-30}\}}.$$

The sums run over the current 1,024-token calibration block. Each layer receives the frozen parent's inputs throughout calibration, so upstream student changes do not alter its reconstruction targets.

Calibration uses 65,536 tokens. It cycles one 1,024-token sequence per update over 64 frozen calibration blocks and optimizes all layers in one AdamW instance. It applies 150 shared-$W$-only updates, solves each $W_o$ by ridge regression over the full calibration set, and then performs 50 joint $W, W_o$ updates. Both AdamW stages use learning rate $10^{-4}$, momentum coefficients $(0.9, 0.99)$, a numerical-stability term of $10^{-8}$, zero weight decay, and gradient clipping at 1.0. The ridge solve uses the normalized Gram matrix, a prior at the incoming $W_o$, and a penalty of $10^{-3}$ times the Gram diagonal mean.

A disjoint 16,384-token stream measures held-out normalized FFN-output mean-squared error with fixed parent inputs. For each layer, the numerator and denominator are accumulated over all held-out blocks before division, and the resulting ratios are averaged across layers. Entry NLL excess is measured separately on the development buffer. The prespecified target is to remove at least 80% of this NLL excess while reducing the held-out FFN error and retaining finite NLL. Calibration optimizer state is discarded before the common 2k recovery run.

## E Route-Expansion Results and Controls

This appendix presents the primary route-expansion comparisons first, followed by fixed-checkpoint evaluations across corpora, public-checkpoint evaluations, and downstream results.

### E.1 Parent and Diverse-Initialization Comparisons

For the comparisons in Table 1, PARE's paired endpoint improvements are $0.4048 \pm 0.0027$ PPL over fixed single-mask recovery and $0.2114 \pm 0.0046$ PPL over the 10% quota-preserving diverse entry. Table 6 complements these endpoints with trajectory areas and paired gaps under the protocol in Section 5.3.

Full NLL-AUC is sensitive to sampling during the rapid initial recovery. Including the recorded steps 50, 100, and 150 reduces the mean diverse-minus-PARE full-area gap from 120.6770 to 60.7305 NLL-steps. The post-200 gap is unchanged at 31.3709 NLL-steps, corresponding to an average NLL gap of 0.01743 over steps 200–2000.

### E.2 Reconstruction-Calibrated Diverse Entry

The matched diverse baseline retains the single-mask parent's shared $W$ after swapping branch ownership in routes 2–4. A swapped coordinate can therefore apply a coefficient copied for one source

Table 6: Trajectory summaries underlying Table 1. All NLL-AUC values use trapezoidal integration on the common $0, 200, \ldots, 2000$ grid and are expressed in NLL-steps. Each comparison uses its own paired PARE runs. Positive gaps favor PARE, and the post-200 column excludes the initial 200-step window.

| Comparison | Condition | Full NLL-AUC | Gap to PARE | Post-200 gap |
|---|---|---|---|---|
| Parent comparison | PARE | $5704.4040 \pm 0.1770$ | 0 | 0 |
| | Fixed single-mask | $5747.4107 \pm 0.2514$ | $+43.0067 \pm 0.1640$ | $+40.0666 \pm 0.1326$ |
| Entry-path comparison | PARE | $5704.3664 \pm 0.1478$ | 0 | 0 |
| | Quota-preserving diverse entry | $5825.0434 \pm 7.2841$ | $+120.6770 \pm 7.1937$ | $+31.3709 \pm 0.3985$ |
| | Reconstruction-calibrated diverse entry | $5748.0462 \pm 0.8914$ | $+43.6799 \pm 0.9946$ | $+33.9240 \pm 0.4347$ |

branch to the other branch. The reconstruction-calibrated condition tests whether this coefficient-role mismatch is sufficient to explain the path gap. It keeps each seed's diverse binary masks and absolute mask-logit margins unchanged, freezes all surrounding parameters, and reconstructs only $W$ and $W_o$ on a separate calibration stream using the fixed-parent-input objective and optimization schedule in Appendix D.6.

For each seed, "entry NLL removed" is the fraction of the diverse entry's NLL excess above PARE removed by reconstruction:

$$\frac{\mathrm{NLL}_{\mathrm{diverse}} - \mathrm{NLL}_{\mathrm{recon}}}{\mathrm{NLL}_{\mathrm{diverse}} - \mathrm{NLL}_{\mathrm{PARE}}}.$$

On the disjoint reconstruction validation, mean normalized FFN-output mean-squared error falls from approximately 0.0656 to 0.0402–0.0408.

Table 7 gives the per-seed reconstruction and recovery results.

Table 7: Reconstruction-calibrated diverse entry and recovery. Entry NLL removed is summarized by its range across seeds. Positive gaps favor PARE. Full NLL-AUC uses the common $0, 200, \ldots, 2000$ grid, while post-200 NLL-AUC excludes the initial 200-step window.

| Seed | Entry NLL removed | Recon. PPL@2k | Recon.−PARE PPL@2k | Full NLL-AUC gap | Post-200 NLL-AUC gap |
|---|---|---|---|---|---|
| 1 | 91.05% | 15.8906 | +0.2392 | +44.5442 | +34.3334 |
| 2 | 91.53% | 15.8845 | +0.2329 | +43.9026 | +33.4678 |
| 3 | 94.14% | 15.8909 | +0.2401 | +42.5928 | +33.9707 |
| Summary | 91.05–94.14% (range) | 15.8887 $\pm$0.0036 | +0.2374 $\pm$0.0039 | +43.6799 $\pm$0.9946 | +33.9240 $\pm$0.4347 |

Reconstruction reduces the initial discrepancy and full NLL-AUC relative to the uncalibrated diverse entry, but does not improve its 2k endpoint or post-200 trajectory. The recovery gap to PARE therefore persists after calibration. Residual function error and calibration-induced parameter changes remain coupled in this comparison.

