Manifold-Constrained Quantized Sparse MoE: Error-Driven Routing, Shared-Anchor Compression, and MCQSMoE Evidence

Agent-GO Foundation Models Team

Run Snapshot: 2026-02-08

Abstract

This paper studies a concrete architectural claim: sparse expert routing can be made more interpretable and compression-native by replacing learned router logits with reconstruction-error routing, then parameterizing experts as manifold-constrained shared anchors plus quantized deltas. In MCQSMoE, each expert acts as a compressor-reconstructor, routing is *top-k lowest reconstruction error*, and expert weights are represented as $W_e = \mathcal{R}(A+\Delta_e)$, where A is a shared anchor and Δ_e is low-bit groupwise quantized. This couples three practical goals in one design: deterministic routing, memory-efficient expert storage, and auditable per-token decisions.

On the 2026-02-08 run snapshot, all evaluation gates pass (Q/QS/P/PS/DB/T/X). Multi-run performance shows `ParallelForward_medium_b32/Parallel` at 529.7 tok/s versus 100.6 tok/s for serial (5.26x), and `ParallelZeroCopy` at 512.8 tok/s (5.10x). Under a matched dense baseline, QSMoE is faster in both tested regimes (1.252x small, 1.193x large). Trained-artifact parity reports tight numerical agreement (max absolute error $\leq 7.84 \times 10^{-4}$, NRMSE $\approx 0.02\%$) and deterministic replay over 10 runs, while imported FFN artifacts preserve 1.94x memory savings.

These results establish internal algorithmic validity and systems efficiency for the proposed design. We do not claim public-benchmark state-of-the-art yet; the external harness remains a small-scale calibration checkpoint.

1 Introduction

Sparse MoE systems are attractive because they promise higher effective model capacity at bounded per-token compute [5, 4, 2]. In practice, two issues repeatedly complicate deployment: routing opacity (learned router logits are hard to audit) and memory pressure (many experts with expensive full-precision weights).

This work focuses on a different operating point: make routing itself compression-native and directly measurable. MCQSMoE replaces learned router logits with reconstruction-error routing, where each expert is a compressor-reconstructor and selection is based on which experts best reconstruct the token. It combines this with a shared-anchor expert parameterization and low-bit deltas, constrained on a row-sphere manifold. The resulting stack targets three engineering outcomes simultaneously: *(i)* deterministic routing behavior, *(ii)* memory/computation efficiency, and *(iii)* auditable decision traces.

What is new in this paper.

1. **Routing mechanism:** error-driven top-k expert selection instead of learned router logits, with explicit per-token routing audit statistics.

2. **Parameterization:** manifold-constrained shared-anchor plus quantized per-expert deltas for compressed expert storage and reconstruction.

3. **Execution and validation:** grouped expert dispatch and repeated-run measurement with matched dense baselines and trained-artifact parity checks.

4. **Evidence boundaries:** explicit separation between proven internal claims and unproven external-SOTA claims, aligned with technical-report style discipline [1, 3].

Why this matters. If sparse routing is deterministic, inspectable, and cheaper than matched dense alternatives under controlled settings, it becomes substantially easier to trust and iterate in production systems. The paper is therefore not only about reproducibility mechanics; it is about validating a specific sparse modeling strategy that can reduce serving cost while keeping behavior diagnosable.

2 Method

2.1 Error-Driven Sparse Routing

For token x, each expert e reconstructs $\hat{x}_e$. Routing score is the reconstruction error $\text{err}_e = \|x - \hat{x}_e\|^2$ (or configured variant). MCQSMoE selects the top-k experts with lowest error:

$$\mathcal{K}(x) = \text{TopK}_{\min e} \ \text{err}_e.$$

Selected experts are combined by softmin weighting:

$$w_e = \frac{\exp(-\text{err}_e/T)}{\sum_{e' \in \mathcal{K}(x)} \exp(-\text{err}_{e'}/T)},$$

with fixed $T = 0.1$ in the current implementation. Unlike standard learned-router MoE, this routing path is directly auditable: the full error matrix, selected experts, weights, entropy, and residual norms are exported by the audit object for each batch.

