LEARNING BALANCED SPARSE ROUTING WITH UNIFORM PAIRWISE DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse Mixture-of-Experts (MoE) layers activate only a small subset of experts per token, so practical runtime efficiency heavily depends on how evenly the router distributes token-level demand. Skewed demand creates overloaded hot experts, underutilized experts, rank imbalance, and extra runtime repair overhead. Existing approaches either apply indirect global balancing losses, or only perform ephemeral runtime corrections. We introduce Uniform Pairwise Distillation (UPD), a lightweight local self-distillation learning objective for training better-balanced router proposals. For every route slot modified by the strict capacity-aware execution policy during training, UPD constructs one sparse pairwise ranking target that favours the accepted replacement expert over the displaced expert. It assigns equal weight to each observed correction pair, reuses the router's own logits as student signals, and introduces neither an external teacher network nor additional trainable parameters. Across matched from-scratch MoE configurations ranging from 350M to 7B total parameters, UPD reduces raw-demand imbalance and runtime reassignment work while maintaining perplexity comparable to the no-distillation baseline. Improvements in routing behaviour transfer to held-out in-domain and out-of-domain test inputs. In a two-rank expert-parallel pipeline, these learned routing gains translate to higher prefill throughput and modest, repeatable decode improvements, while the original execution policy continues to enforce hard capacity feasibility guarantees.

1 INTRODUCTION

Sparse mixture-of-experts (MoE) layers activate only a small subset of experts per token, so their real-world efficiency heavily depends on how uniformly the router distributes token-level demand (Du et al., 2022; Jiang et al., 2024; Muennighoff et al., 2025). Under token-choice routing, every token scores experts independently. As a result, numerous tokens may gravitate toward the same popular experts while other experts stay largely idle. Such concentrated demand reduces expert utilization, raises dispatch and all-to-all communication latency, and can even degrade model quality when oversubscribed expert slots cannot be fully satisfied. These adverse effects manifest in both the router's raw proposed demand and the subsequent work carried out by the execution runtime.

Existing load-balancing approaches for MoE routing largely fall into two distinct families. The first family leverages global training-time signals, including Switch/GShard-style auxiliary losses and the expert-wise bias updates used in Loss-Free Balancing (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Wang et al., 2024). These objectives drive aggregate expert usage statistics toward a desired target, yet they cannot pinpoint which token-level local routing decision originally caused a capacity conflict. They also impose supervision over numerous routing choices that are already valid and feasible, introducing unnecessary training signal overhead. The second family modifies assignment or runtime-execution logic, encompassing balanced assignment, expert-choice routing, score-guided constrained selection, and capacity-aware rerouting schemes (Lewis et al., 2021; Kool et al., 2021; Zhou et al., 2022; Dong et al., 2025; He et al., 2026; Li et al., 2026). Methods within this family guarantee feasibility for the current batch, yet their corrections are typically ephemeral: the underlying router receives no feedback about which future routing decisions ought to be adjusted. Taken together, global training-time signals suffer from indirect supervision, while runtime-only corrections produce improvements that do not persist to future batches.

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.

We address this gap with Uniform Pairwise Distillation (UPD), a lightweight local self-distillation objective tailored for the MoE router. During standard training forward passes, each expert is subject to a hard capacity limit. It can process only a fixed maximum number of tokens per batch to avoid compute overload. When router proposals exceed this bound, some router-suggested experts must be replaced with other feasible experts. For every such replaced route slot, we construct a sparse pairwise ranking label that favors the accepted replacement expert over the originally displaced expert. We detach these labels from the execution graph. The same router produces the raw logits and serves as the student, and we assign equal weight to every observed swap pair. These local comparison signals train the router to steer away from poor routing choices in future batches, while the pre-existing execution policy continues to enforce feasibility. UPD therefore provides direct token-level supervision without a separate teacher network or trainable parameters.

We investigate this objective on from-scratch MoE models supporting both top-2 and top-8 routing, across 350M, 1B, and 7B-total-parameter configurations. Under frozen held-out in-domain and out-of-domain inputs, UPD mitigates imbalance in router-proposed demand and lowers the volume of runtime execution corrections. We further measure a two-rank expert-parallel pipeline using a route-equivalent optimized implementation. This setup isolates performance gains from improved router proposals apart from overhead inherent to the fixed assignment policy.

We make three contributions.

- We introduce a lightweight local self-distillation signal for MoE routing. Every observed changed slot yields one equally weighted pairwise ranking target, and our approach requires neither unaffected-slot supervision nor an external teacher model.
- We demonstrate that this sparse supervision signal improves router- proposed demand balance and reduces runtime feasibility corrections under strict execution. These gains hold across model scales, routing arities, random seeds, and held-out test data, while language-model quality remains comparable to the matched no-distillation baseline within our scratch budgets.
- We establish the practical system-level implication of better routing proposals for expert-parallel execution. Using an identical route-equivalent solver and fixed hardware, reduced recurring feasibility-repair work yields higher prefill throughput together with a modest yet repeatable decode improvement.

2 Related Work

Global training signals. Sparse MoEs often adopt auxiliary objectives to steer aggregate router statistics toward target expert-usage distributions (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). Loss-Free Balancing adjusts expert-wise biases from batch statistics without auxiliary loss gradients (Wang et al., 2024). Though these methods impose useful global balancing pressure, aggregate statistics cannot pinpoint the token-level routing choices that trigger capacity conflicts. Unlike these global signals, UPD introduces local pairwise targets derived directly from per-slot execution corrections.

Assignment and execution rules. Balanced assignment, Expert Choice, MaxScore Routing, and capacity-aware overflow schemes govern how congested batches are resolved (Lewis et al., 2021; Kool et al., 2021; Zhou et al., 2022; Dong et al., 2025; Fedus et al., 2022; He et al., 2026; Li et al., 2026). They guarantee feasibility for the current batch, yet provide no persistent training feedback to adjust future router behaviour. UPD keeps the existing execution policy unchanged and treats its runtime decisions as detached supervision labels for router updates. MaxScore* applies differentiable constrained selection at assignment time but does not inherently teach the router to avoid conflicts in subsequent batches.