### E.3 Mask-Adaptation and Alternative-Entry Controls

**Single-mask adaptation from the primary parent.** From the layerwise balanced parent, PARE lowers endpoint PPL by 0.4042 relative to fixed-mask recovery, whereas learning the single mask increases PPL by 0.2762 (Table 8).

**Strict-parent multi-mask controls.** From the strict 2:4 parent, four routes with independent mask logits lower PPL by 0.2745 relative to the single-mask reference. Sharing mask logits across routes increases PPL by 0.5183 relative to independent logits.

Table 8: Single-seed mask-adaptation and route-expansion comparisons at 2k recovery steps, grouped by parent assignment. The strict-parent multi-route controls repeat the parent mask and rescale the shared $W$, whereas PARE rescales $W_o$ to preserve its projected parent's function.

| Parent | Condition | PPL@2k |
|---|---|---|
| Layerwise balanced | Fixed single-mask | 16.0578 |
| | Learned single-mask | 16.3340 |
| | PARE ($n_m = 4$) | 15.6536 |
| Strict 2:4 | Fixed single-mask | 16.3535 |
| | Four routes, independent mask logits | 16.0790 |
| | Four routes, shared mask logits | 16.5973 |

Across three seeds, the strict single-mask and untied $n_m = 4$ endpoints are $16.3530 \pm 0.0017$ and $16.0792 \pm 0.0056$, with paired gain $0.2738 \pm 0.0040$ and no duplicated terminal mask pair. A PARE run from the strict parent reaches 15.8665.

Additional initializations from this strict parent use $n_m = 4$ conversion under the original raw-sum aggregation, a shared $W$, and deterministic binary-mask STE. The initialization rules and shared-weight scales are specified in Appendix D.3. Table 9 compares the protocol-matched 200-step trajectories. All conditions share the source, calibration and evaluation artifacts, seed, recovery data order, and 2k learning-rate horizon, with token and step budgets matched to the single-mask reference.

Table 9: Single-seed $n_m = 4$ initializations that do not preserve the parent function. The identical-mask entry scales shared $W$, instead of applying PARE's $W_o/4$ scaling. The identical-mask initializer improves the matched single-mask endpoint by 0.2658 PPL and NLL-AUC by 1.796% at 200 steps.

| Method | PPL@0 | @50 | @100 | @150 | @200 | NLL-AUC |
|---|---|---|---|---|---|---|
| Fixed single-mask | 111.5572 | 20.3659 | 19.0114 | 18.4158 | 18.0453 | 633.7914 |
| Perturbed / constant margins | 104.0559 | 21.9075 | 20.0725 | 19.2856 | 18.7956 | 641.7404 |
| Perturbed / sampled margins | 104.0559 | 21.2540 | 19.5633 | 18.8411 | 18.3840 | 637.2224 |
| Identical / sampled margins | **78.7826** | **20.0415** | **18.7221** | **18.1423** | **17.7795** | **622.4064** |

The identical-mask initializer's recovery advantage persists through 2k steps, with 0.659% lower full NLL-AUC than the matched single-mask reference. Its initially identical masks diverge under independent mask-logit margins, reaching mean pairwise Hamming distance 0.0725 with no duplicate pair at 2k. Figure 4 shows the recovery and mask trajectories.

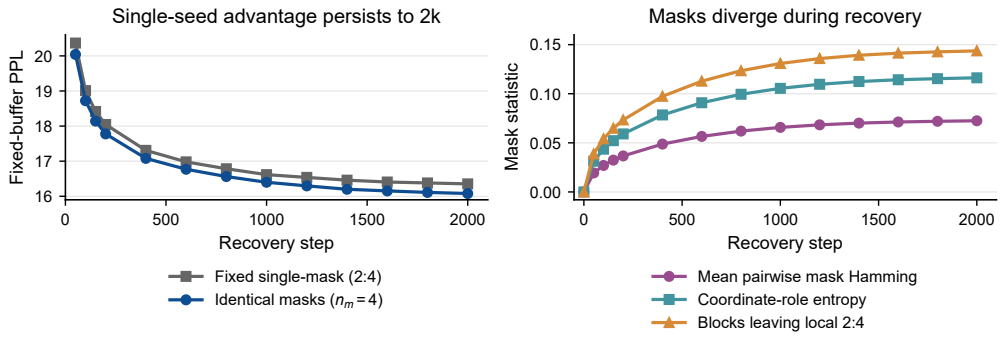


Figure 4: Single-seed recovery from an identical-mask $n_m = 4$ initialization that scales shared $W$ and does not preserve the parent function. Left: the identical-mask initializer remains below the protocol-matched single-mask reference at every recorded recovery point after conversion. Right: mean pairwise mask Hamming distance and role entropy increase while some blocks leave the initial local-2:4 pattern.