2.2 Manifold-Constrained Shared-Anchor Expert Weights

Each expert weight is represented as:

$$W_e = \mathcal{R}(A + \Delta_e),$$

where A is a layer-level shared anchor, Δ_e is a per-expert quantized delta (int4/int2 groupwise), and $\mathcal{R}$ denotes manifold retraction (row normalization on sphere rows in the tested configuration). This avoids storing full dense weights per expert and enables explicit memory control. In imported artifacts, the tested FFN path reports 1.94x memory reduction versus naive per-expert fp16 storage.

2.3 Execution Path

Inference uses linear decomposition:

$$Y = X(A + \Delta_e)^T = XA^T + X\Delta_e^T.$$

The anchor term is computed once, then delta terms are computed for selected experts and combined. Tokens are grouped by expert/mask assignment for batched dispatch, reducing redundant work in sparse execution.

Protocol Flow: Claim definitions (Protocol) → Track execution (Q/QS/P/PS/T) → Scripted runs → Artifact logs → Gate checks A–F → Summary output.

Strictness: missing trained artifacts or skip-detected Track T behavior triggers failure. Missing benchmark coverage in PS triggers failure.

Figure 1: Claim-to-artifact pipeline used in this paper package.

Claim	Primary Metric	Verification Route	Artifact Path
Routing/composition determinism and stability	Quality test pass/fail; routing audit stats	Track Q + QS test suites	`artifacts/logs/quality.log`, `artifacts/logs/quality_multiseed.lo`
Forward-path throughput gains for selected paths	`ns/op`, `tok/s`, 95% CI	Track P + PS benchmarks	`artifacts/logs/perf.log`, `artifacts/bench/perf_multirun_stats`
Compression/quantization correctness within tolerance	Golden parity errors; invariant checks; determinism	Track T strict gate	`artifacts/logs/track_t.log`

Table 1: Claim-to-evidence mapping from protocol to reproducible artifacts.

2.4 Claim Compiler and Falsifiability

The evaluation stack is organized as:

$$\text{Claim} \rightarrow \text{Track} \rightarrow \text{Script} \rightarrow \text{Artifact} \rightarrow \text{Gate}.$$

This structure enforces falsifiable claims: every statement in the paper must map to a metric, an executable test path, and a concrete artifact.

3 Experimental Setup

3.1 Environment

Environment metadata is captured in `artifacts/env/env_snapshot.txt`: Linux 6.17.12, Go 1.25.5, CPU AMD Ryzen 7 7700X (16 logical cores), and 31,184 MiB RAM.

3.2 Evaluation Axes

We evaluate three axes:

1. **Algorithmic validity:** correctness, determinism, routing behavior, and trained-artifact parity (Q, QS, T).
2. **Systems efficiency:** throughput/latency on sparse forward paths with repeated runs and confidence intervals (P, PS).
3. **Comparative sanity checks:** matched dense baseline and a fixed external-style MCQA harness snapshot (DB, X).

Proven in this paper: internal correctness tests, deterministic replay for defined parity tests, benchmark throughput/latency statistics on one machine.

Not proven yet: matched dense-vs-MCQSMoE external benchmark superiority, public leaderboard quality claims, cross-hardware deployment guarantees.

Figure 2: Claim boundary used to prevent over-interpretation of results.

Benchmark	n	Mean ns/op	Std ns/op	95% CI ns/op	Mean tok/s	95% C
ParallelForward_medium_b32/Parallel	5	60,432,829.8	1,215,796.0	1,065,692.2	529.70	
ParallelForward_medium_b32/ParallelZeroCopy	5	62,453,989.8	1,970,309.2	1,727,052.2	512.78	
ParallelForward_medium_b32/Serial	5	318,085,866.4	4,274,570.8	3,746,826.5	100.61	
QSMoEForward	5	9,353,816.2	124,841.4	109,428.3	NA	
QSMoEForwardLarge	5	152,751,385.2	818,286.3	717,259.6	NA	
WorkspaceForward_medium_b32	5	286,005,141.2	4,131,761.1	3,621,648.3	NA	

Table 2: Canonical multi-run performance statistics from `artifacts/bench/perf_multirun_stats.csv`.