Distillation and MoE systems. StableMoE and teacher-guided routing rely on an extra router or external teacher module (Dai et al., 2022; Kada et al., 2026). By contrast, UPD leverages the same router for both teacher-signal generation and student learning, preserving content-aware routing and only modifying locally-affected logits. Work including DeepSpeed-MoE, Tutel, and MegaBlocks optimizes dispatch and expert computation kernels (Rajbhandari et al., 2022; Hwang et al., 2023; Gale et al., 2023). UPD alters the input demand for these existing systems instead of replacing their

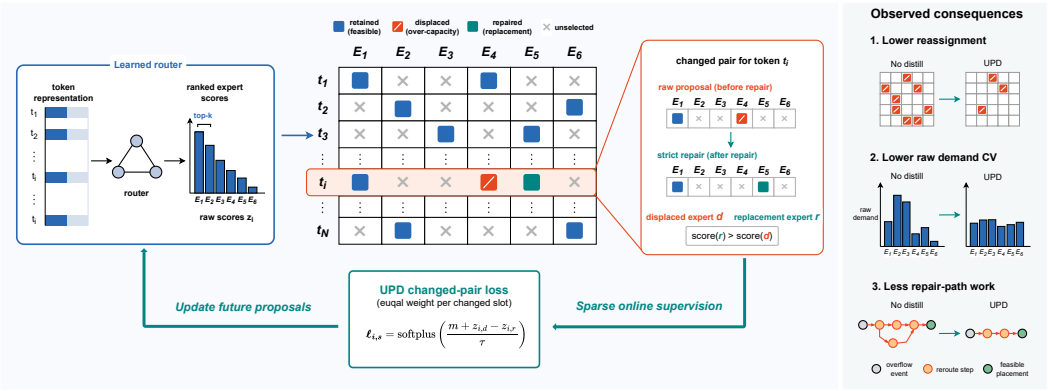


Figure 1: Uniform Pairwise Distillation converts execution-aligned route adjustments into sparse online self-distillation. Every modified route slot provides one equally weighted ranking constraint. The original strict execution policy remains in place and continues to enforce feasibility guarantees.

underlying kernels. Our two-rank system measurements isolate the routing-learning benefit from pure solver-implementation overhead.

3 Uniform Pairwise Distillation

UPD is a router-level learning objective for improving demand balance. It adds sparse pairwise self-distillation labels to the normal token-choice forward pass. The current execution decision is used only as a source of local labels; UPD does not replace the execution policy or add a second routing mechanism. We first describe how these labels are exposed, then define the loss and its cost.

Our learning procedure has three stages. Section 3.1 obtains local labels from the current execution decision. Section 3.2 applies an equally weighted pairwise ranking loss to the implicated logits. Section 3.3 uses the resulting update to improve future router proposals. Figure 1 summarizes this learning flow. Section 3.4 describes a separate route-equivalent solver optimization that reduces execution overhead without changing any routing decision; it is an execution optimization, not a fourth learning stage.

3.1 Execution-aligned sparse labels

Consider a sparse layer with N valid tokens, E experts, and top-k routing. Here $i \in \{1, \ldots, N\}$ indexes valid tokens, $e \in \{1, \ldots, E\}$ indexes experts, and $s \in \{1, \ldots, k\}$ indexes a token's route slots. The router produces a logit vector $z_i \in \mathbb{R}^E$, with $z_{i,e}$ denoting its component for expert e, and provisional route slots $a_{i,s} \in \{1, \ldots, E\}$. Under the strict execution policy used in our experiments, each expert accepts at most

$$C = \left\lceil \frac{\phi N k}{E} \right\rceil \tag{1}$$

routes, where ϕ is the capacity factor. The execution path obtains a feasible assignment $\tilde{a}$ by retaining high-scoring proposals and completing the remaining slots with available candidates. If a greedy completion reaches a within-token uniqueness dead end, the implementation uses an augmenting repair. This standard capacity-aware execution step is related to iterative rerouting (Fedus et al., 2022; Dong et al., 2025); it supplies local labels for UPD but is not itself the learning contribution. The output $\tilde{a}_{i,s} \in \{1, \ldots, E\}$ contains no duplicate expert within a token and assigns at most C routes to each expert. This feasible assignment executes the current forward pass and defines the correction pairs used below.

The raw assignment a records the router's learned demand, whereas $\tilde{a}$ is the assignment executed by the experts. We measure the raw demand coefficient of variation (CV) from the per-expert counts in a and reassignment as $\sum_{i,s} \mathbf{1}[a_{i,s} \neq \tilde{a}_{i,s}]/(Nk)$. With the execution policy fixed, these metrics

separate the router's demand from how often the system must correct it. The complete solver contract and metric definitions are given in Sections A and B.

3.2 SPARSE PAIRWISE SELF-DISTILLATION

Each changed slot yields one local pairwise label

$$\mathcal{P} = \{(i, s, d, r) : d = a_{i,s},\ r = \tilde{a}_{i,s},\ d \neq r\}, \tag{2}$$

where d is the displaced expert and r the accepted replacement. $z_{i,d}$ and $z_{i,r}$ denote the router logits of token i for the displaced expert d and replacement expert r, respectively. The current execution result acts as an online teacher, but its discrete indices are detached. UPD applies the smooth pairwise margin loss

$$\ell_{i,s} = \mathrm{softplus}\left(\frac{m + z_{i,d} - z_{i,r}}{\tau}\right), \qquad \mathcal{L}_{\mathrm{UPD}} = \frac{1}{|\mathcal{P}|} \sum_{(i,s,d,r)\in\mathcal{P}} \ell_{i,s}. \tag{3}$$

Here m is the margin and $\tau > 0$ is the temperature. We use $\mathrm{softplus}(x) = \log(1 + \exp x)$ and define $\mathcal{L}_{\mathrm{UPD}} = 0$ when $|\mathcal{P}| = 0$, so a batch with no changed slots contributes no distillation update. Minimizing Eq. (3) raises $z_{i,r}$ relative to $z_{i,d}$; the softplus is a smooth ranking surrogate whose gradient decreases once the desired margin is satisfied. Teacher indices are detached, and the loss is zero when no slot changes. A top-2 token therefore contributes zero, one, or two pairs. Unchanged slots receive no target.

The unit of supervision is a changed *slot*. Suppose a token proposes (e_2, e_5) and the feasible execution uses (e_2, e_7). UPD adds only the comparison $e_7 \succ e_5$. The retained choice e_2 receives no target, even though it belongs to a changed token. This matters because feasibility depends on the batch: another set of tokens can leave capacity for e_5. Restricting the target to the observed pair avoids reinforcing every already-feasible route in that batch. The objective ablation in Table 3 tests this choice against supervision over full expert sets and changed tokens.

Let $q = (m + z_{i,d} - z_{i,r})/\tau$. A changed pair contributes gradients $\partial\ell/\partial z_{i,d} = \sigma(q)/\tau$ and $\partial\ell/\partial z_{i,r} = -\sigma(q)/\tau$. At the logit level, the loss acts on these two entries and gives a larger gradient when the replacement scores poorly relative to the displaced expert. Shared model parameters can also change other logits after an update. Every pair has the same loss weight; its gradient still depends on the score gap. Here σ denotes the logistic sigmoid, $\sigma(x) = 1/(1 + \exp(-x))$. "Uniform" refers to the equal loss weight assigned to every observed pair; it imposes no target of equal expert probabilities or token counts.