**Assignment constraints and recovery schedules.** Relaxing exact balance, learning a single mask, and the tested structural curriculum do not improve recovery in these controls. Without the layerwise balanced quota, activation-aware and magnitude assignments reach PPL 17.4405 and 17.4479 at step 200, compared with 17.4400 and 17.4443 under exact balance.

Learning a single conversion mask also performs worse than freezing it, reaching PPL 19.8981 versus 18.0453 at step 200, with 2.451% higher NLL-AUC.

Strict 2:4 warm starts released to a layerwise balanced mask at steps 50 and 100 reach PPL 16.0639 and 16.0732 at step 2k, respectively, versus 16.0578 for direct layerwise balanced recovery. Over the full recovery trajectory, these warm starts also incur 0.765% and 0.812% higher NLL-AUC, respectively.

### E.4 Fixed-Checkpoint Results Across Corpora

To evaluate recovery beyond the development buffer, we use source-excluded FineWeb-Edu and English C4 under the protocol in Appendix D.2. The FineWeb-Edu set contains 2,381 documents and 3,000,071 prediction tokens. At the frozen endpoint, dense continuation and the frozen dense source reach $16.01661 \pm 0.00034$ and 16.23744 PPL, respectively. The converted results are in Table 2.

To test whether the 159M endpoint ordering is specific to FineWeb-Edu, we also evaluate the fixed checkpoints on 5,611 C4 validation documents containing 3,001,745 prediction tokens. Table 10 presents the C4 evaluation results.

Table 10: Fixed-checkpoint 159M evaluation on C4, grouped by stage. Frozen dense and conversion-time results each use one checkpoint per condition. Dense continuation and all recovered conditions use 2k training steps. Bold values identify the best results among the four recovered conversion conditions.

| Stage | Condition | NLL | PPL |
|---|---|---|---|
| Dense references | Frozen dense | 3.130455 | 22.8844 |
| | Dense continuation | $3.118492 \pm 0.000066$ | $22.6123 \pm 0.0015$ |
| At conversion | Projected parent | 4.588139 | 98.3114 |
| | PARE | 4.588139 | 98.3114 |
| After recovery | Fixed single-mask | $3.214272 \pm 0.000074$ | $24.8852 \pm 0.0019$ |
| | PARE | $\mathbf{3.190082 \pm 0.000061}$ | $\mathbf{24.2904 \pm 0.0015}$ |
| | Quota-preserving diverse entry | $3.203107 \pm 0.000089$ | $24.6089 \pm 0.0022$ |
| | Reconstruction-calibrated diverse entry | $3.204597 \pm 0.000177$ | $24.6456 \pm 0.0044$ |

On C4, PARE has lower endpoint NLL than fixed-mask recovery, quota-preserving diverse entry, and reconstruction-calibrated diverse entry by 0.024190, 0.013025, and 0.014515, respectively. This ordering is consistent with the FineWeb-Edu results. PARE remains above the frozen dense source by 0.059627 NLL.

### E.5 Public-Checkpoint Evaluations

**TinyLlama recovery from a layerwise balanced parent.** Table 11 presents recovery from the layerwise balanced parent using a 2k learning-rate horizon.

At step 1k, PARE improves PPL by 0.1281 over fixed-mask recovery and by 0.1354 over learned-mask recovery. Its full-trajectory NLL-AUC is also lower than both single-mask controls. Meanwhile, its initially identical masks diverge, reaching a mean pairwise Hamming distance of 0.03228 with no duplicate pair at step 1k. PARE remains 0.5733 PPL above the frozen dense source at 9.0277.

The matched diverse entry raises step-zero NLL from 10.39660 to 10.83049 and reaches PPL 9.8102 at 1k steps, compared with PARE's 9.6010. PARE has lower PPL than all three controls at every

Table 11: Single-seed public TinyLlama-1.1B SwiMGLU recovery PPL from a layerwise balanced parent through 1k steps under a 2k learning-rate horizon. PARE and the diverse entry share the four-mask architecture; the latter perturbs three parent masks by quota-preserving ownership exchanges.

| Step | Fixed single-mask | Learned single-mask | PARE | Quota-preserving diverse |
|---|---|---|---|---|
| 0 | 32,748.2042 | 32,748.2042 | 32,748.2042 | 50,538.5612 |
| 50 | 11.8645 | 11.8651 | **11.5593** | 12.4657 |
| 100 | 11.0562 | 11.0561 | **10.8962** | 11.4779 |
| 150 | 10.7710 | 10.7712 | **10.6399** | 11.1269 |
| 200 | 10.5728 | 10.5730 | **10.4462** | 10.8744 |
| 400 | 10.1714 | 10.1716 | **10.0849** | 10.3963 |
| 600 | 9.9689 | 9.9745 | **9.9015** | 10.1591 |
| 800 | 9.8365 | 9.8389 | **9.7439** | 9.9736 |
| 1k | 9.7291 | 9.7364 | **9.6010** | 9.8102 |

recorded post-entry point. The diverse entry's terminal mean pairwise mask Hamming distance is 0.16092, with no duplicate pair.

**Gemma-3-1B-PT GeGLU-to-GeMGLU endpoint.** Under the matched Gemma recovery protocol in Appendix D.3, all three conditions share step-zero uncapped NLL 22.70920. Table 12 reports the subsequent trajectory.

Table 12: Gemma-3-1B-PT fixed-buffer PPL through 1k steps (one seed, 2k learning-rate horizon). PARE leads early and at the endpoint, while fixed-mask recovery is lower at 400–800 steps.

| Step | Fixed single-mask | Learned single-mask | PARE |
|---|---|---|---|
| 50 | 23.8988 | 23.9503 | **19.7013** |
| 100 | 17.5982 | 17.6171 | **17.2106** |
| 150 | 16.2916 | 16.3101 | **16.0964** |
| 200 | 15.7030 | 15.7159 | **15.6169** |
| 400 | **14.7731** | 14.7830 | 14.8340 |
| 600 | **14.3271** | 14.3354 | 14.4149 |
| 800 | **14.0498** | 14.0563 | 14.1032 |
| 1k | 13.8599 | 13.8748 | **13.8145** |

At 1k, PARE improves PPL by 0.0454 over fixed-mask recovery on Gemma. Its full NLL-AUC is lower than both single-mask controls, but fixed-mask recovery has 1.9246 lower post-200 NLL-AUC.

PARE's masks reach a mean pairwise Hamming distance of 0.02937 with no duplicate pair. The frozen source reaches PPL 19.1939, but no dense continuation is available for comparison after the additional 524M recovery tokens.

### E.6 Downstream Evaluation of PARE

We evaluate six downstream tasks in the zero-shot setting using the protocol in Appendix D.5. The six tasks are HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), ARC-Easy and ARC-Challenge (Clark et al., 2018), SciQ (Welbl et al., 2017), and WinoGrande (Sakaguchi et al., 2020). We use normalized accuracy for the first five tasks and accuracy for WinoGrande. The macro score is their unweighted mean.

Four-mask PARE achieves a modest improvement in average zero-shot downstream performance over fixed-mask recovery in the primary 2k layerwise balanced comparison (Table 13), increasing the six-task macro score from 0.4566 to 0.4579 across three recovery seeds. Changes across individual tasks are mixed.

Table 13: Six-task zero-shot results. The dense source is one checkpoint.

| Task | Dense source | Fixed single-mask | PARE |
|---|---|---|---|
| HellaSwag | 0.3158 | $0.3056 \pm 0.0002$ | $0.3084 \pm 0.0013$ |
| PIQA | 0.6224 | $0.6148 \pm 0.0016$ | $0.6150 \pm 0.0013$ |
| ARC-Easy | 0.4545 | $0.4374 \pm 0.0011$ | $0.4401 \pm 0.0019$ |
| ARC-Challenge | 0.2457 | $0.2497 \pm 0.0005$ | $0.2480 \pm 0.0047$ |
| SciQ | 0.6460 | $0.6280 \pm 0.0035$ | $0.6303 \pm 0.0023$ |
| WinoGrande | 0.4886 | $0.5043 \pm 0.0070$ | $0.5057 \pm 0.0033$ |
| Macro | 0.4622 | $0.4566 \pm 0.0012$ | $0.4579 \pm 0.0006$ |

# F Single-Mask Assignment and Recovery Results

This appendix reports first-stage assignment structure and fixed-class initializer recovery at 159M, followed by initializer transfer to TinyLlama.

## F.1 Single-Mask Assignment Structure

Figure 5 extends the single-seed structural ablation in Figure 2 with results at 200 recovery steps. For strict 1:2, magnitude initialization reaches PPL 331.8854 before recovery, 18.8541 at 200 steps, and 16.7121 at 2k steps, while activation-aware initialization reaches 188.4310, 18.7302, and 16.6619, respectively.

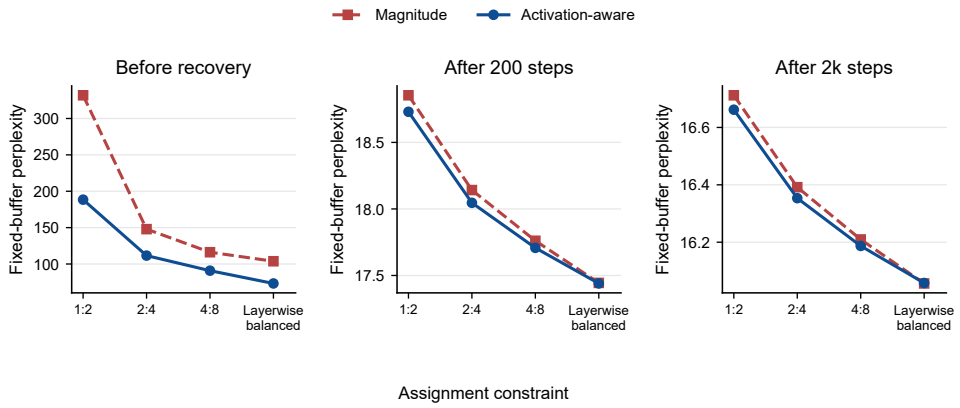


Figure 5: 159M structure–fidelity ablation across assignment constraints. Left to right: PPL before recovery, at 200 steps, and at 2k steps.