3.3 TRAINING OBJECTIVE AND OPTIMIZATION

For scratch training, the total objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \lambda_{\mathrm{UPD}}\mathcal{L}_{\mathrm{UPD}} + \lambda_z\mathcal{L}_z. \tag{4}$$

Here $\mathcal{L}_{\mathrm{LM}}$ is the language-model loss, $\mathcal{L}_z$ is the router z-loss, and λ_{UPD} and λ_z are scalar coefficients. The trainable parameter vector is θ. The no-distillation baseline sets $\lambda_{\mathrm{UPD}} = 0$ while retaining the same strict execution. The current forward pass supplies the teacher indices. They are detached, so backpropagation follows the gathered logits without differentiating through the discrete rerouting decisions. Hyperparameters and scheduling are specified in Sections A.1 and 4. In the algorithm, B denotes the current minibatch and $\mathrm{reroute}$ is the fixed capacity-feasible execution operator described above.

At evaluation, the *strict-routing view* denotes the complete operational path. We also evaluate raw top-2 without rerouting to diagnose what the router internalized; raw top-2 has no hard capacity guarantee and is not a deployment baseline.

The additional loss computation is small and explicit: two gathered logits and one scalar softplus per changed slot, followed by a mean over pairs. Once the pair set is available, this takes $O(|\mathcal{P}|)$ work and storage, compared with $O(NE)$ entries for a dense distribution target. Constructing the pair set also requires comparing the Nk raw and repaired slots. UPD adds no trainable parameters or separate teacher forward pass.

Algorithm 1 Route-equivalent strict rerouting

Require: Logits z, proposals a, capacity C; expert ranks $\rho_i(e) = \mathrm{rank}_{\downarrow}(z_{i,e})$
$S_e \leftarrow \{(i,s) : a_{i,s} = e\}$, $R_e \leftarrow \mathrm{TopC}_{(i,s)\in S_e} z_{i,e}$
$\tilde{a}_{i,s} \leftarrow e$ for $(i,s) \in R_e$; $V \leftarrow (\{1{:}N\} \times \{1{:}k\}) \setminus \bigcup_e R_e$
$O_e \leftarrow \{(i,s) : \tilde{a}_{i,s} = e\}$, $A_i \leftarrow \{\tilde{a}_{i,s} : \tilde{a}_{i,s} \text{ assigned}\}$
for all $v = (i,s) \in V$ in fixed order **do**
    $F_v \leftarrow \{e : |O_e| < C \ \wedge \ e \notin A_i\}$
    if $F_v \neq \varnothing$ **then**
        $e^\star \leftarrow \arg\min_{e\in F_v} \rho_i(e)$; $\Pi \leftarrow \varnothing$
    else
        $(e^\star, \Pi) \leftarrow \textsc{Augment}(v, \rho, O, A, C)$
    end if
    $(\tilde{a}, O, A) \leftarrow \textsc{Apply}(v, e^\star, \Pi, \tilde{a}, O, A)$
end for
Ensure: $|O_e| \le C$ and $|A_i| = k$; return $\tilde{a}$

This cost accounts for the supervision added to a strict-routing pipeline; training still pays for feasibility enforcement and backpropagation. At inference, the loss is absent and the execution policy remains the capacity guarantee. Any speed gain therefore comes from the learned proposals reducing the work of that execution path. The loss can consume aligned labels from a top-k feasibility operator without access to its search internals. Our experiments test this interface across model sizes and route-equivalent solver implementations.

3.4 Route-equivalent solver optimization

The original strict rerouting implementation reconstructs and sorts an expert's occupant list whenever an augmenting search visits that expert. This preserves the intended assignment but creates repeated host-side work. Our optimized implementation precomputes each token's expert order and maintains a score-ordered occupant index per expert. It changes only how the same candidates are retrieved: candidate order, occupant order, recursive moves, and backtracking are unchanged (Algorithm 1).

In Algorithm 1, e indexes experts and $\rho_i(e)$ is the rank of expert e in token i's descending logit order. S_e is the set of raw slots proposing expert e, and R_e is the retained top-C subset under the TopC operator. V contains unresolved slots, O_e is the current occupant set of expert e, A_i is the set of experts already assigned to token i, and F_v is the feasible candidate set for slot v. The selected expert is $e^\star$; Π denotes an augmenting path returned by AUGMENT and applied by APPLY. The symbol Π is distinct from the pair set $\mathcal{P}$ used by UPD's training loss.

We accept the optimized solver only when every intercepted call has bitwise-equal route indices and weights and identical capacity metadata; full-test PPL and generated-token hashes must also match. The original solver is retained as a mechanism audit, while the optimized solver supports the primary systems claim. Optimizing the solver makes each feasibility correction cheaper. Holding its route choices fixed lets us test whether learning better proposals still helps under this faster implementation. Each timing comparison then uses the same solver and pipeline for both checkpoints.

4 Experimental Setup

Models and data. Our primary model is a 973M-parameter scratch Qwen2-MoE-style causal LM (Yang et al., 2024) with eight routed experts and top-2 routing. We train it on 40.0M WikiText-103 (Merity et al., 2016) tokens at capacity factor 1.0. The no-distillation baseline and UPD share initialization, data order, optimizer, and strict execution for seeds 52–54. We evaluate the frozen WikiText-103 test split, 2048 nonempty C4 validation documents (Raffel et al., 2020), and the complete LAMBADA OpenAI test text (Paperno et al., 2016). Matched 350M and randomly initialized OLMoE-7B-total scratch studies provide scale evidence. Architectures, schedules, and data hashes appear in Sections A.1 and B.

Table 1: Frozen 1B WikiText-103 test results over seeds 52–54. Entries are absolute means ± population standard deviations. Native rows use the method's strict execution view; raw rows are diagnostic top-2 evaluations. Bold marks the lowest mean in each column within the strict or raw view; underlining marks the second-lowest PPL in each view.

Method / view	PPL ↓	Reassign. ↓	Demand CV ↓
No distill.	35.047 ± 0.078	$11.177 \pm 0.299\%$	0.11875 ± 0.00694
Aux	35.084 ± 0.132	$11.058 \pm 0.093\%$	0.11766 ± 0.00472
LFB	$\mathbf{34.870 \pm 0.096}$	$9.959 \pm 0.399\%$	0.59255 ± 0.01629
MaxScore*	35.783 ± 0.306	$18.710 \pm 0.632\%$	0.16724 ± 0.01255
UPD	$\underline{35.042 \pm 0.165}$	$\mathbf{4.173 \pm 0.053\%}$	$\mathbf{0.01628 \pm 0.00073}$
No distill. raw	$\mathbf{34.981 \pm 0.082}$	—	0.11803 ± 0.00666
UPD raw	$\underline{35.031 \pm 0.169}$	—	$\mathbf{0.01646 \pm 0.00080}$

Baselines and metrics. We compare with a Switch/GShard-style auxiliary loss (Aux), Loss-Free Balancing (LFB) (Wang et al., 2024), and a paper-aligned MaxScore* implementation (Dong et al., 2025). Native execution is the operational comparison; matched strict rerouting and raw top-2 views separate runtime assignment from learned router demand. We report causal-LM perplexity (PPL), raw demand coefficient of variation (CV), reassignment, and capacity violations as means with population standard deviations over three seeds. Frozen manifests, definitions, and view contracts are in Sections B, E.1 and E.2.