## F.2 Fixed-Class Initializer Recovery

Across the short- and long-horizon trajectories in Figure 6, activation-aware initialization retains a small advantage over magnitude. This difference contracts during recovery, while both converted models remain above token-matched dense continuation by a larger margin.

Figure 7 presents 159M excess NLL-AUC savings across recovery windows. Over 0–10B tokens, activation-aware assignment reduces excess NLL-AUC by $6.42\% \pm 0.02\%$ relative to magnitude, with a dense-independent paired difference of $154.36 \pm 0.45$ NLL-steps.

## F.3 Public-Checkpoint Initializer Transfer

Table 14 records the public-checkpoint strict-mask initializer comparison.

Activation-aware assignment starts with 15.17% higher PPL and has 0.2929% higher full-window NLL-AUC, although it is lower from step 50 onward and ends 0.0674 PPL below magnitude.

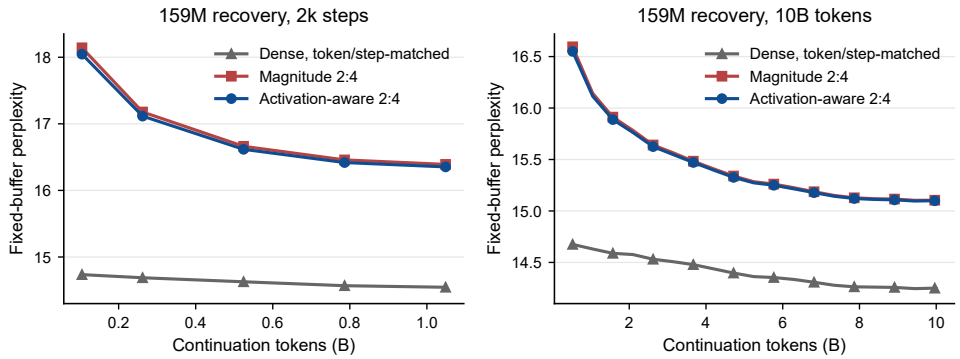


Figure 6: 159M fixed-buffer PPL during recovery. Left: three-seed means for token- and step-matched dense continuation and fixed-mask conversion over 2k steps. Right: three-seed conversion means and a token- and step-matched single-seed dense reference over 10B continuation tokens. Activation-aware remains below magnitude, but both converted models retain a larger architecture gap to dense.

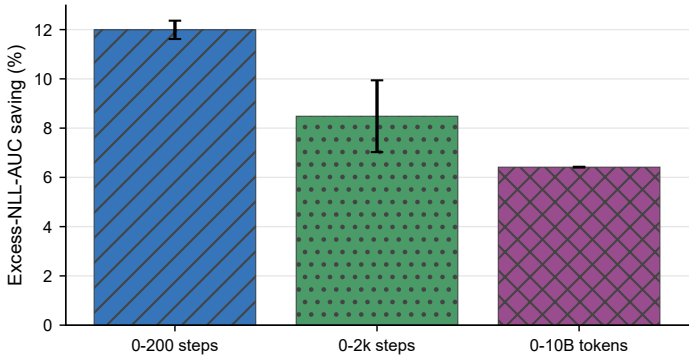


Figure 7: 159M excess-NLL-AUC savings of activation-aware over magnitude 2:4 conversion across recovery windows. Bars and error bars show the mean and sample standard deviation across three conversion seeds. The 0–10B window uses a token- and step-matched single-seed dense trajectory as the common reference.

Table 14: Single-seed public TinyLlama-1.1B strict 2:4 conversion. NLL-AUC is the raw trapezoidal area over steps 0–200.

| Assignment | PPL@0 | PPL@50 | PPL@100 | PPL@150 | PPL@200 | NLL-AUC |
|---|---|---|---|---|---|---|
| Magnitude | **25,783.4136** | 12.3575 | 11.4717 | 11.2079 | 11.1158 | **682.6844** |
| Activation-aware | 29,694.9301 | **12.1901** | **11.3866** | **11.1351** | **11.0484** | 684.6840 |

## G Deployment Measurements

We present four-mask projection and fused-FFN performance, followed by a separate structured single-mask FFN study. Implementation details and measurement protocols are collected in Section G.1.

### G.1 Implementation Details and Measurement Protocols

**Software and common FFN operations.** Both FFN studies use PyTorch 2.5.1. TinyLlama uses CUDA 12.4. The structured single-mask study uses CUDA 11.8, Triton 3.1.0, and PyTorch's bundled CUTLASS backend (Thakkar et al., 2023) (version 3.4.1, commit `bbe579a9`).

All complete FFNs retain a separate dense down projection. In each paired window, the dense GEMM baseline is the fastest of separate gate/value projections, concatenated projections, and concatenated projections with a `torch.compile` SiLU/product epilogue.