Systems. System experiments use one host with two A800 80GB GPUs and two expert-parallel ranks. We compare the original and a route-equivalent optimized rerouting implementation only after exact route, metric, and generated-token checks. The optimized decode result uses six balanced timing groups with three repeats per checkpoint and reversed method order. We report time to first token (TTFT) and the 95th-percentile latency (p95) in addition to throughput. Workload and instrumentation details are in Section D.

5 Results

5.1 Primary 1B evidence

Table 1 keeps the complete frozen WikiText-103 comparison in one place. It reports absolute means and population standard deviations for all five native methods, with raw-top-2 diagnostics for the no-distillation baseline and UPD. Relative effects are stated in the text and detailed in the appendix. Against the same-seed no-distillation baseline, UPD reduces validation reassignment by $63.644 \pm 0.651\%$ and raw demand CV by $84.731 \pm 2.004\%$, with a $-0.021 \pm 0.363\%$ PPL change. On the separately frozen test, mean reassignment falls from 11.177% to 4.173% and demand CV from 0.11875 to 0.01628, corresponding to reductions of $62.655 \pm 0.565\%$ and $86.272 \pm 0.510\%$.

The effect is consistent across the three test seeds: reassignment reductions are 63.191%, 61.874%, and 62.901%, while demand-CV reductions are 86.782%, 86.457%, and 85.575%. PPL deltas instead straddle zero (-0.115%, $+0.363\%$, and -0.292%), so the load improvement has a clearer direction than the small quality differences. Both strict methods have zero post-correction capacity violations.

The raw-top-2 rows show that the router itself learns more balanced demand. With execution correction disabled, demand CV falls from 0.11803 to 0.01646, an $86.044 \pm 0.508\%$ reduction, while raw PPL changes by only $+0.142 \pm 0.283\%$. Raw top-2 is diagnostic rather than a deployment policy because it does not enforce a feasibility guarantee.

The external baselines expose the distinction between global and local signals. UPD lowers native reassignment and demand CV relative to Aux, LFB, and MaxScore* at every seed. Its mean PPL is 0.120% lower than Aux and 2.059% lower than MaxScore*; LFB has 0.494% lower PPL but a much larger raw demand CV (0.59255 versus 0.01628). Under common strict rerouting, UPD reduces reassignment by 84.835% versus LFB and 69.682% versus MaxScore*. The full native, matched, and raw external views remain in Table 15. Figure 2 gives a layer-level view of the same routing

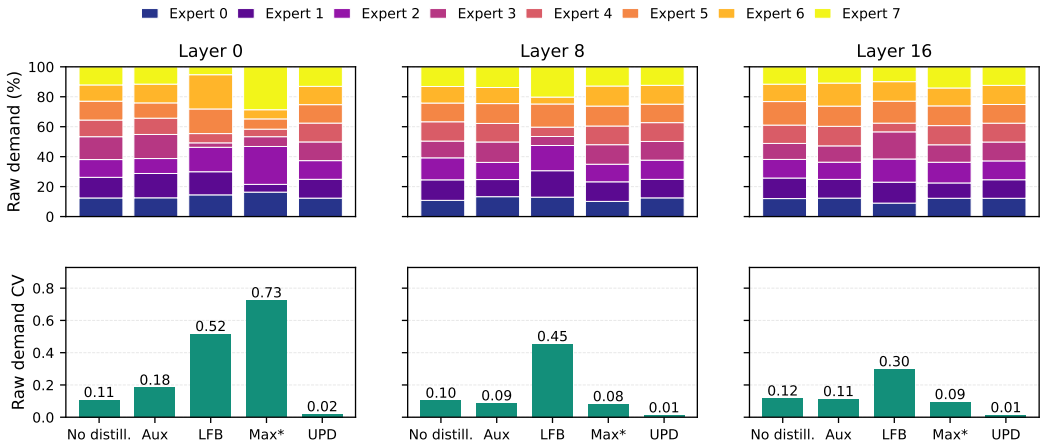


Figure 2: Per-expert raw demand and demand CV in zero-based MoE layers 0, 8, and 16 on the frozen WikiText-103 test split for seed 52. In each layer group, the upper panel shows the percentage of pre-capacity-repair top-2 assignments received by each expert, and the lower panel shows the corresponding CV computed from the same counts.

behavior for seed 52. The figure is illustrative rather than a substitute for the three-seed aggregate, but its layer-wise pattern is consistent with the measured reduction in proposed demand imbalance.

Table 2: Frozen 1B OOD results over seeds 52–54. Entries are absolute means ± population standard deviations under strict execution. Bold marks the lowest mean in each column within each dataset; underlining marks the second-lowest PPL. C4 contains the first 2048 nonempty English validation documents, and LAMBADA is causal-LM PPL over complete test text. Standard deviations are population values over the three seeds.

Data	Method / view	PPL ↓	Reassign. ↓	Demand CV ↓
C4	No distill.	1297.824 ± 50.572	15.535 ± 0.229%	0.33923 ± 0.00516
C4	Aux	1268.272 ± 68.425	14.854 ± 0.291%	0.31241 ± 0.01007
C4	LFB	1286.425 ± 59.242	14.880 ± 0.309%	0.65790 ± 0.00343
C4	MaxScore*	**1261.218 ± 26.360**	26.005 ± 0.534%	0.44024 ± 0.01247
C4	UPD	1275.681 ± 70.217	**9.381 ± 0.144%**	**0.19962 ± 0.00430**
LAMBADA	No distill.	2717.851 ± 119.879	19.528 ± 0.814%	0.46161 ± 0.02068
LAMBADA	Aux	2621.832 ± 170.483	18.422 ± 0.912%	0.43321 ± 0.02053
LAMBADA	LFB	2679.196 ± 96.407	17.869 ± 0.799%	0.71638 ± 0.01111
LAMBADA	MaxScore*	**2597.684 ± 125.360**	31.880 ± 1.374%	0.57625 ± 0.03578
LAMBADA	UPD	2620.234 ± 165.942	**11.104 ± 0.536%**	**0.25956 ± 0.01680**

5.2 Frozen out-of-domain evaluation

To test whether the learned signal transfers beyond the training distribution, we evaluate frozen C4 and LAMBADA inputs. Table 2 reports the results. Relative to the no-distillation baseline, C4 reassignment and demand CV fall by $39.605 \pm 0.907\%$ and $41.142 \pm 1.545\%$; on LAMBADA they fall by $43.041 \pm 3.568\%$ and $43.601 \pm 5.084\%$. Every seed improves both load metrics on both datasets. Mean PPL changes by -1.767% on C4 and -3.674% on LAMBADA, with larger between-seed variation than in-domain.