**Fused FlashMGLU.** We extend the public MGLU CUDA implementation (Tajima et al., 2025) through a PyTorch C++/CUDA extension. The epilogue-only variant takes the eight gate/value vectors from the four-mask projection and combines SiLU, gate–value multiplication, and route summation in one kernel. The joint projection–epilogue variant keeps the gate/value intermediates within one kernel and writes only the summed hidden vector (Figure 8). It uses split-K one, with one 64-thread block per output row. It loads the shared weights once, accumulates the unmasked dot product and four masked gate dot products in FP32, and obtains each complementary value projection by subtraction:

$$u = xW^\top, \qquad g_i = x(M_i \odot W)^\top, \qquad v_i = u - g_i, \quad i = 1, \ldots, 4.$$

The four route bits are packed into one byte per weight coordinate. After the input-dimension reduction, the kernel computes the nonlinear products and route sum. It preserves the reference's FP16 rounding at the gate/value, SiLU, product, and sequential route-sum stages.

On 524,288 prediction tokens, the recovered TinyLlama PARE checkpoint's fused implementation differs from its FP16 reference by $4.36 \times 10^{-7}$ in NLL.

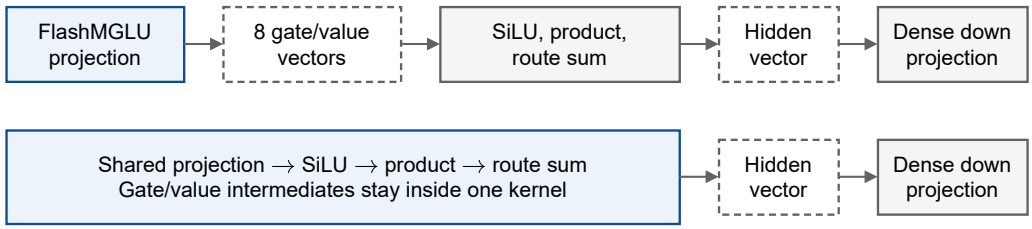


Figure 8: Joint projection–epilogue fusion avoids materializing the eight gate/value vectors of four-mask FlashMGLU. Top: the public projection kernel followed by an eager epilogue. Bottom: joint fusion computes the projections, SiLU, products, and route sum in one kernel, writing only the summed hidden vector before the separate dense down projection. Both paths take the same token, shared weights, and four masks. Blue outlines denote individual kernels, and dashed boxes denote tensors written to memory.

**Token-wise single-mask implementations.** The public FlashMGLU kernel uses split-K one and is called once per token. Its gate/value outputs are stacked before batched SiLU/product and down projection. The local fused variant specializes the joint kernel above to one mask, retaining its 64-thread reduction and FP16 rounding boundaries. It writes one hidden vector per token, and these vectors are stacked before the common batched down projection.

**Batched shared-weight implementations.** The two local Triton kernels store one binary mask bit per weight coordinate and decode the packed mask while loading weight tiles. They compute the gate and value projections with ordinary Tensor Core dot products and FP32 accumulation. One writes the two projection tensors for a separate compiled SiLU/product epilogue. The other applies SiLU/product after the full input-dimension reduction and writes only the hidden activations, preserving the FP16 gate/value, activation, and product rounding boundaries. Both autotune before measurement over the same token/hidden/input tile sizes: $(16, 64, 32)$, $(32, 64, 32)$, and $(64, 64, 32)$, using respectively four, eight, and eight warps and three, four, and four pipeline stages.

2:4 **structured-sparse GEMM.** We form the complementary branch matrices $M \odot W$ and $(1 - M) \odot W$ from the strict single-mask checkpoint and evaluate them with two separate sparse GEMMs (Figure 9). Each branch has two selected coefficients per four consecutive input coordinates and is packed separately into compressed values and position metadata using PyTorch's `to_sparse_semi_structured`, with its CUTLASS backend explicitly selected. At inference, two `torch.nn.functional.linear` calls compute the gate and value projections on NVIDIA

Sparse Tensor Cores (Mishra et al., 2021). SiLU and multiplication follow as separate PyTorch operations before the dense down projection. Weight packing is performed before timing. Fusion reduces intermediate memory traffic and kernel launches, while structured-sparse GEMM exploits the 2:4 pattern to reduce projection arithmetic. Across the tested 159M strict 2:4 FFN configurations, outputs match the same-weight FP32 reference within a relative root-mean-square error of $5.05 \times 10^{-4}$.

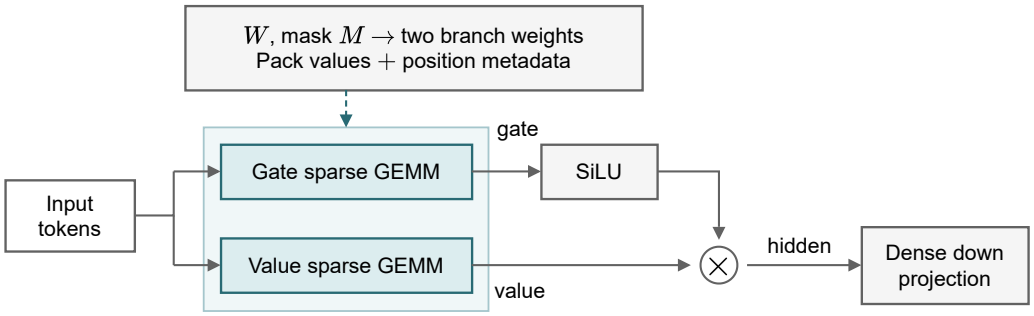


Figure 9: Strict 2:4 single-mask execution uses two separate sparse GEMMs for the complementary gate and value branches. Each branch selects two weights per four consecutive input coordinates. Compressed values and position metadata are packed before timing. At inference, the two projections are followed by separate SiLU and multiplication operations and a dense down projection.