The OOD gains are smaller than the in-domain $62.655\%/86.272\%$ reassignment/CV effects, but the direction is consistent across all six seed–dataset cases. Raw-top-2 CV reductions of 40.344% and 42.278% show that the router proposes more balanced routes on these new inputs as well. The favorable load direction also holds against Aux, LFB, and MaxScore* on both datasets, although PPL ordering varies by reference; the complete matched and raw views are in Table 16.

Table 3: Supervision-target granularity ablation at seed 42 after 1221 steps. All conditions use the same strict rerouting execution.

Objective	PPL ↓	Reassign. ↓	Δ reassign.	Demand CV ↓	Δ CV
No distill.	126.7830	11.116%	—	0.190549	—
Full-set CE	124.4588	8.219%	26.062%	0.133235	30.078%
Changed-token CE	124.8781	6.650%	40.176%	0.074930	60.677%
UPD pairwise	125.7317	4.614%	58.492%	0.029442	84.549%

5.3 Supervision-target granularity

To identify how supervision scope affects learning, we compare three objectives under the same strict execution policy. Full-set CE supervises every expert slot in the corrected assignment, including routes that needed no change. Changed-token CE removes untouched tokens but still supervises every corrected slot of a changed token. UPD is narrower and emits only the displaced-versus-replacement comparison for each changed slot.

Table 3 reports the resulting ablation. This progression reduces reassignment by 26.062%, 40.176%, and 58.492% relative to the no-distillation baseline, respectively; demand-CV reductions are 30.078%, 60.677%, and 84.549%. Relative to changed-token CE, UPD further reduces reassignment by 30.618% and demand CV by 60.708%, with a $+0.684\%$ PPL change. The screen varies target scope and loss form together, but it supports the claim that the smallest local supervision unit is the most effective among these choices.

The local pair avoids reinforcing feasible choices and avoids assigning a full target distribution to unaffected slots. This gives UPD a cleaner signal per correction, which helps explain why it improves load metrics more than the broader CE targets in this screen.

Table 4: Scaling evidence across heterogeneous training regimes. Values are UPD relative to the matched no-distillation baseline.

Model	Training evidence	Reassign. reduction	CV reduction	PPL Δ
350M	Full scratch, 3 seeds	$63.304 \pm 1.157\%$	$88.028 \pm 1.132\%$	$+0.235 \pm 1.211\%$
1B	Full scratch, 3 seeds	$62.655 \pm 0.565\%$	$86.272 \pm 0.510\%$	$-0.015 \pm 0.277\%$
7B-total	Full scratch, 3 seeds	$53.402 \pm 0.803\%$	$71.767 \pm 0.796\%$	$-0.431 \pm 0.156\%$

5.4 Evidence across model scales

Table 4 reports the same objective at 350M, 1B, and 7B-total parameters. Across fresh seeds at 350M, UPD reduces reassignment by $63.304 \pm 1.157\%$ and demand CV by $88.028 \pm 1.132\%$, while PPL changes by $+0.235 \pm 1.211\%$. At 7B total parameters, three matched all-parameter runs reduce reassignment by $53.402 \pm 0.803\%$ and demand CV by $71.767 \pm 0.796\%$, while PPL changes by $-0.431 \pm 0.156\%$.

The 7B runs use random initialization, train all 6.919B parameters with two-rank FSDP, and use 64 experts with top-8 routing; the 350M and 1B settings use eight experts and top-2 routing. The same changed-pair objective and coefficient apply across these heterogeneous scratch settings. These results support transfer across the tested scales, but they do not establish a scaling law.

5.5 Expert-parallel systems evaluation

To connect learned routing behavior to expert-parallel execution, we evaluate the two-rank A800 pipeline with both rerouting implementations. Figure 3 shows the three-seed means and capped ranges spanning the observed minimum and maximum. Panels (a) and (b) compare optimized and original rerouting for prefill and decode throughput, while panels (c) and (d) report p95 batch latency and time to first token (TTFT), respectively. Under the optimized implementation, mean prefill throughput is $5{,}415.2$ tok/s for the no-distillation baseline and $6{,}054.9$ tok/s for UPD; the corresponding decode

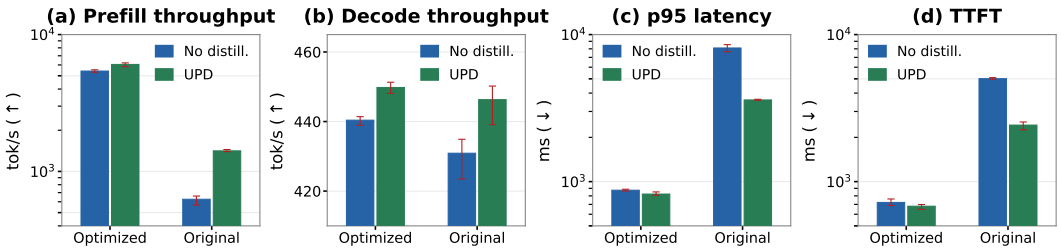


Figure 3: Two-rank A800 system measurements for the no-distillation baseline and UPD. Panels show (a) prefill throughput, (b) decode throughput, (c) p95 batch latency, and (d) time to first token (TTFT), each comparing the optimized and original rerouting implementations. Higher throughput is better (↑), while lower latency is better (↓).

means are 440.4 and 449.8 tok/s. Optimized p95 and TTFT means are 874.7/722.2 ms for the no-distillation baseline and 823.3/680.7 ms for UPD.

The original implementation exposes a larger implementation-sensitive gap: prefill throughput is 626.4 versus 1,414.0 tok/s, decode throughput is 430.9 versus 446.3 tok/s, p95 latency is 8,110.7 versus 3,594.7 ms, and TTFT is 5,015.8 versus 2,424.6 ms. We use the optimized measurements as the implementation-robust headline and retain the original solver values as a diagnostic of how the same learned demand interacts with a slower repair path. Additional inter-token and communication measurements are reported separately in the appendix; detailed per-seed systems results are in Table 17. The separate solver path audit records reductions of $62.395 \pm 0.289\%$ in expected overflow routes, $62.437 \pm 0.589\%$ in slow-path work, and $62.699 \pm 3.034\%$ in repair vacancies.

6 Limitations

UPD's small loss computation does not remove the training cost of the chosen feasibility policy. Its labels also depend on that policy. We test one policy with two route-equivalent implementations; benefits under other policies remain untested. The 350M, 1B, and 7B studies use bounded scratch-training budgets and different architectures, so they establish results at several scales but do not establish a scaling law or full-corpus pretraining performance. The one-seed component study varies target scope and loss form together, and does not measure sensitivity to margin, temperature, coefficient, delay, or ramp schedule. MaxScore* is a paper-aligned reproduction.

Systems measurements cover one host, two A800 GPUs, and two expert-parallel ranks. Longer pretraining, more architectures and input distributions, and multi-node serving are the next tests of how broadly these gains hold.

7 Conclusion

Uniform Pairwise Distillation provides a lightweight mechanism to improve router proposals for sparse MoE models. It builds equally-weighted pairwise ranking constraints solely from slot-level corrections observed during standard training. UPD reuses the existing router as both teacher and student, requiring no separate teacher network or extra trainable parameters. In matched from-scratch experiments across 350M, 1B, and 7B-parameter MoEs, UPD reduces raw-demand imbalance and runtime reassignment overhead while preserving perplexity close to the no-distillation baseline. The learned balanced routing generalizes to held-out in-domain and out-of-domain inputs. Under two-rank expert-parallel execution, better router proposals cut feasibility-repair work and improve prefill throughput with modest repeatable decode gains. Future work may explore applying this pairwise self-distillation paradigm to larger-scale pretraining, alternative execution policies, and combinations with existing global balancing objectives.

AI USE STATEMENT

In this work, we used generative AI tools for software-engineering assistance, debugging support, organization of experiment records, and language drafting and editing. The authors determined the research questions, experimental protocols, method choices, and scientific interpretations. We reviewed AI-assisted code and text against the executable implementation, frozen experiment reports, and cited literature; no result was generated, altered, or selected by a language model in place of an executed experiment. We will re-check the final manuscript, code, citations, and all reported numbers before submission. The authors take responsibility for the complete content, including text and artifacts produced with AI assistance.

ETHICS STATEMENT

This work studies routing and systems behavior in language-model MoEs using publicly available language-modeling datasets and checkpoints. It does not involve human-subject experiments or the release of personal data. Improvements to MoE efficiency may reduce computational cost, while the usual risks and biases of the underlying language models and datasets remain. We report negative and heterogeneous results to reduce the risk of overstating the method's scope.

REPRODUCIBILITY STATEMENT

The method is specified in Section 3, and model, data, optimization, and system workloads are described in Section 4. The appendix reports per-seed effects and the solver-equivalence contract. The supplementary code contains frozen configurations, dataset/checkpoint hashes, evaluation entry points, report generators, and protocol manifests. Every claimed result is linked to its source report and permitted scope in the manuscript evidence ledger. The retrospective baselines are included only from complete reports that pass the same infrastructure and provenance checks.

REFERENCES

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. Stablemoe: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7085–7095, 2022.

Bowen Dong, Yilong Fan, Yutao Sun, Zhenyu Li, Tengyu Pan, Zhou Xun, and Jianyong Wang. Maximum score routing for mixture-of-experts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 12619–12632, 2025.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pp. 5547–5569. PMLR, 2022.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304, 2023.

Shwai He, Weilin Cai, Jiayi Huang, and Ang Li. Capacity-aware inference: Mitigating the straggler effect in mixture of experts. In *International Conference on Learning Representations*, volume 2026, pp. 92274–92299, 2026.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems*, 5:269–287, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Masahiro Kada, Ryota Yoshihashi, Satoshi Ikehata, Rei Kawakami, and Ikuro Sato. Teacher-guided routing for sparse vision mixture-of-experts. *arXiv preprint arXiv:2604.21330*, 2026.

Wouter Kool, Chris J Maddison, and Andriy Mnih. Unbiased gradient estimation with balanced assignments for mixtures of experts. *arXiv preprint arXiv:2109.11817*, 2021.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pp. 6265–6274. PMLR, 2021.

Bo Li, Chuan Wu, and Shaolin Zhu. Macs: Modality-aware capacity scaling for efficient multimodal moe inference. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22134–22148, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. In *International Conference on Learning Representations*, volume 2025, pp. 62061–62121, 2025.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1525–1534, 2016.

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.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pp. 18332–18346. PMLR, 2022.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

A IMPLEMENTATION AND EXPERIMENTAL DETAILS

Our strict execution step builds on iterative overflow rerouting (Fedus et al., 2022; Dong et al., 2025). Given raw top-k routes, it first retains the highest-scoring incoming routes to each expert up to the capacity in Eq. (1). Vacant slots then propose their highest-ranked expert with remaining capacity, excluding experts already selected for the same token. If a greedy proposal sequence reaches a uniqueness dead end, a depth-first augmenting path moves the lowest-scoring eligible occupant first. A deterministic cyclic assignment is an explicitly logged fallback; reported strict-routing evaluations have zero post-correction capacity violations and bounded fallback use.

The optimized solver preserves this search contract while maintaining an index of assigned routes. The original implementation rebuilds an expert's occupant list by scanning all assigned routes at every augmenting visit. The optimized implementation precomputes token candidate orders and maintains the same score-ordered occupant list with indexed insertion and removal. Equivalence requires exact route indices, bitwise-equal weights, identical capacity/overflow/reassignment/fallback metadata, full-test PPL agreement within 10^{-6}, and identical generated-token hashes.

A.1 EXPERIMENTAL CONFIGURATIONS

Tables 5 and 6 record the configurations used for the three reported scale settings. The OLMoE-7B column describes the completed random-initialization, full-parameter scratch pilot.

Table 5: Model and routing configurations. OLMoE-7B uses the released architecture configuration and tokenizer but randomly initializes all weights.

Setting	350M	1B	OLMoE-7B
Initialization	Scratch	Scratch	Scratch
Layers	16	24	Released config
Hidden size	768	1024	Released config
Routed experts	8	8	64
Active routed experts	2	2	8
Capacity factor	1.0	1.0	1.0
Trainable parameters	Full model	973,483,008	Full model
Backbone	Trainable	Trainable	Trainable

OLMoE-7B denotes OLMoE-1B-7B-0924: roughly 7B total and 1B active parameters. Pretrained model weights are never loaded.

Table 6: Optimization configurations for the reported scale experiments.

Setting	350M	1B	OLMoE-7B
Seed(s)	52–54	52–54	3 seeds
Sequence length	512	512	256
Tokens/step	8,192	8,192	8,192
Optimizer steps	2,442	4,883	2,442
Total tokens	20,004,864	40,001,536	20,004,864
Peak learning rate	3×10^{-4}	2×10^{-4}	2×10^{-4}
Weight decay	0.1	0.1	0.1
UPD coefficient	0.02	0.02	0.02
UPD delay/ramp	60/120 steps	60/120 steps	60/120 steps
Router z-loss	0.001	0.001	0.001
Router KL	—	—	—
Precision	bf16	bf16	bf16

The OLMoE run uses two-rank FSDP full sharding and activation checkpointing on two A800 80GB GPUs.

Table 7: Role of each dataset and execution view. "Frozen" means that the selection and packed-token manifest were fixed before checkpoint evaluation.