**Timing and aggregation.** All latency comparisons use three independent runs with paired inputs. Unless specified otherwise, each workload uses 10 warmup calls followed by five timing windows, each enclosing 100 consecutive calls in one CUDA-event interval. Latency per call is the interval duration divided by 100. Provided ranges span the three run-level medians of paired latency ratios. Hardware, precision, and workloads accompany the corresponding results. For projection-only measurements, method order is randomized within each window, and each run is summarized by the median of its five paired ratios.

**Projection weight-address reuse.** The split-K-one comparison uses eager FP16 execution on RTX 4090 and one or eight copies of each method's weight tensors at distinct addresses. Copies have identical values and share one input, with FlashMGLU masks copied alongside the weights. Each window measures 100 cycles through the copies after 10 warmup cycles, normalized per projection. The dense reference is the faster implementation within each paired window. Separate Nsight Compute (v2023.1) profiles collect counters once per condition using application replay without profiler cache flushing or clock locking.

**TinyLlama FFN measurements.** Each process measures all 22 layers with five paired windows per layer. Each run is summarized by the median of 110 paired layer/window ratios. In a separate weight-access comparison, each timed interval covers 100 cycles through all 22 FFNs in eager mode, using a fixed input for each layer. Layer outputs are not passed to subsequent layers. After 10 warmup cycles, each run is summarized by the median of five paired window ratios.

**Structured single-mask FFN measurements.** Inputs are shared across implementations within each run and token count, with entries sampled from $\mathcal{N}(0, 0.25^2)$. Method order rotates across windows, and each run is summarized by the median of five paired latency ratios per workload. Measurements use continuous-call timing without an explicit cache sweep. Packing, compilation, and autotuning are excluded, while intermediate allocation and token-wise output stacking are timed.

### G.2 Four-Mask Projection Kernels

We benchmark only the gate and value projections of four-mask MGLU using the CUDA kernels from its public implementation (Tajima et al., 2025). We implement two dense references: separate gate/value linear calls and a single call with pre-concatenated weights.

Measurements use the public benchmark timing function on an RTX 4090 with FP16, one input vector, and weight-matrix sizes $h \times d$ of $8192 \times 2048$ and $14336 \times 4096$. Timing and aggregation protocols are collected in Section G.1. Split-K denotes the number of parallel partitions along the reduction dimension. Table 15 presents results for split-K values $1, 2, 4, 8$.

Table 15: Four-mask projection speedups on RTX 4090, expressed as dense latency divided by MGLU-kernel latency. Values above one indicate faster MGLU execution.

| Projection size | Split-K | vs. two dense calls | vs. concatenated dense |
|---|---|---|---|
| $8192 \times 2048$ | 1 | 3.291–3.304 | 3.159–3.178 |
| | 2 | 2.928–2.945 | 2.811–2.827 |
| | 4 | 2.321–2.324 | 2.228–2.232 |
| | 8 | 1.595–1.600 | 1.532–1.536 |
| $14336 \times 4096$ | 1 | 1.281–1.282 | 1.267–1.269 |
| | 2 | 1.268–1.271 | 1.254–1.256 |
| | 4 | 1.303–1.304 | 1.289–1.290 |
| | 8 | 1.304–1.307 | 1.287–1.292 |

The four-mask kernel is faster than both dense references at every tested matrix size and split-K. At $8192 \times 2048$, the speedup decreases as split-K increases, whereas the larger $14336 \times 4096$ case remains within roughly $1.25$–$1.31\times$ across the tested settings. Increasing split-K shortens each thread block's reduction segment but increases the number of block-level reductions and atomic updates to the eight projection outputs. These overheads can offset the benefit of shared weight loading, particularly when shorter segments provide less work over which to amortize them.

**Effect of cross-call weight reuse.** To examine the size-dependent speedup, we repeat split-K-one timing with one reused weight set or eight copies at distinct addresses (Table 16). Cycling through eight sets substantially reduces the smaller configuration's speedup, while the larger configuration changes little. In separate profiles, the smaller FlashMGLU projection kernel has a near-100% L2 hit rate when one weight set is reused, whereas the dense references still read approximately 64 MiB from DRAM per call. The drop in cache hits with address rotation supports cache residency across calls as an important contributor to the smaller configuration's $3\times$ speedup.

Table 16: Effect of cross-call weight reuse on split-K-one four-mask FlashMGLU projection. Speedup is the faster dense reference's latency divided by FlashMGLU latency. L2 hit-rate ranges span projection-kernel calls within one separate profile per condition.

| Projection size | Weight sets | Speedup | FlashMGLU L2 hit (%) |
|---|---|---|---|
| $8192 \times 2048$ | 1 | 3.1320–3.1586 | 100.00 |
| | 8 | 1.2147–1.2192 | 5.82–5.97 |
| $14336 \times 4096$ | 1 | 1.2633–1.2678 | 4.57 |
| | 8 | 1.2505–1.2673 | 4.49–4.57 |

### G.3 Fused FFN Measurements

We evaluate the four-mask TinyLlama-1.1B PARE checkpoint after 1k recovery steps, with projection size $h \times d = 5632 \times 2048$ and FP16 weights and activations on an RTX 4090.