Item	Role	Selection	Permitted use
WT103 train	Training	Fixed order per seed	Scratch optimization
WT103 val.	Development	Complete validation	Fresh-seed evidence
WT103 test	Frozen test	Complete test	Primary held-out claim
C4	Frozen OOD	First 2048 nonempty val. rows	OOD PPL/routing
LAMBADA	Frozen OOD	Complete OpenAI test text	Causal-LM PPL/routing
Strict routing	Operational	Rerouting enabled	Complete-method result
Raw top-2	Diagnostic	Correction disabled	Router internalization
Native	Baseline view	Method runtime retained	External comparison

B Data roles and evaluation views

All frozen dataset rows, tokenized blocks, evaluated-token counts, and checkpoints are content-hashed. Padding is excluded from routing and loss statistics. C4 and LAMBADA were not used to select the UPD coefficient, training step, checkpoint, or method. Matched rerouting holds the assignment policy fixed across external baselines; raw evaluation measures each router's demand before repair.

For raw top-k expert counts n_e, where n_e is the number of proposed route slots for expert e, demand CV is $\mathrm{std}_e(n_e)/\,\mathrm{mean}_e(n_e)$. Reassignment is $\sum_{i,s} \mathbf{1}[a_{i,s} \neq \tilde{a}_{i,s}]/(Nk)$, where $\mathbf{1}[\cdot]$ is one when its condition is true and zero otherwise. For a lower-is-better quantity x, reduction is $100(x_{\mathrm{ref}} - x_{\mathrm{UPD}})/x_{\mathrm{ref}}$, where "ref" denotes the comparison method and "UPD" denotes our method; PPL and throughput deltas use UPD minus reference. Positive load reduction and throughput delta are favorable, while negative PPL delta is favorable.

C Per-seed and scale details

C.1 1B and 350M results

Table 8: Fresh-seed WikiText-103 validation results. Relative effects compare UPD with the same-seed no-distillation baseline.

Seed	No-distill. PPL	UPD PPL	Reassign. ↓	Demand CV ↓
52	31.2110	31.1032	64.382%	85.788%
53	31.7054	31.8595	63.752%	86.480%
54	31.3526	31.2884	62.799%	81.926%

Table 9: Frozen WikiText-103 test effects by seed. "Strict" uses rerouting at evaluation; "raw" reports the unmodified top-2 proposal.

Seed	Strict reassign. ↓	Strict CV ↓	Strict PPL Δ	Raw CV ↓	Raw PPL Δ
52	63.191%	86.782%	−0.115%	86.606%	+0.054%
53	61.874%	86.457%	+0.363%	86.150%	+0.525%
54	62.901%	85.575%	−0.292%	85.376%	−0.152%

The seed-53 pair uses the opposite physical-GPU assignment from the other two pairs. All same-seed runs match initialization, train/evaluation data, parameter count, and token accounting. The device swap reduces the possibility that the consistent routing result is a fixed GPU-placement effect.

Table 10: Fresh-seed 350M scratch confirmation. Relative effects compare UPD with the same-seed no-distillation baseline after 2442 steps.

Seed	No-distill. PPL	UPD PPL	Reassign. reduction	CV reduction	PPL Δ
52	55.7820	55.4313	64.476%	88.927%	-0.629%
53	55.1787	56.2537	63.706%	88.725%	$+1.948\%$
54	54.3221	53.9891	61.730%	86.431%	-0.613%
Mean $\pm$ std.	—	—	$63.304 \pm 1.157\%$	$88.028 \pm 1.132\%$	$+0.235 \pm 1.211\%$

C.2 Full-parameter OLMoE-7B scratch results

The 7B-total experiment compares no distillation and UPD across three matched random initializations, with all 6,919,161,856 parameters trainable for 2442 optimizer steps. Each condition processes 20,004,864 tokens, uses two-rank FSDP full sharding, and computes capacity over the global micro-batch. The architecture source supplies only config and tokenizer files; pretrained weights are not loaded.

Table 11: Three-seed OLMoE-7B-total full-parameter scratch results. Values are UPD relative to the matched no-distillation baseline.

Comparison	Reassign. reduction	CV reduction	PPL Δ
UPD vs no distill.	$53.402 \pm 0.803\%$	$71.767 \pm 0.796\%$	$-0.431 \pm 0.156\%$

Values are mean $\pm$ population standard deviation over three matched seeds.

D Systems and solver details

We use the original solver to measure repair work and the route-equivalent optimized solver for the primary performance comparison. The optimized implementation preserves candidate and occupant search order and is accepted only after bitwise equality of route indices and weights, exact solver metadata, full-test metric equality within 10^{-6}, and identical generated-token hashes.

Table 12: Per-seed UPD effects with the equivalent optimized rerouting implementation.

Seed	Prefill tok/s $\uparrow$	TTFT $\downarrow$	Decode tok/s $\uparrow$	Inter-token p95 $\downarrow$
52	14.729%	-1.542%	$+2.530\%$	$+3.257\%$
53	10.807%	$+14.915\%$	$+2.272\%$	$+2.524\%$
54	9.886%	$+3.000\%$	$+1.643\%$	$+2.109\%$

Prefill uses global batch 8 and sequence length 512. Decode uses global batch 64, 64-token prompts, 32 fixed greedy continuation steps, and KV caching. The prefill and TTFT columns use the original three-repeat optimized benchmark. The decode columns report each seed's mean over six balanced timing groups, each with three repeats after fixed warm-up; method order is reversed across groups. Instrumented collective profiles are run separately from end-to-end timing.

Per-seed expected-overflow reductions are 62.346%, 62.067%, and 62.771%; slow-path-work reductions are 62.225%, 61.846%, and 63.240%. All audited calls remain capacity safe and use the same frozen test inputs. Separate timing with the optimized solver measures the end-to-end effect.

Table 13: Supplementary per-seed records for inter-token tail latency and all-to-all communication. IT-p95 denotes inter-token 95th-percentile latency; P-A2A and D-A2A denote prefill and decode all-to-all time, respectively. Values are milliseconds for each seed; the mean is the arithmetic mean across seeds. Optimized inter-token p95 uses the six-group balanced decode remeasurement; original inter-token p95, P-A2A, and D-A2A are from the original solver benchmark.

Metric	Impl.	Method	Seed 52	Seed 53	Seed 54	Mean
IT-p95 ↓	Optimized	No distill.	149.6	148.3	147.9	148.6
IT-p95 ↓	Optimized	UPD	144.8	144.6	144.8	144.7
IT-p95 ↓	Original	No distill.	160.3	151.3	150.2	153.9
IT-p95 ↓	Original	UPD	145.1	147.2	151.3	147.9
P-A2A ↓	Original	No distill.	3096.0	3818.4	3865.4	3593.2
P-A2A ↓	Original	UPD	1637.0	1451.0	1648.5	1578.8
D-A2A ↓	Original	No distill.	108.1	123.9	129.0	120.3
D-A2A ↓	Original	UPD	63.0	75.6	64.4	67.7

Table 14: Original-solver path audit (UPD reduction relative to the no-distillation baseline, mean ± population standard deviation over seeds 52–54). These instrumented work counts support mechanism attribution; audit runtime is not throughput.

Path quantity	Reduction
Expected overflow routes/call	$62.395 \pm 0.289\%$
Slow-path work routes/call	$62.437 \pm 0.589\%$
Repair vacancies/call	$62.699 \pm 3.034\%$
Repair activation rate	$4.629 \pm 1.143\%$

E METRICS, PROTOCOL, AND EXTERNAL-BASELINE VIEWS

E.1 METRIC DEFINITIONS

For a sparse-layer batch with valid-token raw expert counts n_e, raw demand CV is

$$\mathrm{CV}(n) = \frac{\sqrt{E^{-1}\sum_{e=1}^{E}(n_e - \bar{n})^2}}{\bar{n}}, \qquad \bar{n} = E^{-1}\sum_{e=1}^{E} n_e. \tag{5}$$

Reassignment is the changed-slot fraction

$$R = \frac{\sum_{i=1}^{N}\sum_{s=1}^{k}\mathbf{1}[a_{i,s} \neq \tilde{a}_{i,s}]}{Nk}. \tag{6}$$

For a lower-is-better load quantity x, the reported reduction is $100(x_{\mathrm{ref}} - x_{\mathrm{UPD}})/x_{\mathrm{ref}}$. PPL and throughput deltas use $100(x_{\mathrm{UPD}} - x_{\mathrm{ref}})/x_{\mathrm{ref}}$. Hence a positive load reduction and throughput delta are favorable, while a negative PPL delta is favorable. Standard deviations are population standard deviations over the registered seeds, not standard errors or confidence intervals.

Executed-load CV is not used as the primary balance metric because strict rerouting can make it small for both methods by construction. Active-batch saturation is also descriptive under strict capacity. Raw demand CV, reassignment, conditional correction intensity, fallback, and post-capacity violations jointly distinguish the learned proposal from the feasibility guarantee.

E.2 PROTOCOL AND REPRODUCIBILITY NOTES

All primary 1B pairs match initialization hashes, tokenizer, parameter count, training data and order, evaluated tokens, optimizer schedule, and strict-capacity configuration. Every completed method has finite metrics, complete token accounting, recorded runtime and memory, zero post-capacity violations, bounded fallback, and an observed router z-loss. Test and OOD datasets are content-hashed before evaluation. Capacity correction is disabled only for explicitly labeled raw-router diagnostics.

The external-baseline extension uses the same seeds and token budget. It evaluates three views: each method's native mechanism, common strict rerouting, and raw top-2 routing. Aux uses its training objective with strict rerouting; LFB restores its expert-bias controller state; MaxScore* uses the paper-aligned generalized solver. We reuse the frozen no-distillation and UPD outputs. Each extension records configs, environment fingerprints, and all per-seed outcomes.

We verify provenance separately for the original runs and the later baseline runs, then check that both groups match on initialization, model and tokenizer, training data and order, token and step budgets, optimization schedule, and evaluation inputs. Evaluation uses the fixed final checkpoints without retraining or checkpoint selection.

Table 15: Frozen WT103 test load reductions of UPD relative to external baselines under three evaluation views (mean ± population standard deviation over seeds 52–54). Native retains each baseline's runtime mechanism, matched applies common strict rerouting, and raw disables rerouting to diagnose router demand.

Reference	Native reassign.	Native CV	Matched reassign.	Matched CV	Raw CV
Aux	62.261 ± 0.770%	86.165 ± 0.203%	62.261 ± 0.770%	86.165 ± 0.203%	85.977 ± 0.256%
LFB	58.013 ± 2.262%	97.253 ± 0.072%	84.835 ± 0.556%	97.234 ± 0.072%	97.198 ± 0.082%
MaxScore*	77.670 ± 0.877%	90.243 ± 0.342%	69.682 ± 1.097%	90.850 ± 0.204%	90.649 ± 0.163%

All entries are lower-is-better percentage reductions. Aux native and matched execution are identical because its native protocol already uses the same strict rerouting. Native PPL comparisons appear in Table 1.

Table 16: Frozen OOD load reductions of UPD relative to external baselines under three evaluation views (mean ± population standard deviation over seeds 52–54). Native retains each baseline mechanism, matched applies the common strict rerouting rule to all checkpoints, and raw disables rerouting.

Data	Ref.	Native reassign.	Native CV	Matched reassign.	Matched CV	Raw CV
C4	Aux	36.834 ± 0.698%	36.061 ± 1.713%	36.834 ± 0.698%	36.061 ± 1.713%	35.536 ± 1.702%
C4	LFB	36.940 ± 1.068%	69.657 ± 0.671%	67.878 ± 0.552%	69.535 ± 0.755%	69.385 ± 0.857%
C4	MaxScore*	63.908 ± 1.016%	54.612 ± 1.811%	51.512 ± 2.101%	54.406 ± 2.234%	53.655 ± 2.331%
LAMBADA	Aux	39.630 ± 3.332%	40.055 ± 3.360%	39.630 ± 3.332%	40.055 ± 3.360%	39.079 ± 3.676%
LAMBADA	LFB	37.691 ± 4.661%	63.754 ± 2.484%	64.106 ± 1.716%	63.726 ± 2.548%	63.439 ± 2.442%
LAMBADA	MaxScore*	65.054 ± 2.876%	54.637 ± 5.327%	51.878 ± 5.638%	53.576 ± 5.751%	52.748 ± 5.819%

All entries are favorable lower-is-better percentage reductions. Aux native and matched execution coincide because its native protocol uses the same strict rerouting. Native PPL comparisons appear in Table 2.

F Additional result views

The aggregate result tables are given in the main text so that each claim can be read together with its complete baseline comparison. This appendix retains the per-seed, matched/raw external views, scale-specific details, and solver audit needed to reproduce and interpret those aggregates.

Table 17: Two-A800 expert-parallel effects of UPD relative to the no-distillation baseline under the *same* rerouting implementation. The optimized rows are the primary result; original rows diagnose implementation sensitivity.

Impl.	Metric	Throughput Δ	Latency reduction	A2A reduction
Optimized	Prefill	$+11.807 \pm 2.100\%$	$5.889 \pm 1.307\%$ (p95)	—
Optimized	TTFT	—	$5.458 \pm 6.939\%$	—
Optimized	Decode	$+2.149 \pm 0.372\%$	2.630% (p95)	—
Original	Prefill	$+127.081\%$	55.599% (p95)	55.492%
Original	TTFT	—	51.627%	—
Original	Decode	$+3.599\%$	—	43.599%

Every row compares UPD with the no-distillation baseline within one implementation; $+127.081\%$ is *not* optimized versus original. Optimizing rerouting itself raises no-distillation/UPD prefill throughput by $767.774\%/328.510\%$. Throughput values are mean $\pm$ population standard deviation over seeds 52–54; the decode row uses each seed's mean over six balanced timing groups with three repeats per group, and its p95 cell is the mean of those seed-level group means. Optimized A2A aggregates were not recorded.