**Complete-FFN measurements.** We measure the complete FFN at one input token using continuous repeated execution of the same layer and weight addresses. The dense reference uses the original pretrained TinyLlama weights, with implementations selected under the protocol in Section G.1.

Table 17 compares local epilogue fusion and joint projection–epilogue fusion with this dense reference and with the FlashMGLU projection with an eager epilogue. The latter achieves $1.1717$–$1.1804\times$ speedup over the dense source FFN. Our joint projection–epilogue fusion reaches $1.9892$–$1.9898\times$ the same reference's speed.

In the separate 22-layer rotation comparison on the same GPU, joint fusion achieves 1.2136–1.2172× speedup over the fastest of the same three dense implementations. The smaller gain under layer rotation indicates that the larger continuous-call speedup depends on the weight-access pattern.

Table 17: Complete-FFN speedups for the recovered TinyLlama checkpoint: comparator latency divided by local-implementation latency under continuous-call timing. The dense comparator uses the original source weights, while the FlashMGLU and local MGLU implementations use the recovered PARE weights. The FlashMGLU comparator uses an eager epilogue, and all measurements include the down projection.

| Local implementation | Comparator | Continuous-call speedup |
|---|---|---|
| Fused epilogue | Dense source FFN | 1.6831–1.6859 |
| Fused projection + epilogue | Dense source FFN | 1.9892–1.9898 |
| Fused epilogue | FlashMGLU projection + eager epilogue | 1.4313–1.4403 |
| Fused projection + epilogue | FlashMGLU projection + eager epilogue | 1.6896–1.6999 |

### G.4 Structured Single-Mask FFN Execution

We compare six implementations of the first FFN ($h \times d = 3072 \times 768$) of the strict 2:4 single-mask 159M checkpoint after 10B recovery tokens, using FP16 on an RTX 4090. Table 18 provides CUTLASS structured-sparse GEMM speedups over batched dense GEMM and the batched and token-wise shared-weight MGLU implementations. All six execute the same recovered checkpoint, including the dense GEMM baseline, whereas Section G.3 uses the dense source as its reference. Section G.1 gives the implementation and timing protocols.

Table 18: Complete-FFN speedups of two-call CUTLASS 2:4 structured-sparse GEMM over five implementations of the same 159M strict single-mask checkpoint, expressed as comparator latency divided by CUTLASS latency for $T$ input tokens. Values above one favor CUTLASS. Dense GEMM materializes both branch matrices from this checkpoint's shared weights and mask, whereas both local batched MGLU implementations retain shared weights and packed masks. The official token-wise implementation uses FlashMGLU. All implementations include the down projection.

| Tokens $T$ | Dense GEMM | Batched MGLU | | Token-wise MGLU | |
|---|---|---|---|---|---|
| | | Separate epilogue | Jointly fused | FlashMGLU | Locally fused |
| 1 | 0.278–0.291 | 0.964–0.995 | 0.426–0.432 | 0.407–0.440 | 0.176–0.197 |
| 32 | 0.402–0.415 | 1.274–1.337 | 0.658–0.697 | 5.432–5.759 | 3.052–3.179 |
| 128 | 0.544–0.590 | 1.275–1.347 | 0.948–1.012 | 20.960–22.317 | 11.638–12.235 |
| 512 | 1.215–1.342 | 2.258–2.272 | 2.204–2.265 | 76.134–78.673 | 41.276–41.509 |
| 1024 | 1.100–1.199 | 2.132–2.554 | 2.283–2.287 | 76.076–86.826 | 42.090–45.425 |
| 2048 | 1.186–1.277 | 2.677–2.909 | 2.599–2.612 | 105.229–119.224 | 55.846–61.377 |

At 512–2048 tokens, CUTLASS is faster than all three batched comparators in each of the three runs (Table 18), reaching 1.10–1.34× speedup over dense GEMM and 2.20–2.61× over jointly fused batched MGLU. Joint fusion reduces the separate-epilogue implementation's latency most at small token counts. At 1 and 32 tokens, jointly fused batched MGLU is faster than CUTLASS. At 128 tokens their run-level ratios straddle one.

At one token, the locally fused token-wise implementation has the lowest median latency among the compared implementations, with 1.47–1.56× speedup over dense GEMM. CUTLASS is faster than both token-wise implementations at all tested $T \geq 32$. Their larger ratios include repeated kernel launches and output stacking.

## H LIMITATIONS AND DISCUSSION

**Entry-path attribution.** The entry-path comparisons cover a 10% ownership-exchange rate. Varying this rate would clarify sensitivity to perturbation strength. The reconstruction control reduces initial NLL error while changing shared weights, leaving the effects of residual function mismatch, mask geometry, gradient alignment, and parameter displacement coupled. The tied-logit comparison uses a strict $2{:}4$ parent and changes the optimizer-state representation. Isolating the role of mask differentiation under layerwise balanced initialization therefore requires further controlled comparisons.

**Structured route expansion.** Future work will study $2{:}4$-constrained PARE with Sparse Tensor Core kernels that retain weight sharing across routes. This would assess recovery under structured masks and the cost of executing their distinct sparse layouts.