3.3 Protocol Controls

The canonical runner is:

```
./foundation_models/paper_mcsqoe/scripts/40_run_all.sh
```

QS seeds are fixed (`11 23 47 101 211`). PS uses $n = 5$ repeated runs and reports 95% CI as $1.96 \cdot \sigma/\sqrt{n}$. DB aligns dense and QSMoE parameter/compute budgets for direct micro-benchmark comparison.

3.4 Scope Boundary

This paper validates mechanism-level and systems-level claims for MCQSMoE. It is not positioned as a full external leaderboard paper yet.

4 Results

4.1 Overall Gate Status

From `artifacts/reports/summary.md` (timestamp 2026-02-08T11:37:19Z): Q=PASS, QS=PASS (5 seeds), P=PASS, PS=PASS (5 runs), DB=PASS, T=PASS, X=PASS.

4.2 Sparse Execution Speedup Is Stable

Across five runs, `Parallel` achieves 529.7 tok/s versus 100.6 tok/s for serial (5.26x), and `ParallelZeroCopy` achieves 512.8 tok/s (5.10x). This indicates the sparse grouped-dispatch path preserves a large speed advantage under repeated measurement, not only single-run peaks.

4.3 QSMoE Beats Matched Dense Baseline in Tested Regimes

Under matched parameter/compute settings, QSMoE is faster than dense on both tested scales: 1.252x (small) and 1.193x (large), supporting the claim that the sparse/compression-native path is practically useful, not only compact.

Case	QSMoE ns/op	Dense Matched ns/op	Dense/QSMoE
small	9,242,506	11,571,670	1.2520x
large	152,577,554	182,059,712	1.1932x

Table 3: Dense baseline under matched parameter/compute budget from `artifacts/reports/dense_baseline_summary.md`.

Track T Check	Observed Result
Required artifacts precheck	PASS (all required manifests and golden shard found)
Golden parity (layer 0)	max_abs=0.000732, rmse=0.000185, nrmse=0.02%
Golden parity (layer 5)	max_abs=0.000784, rmse=0.000182, nrmse=0.02%
Manifold invariant checks	max norm error range 0.000024 to 0.000069
Determinism replay	PASS: identical outputs across 10 runs
Exported model memory summary	12,672 KiB total; per-layer FFN savings 1.94x
Final gate status	`TRACK_T_STATUS=PASS`

Table 4: Track T artifact parity evidence from `artifacts/logs/track_t.log`.

4.4 Compression Fidelity and Determinism

Track T shows tight parity against exported golden artifacts, with max absolute error below 8×10^{-4}, low NRMSE, manifold invariant checks passing, and exact deterministic replay across 10 runs.

4.5 External Harness Snapshot (Calibration Checkpoint)

The external snapshot is intentionally small-scale (`mmlu-tiny`, fixed seed) and is used as an early calibration/behavior checkpoint, not a final generalization claim.

5 Limitations and Threats to Validity

The current evidence is strongest on mechanism and systems behavior, and weaker on broad external generalization. Specifically:

1. External evaluation is currently a small harness snapshot (`mmlu-tiny`) rather than full public benchmark suites.
2. Throughput claims are micro-benchmark scoped; end-to-end serving latency/throughput under production traffic is not yet reported.
3. Training-cost claims (e.g., wall-clock convergence, energy efficiency versus alternatives) are not yet established in this package.

Next upgrades for a stronger publication. (1) Expand external evaluation to larger public datasets and stronger baselines. (2) Add serving-level results (batching, queueing, p50/p95/p99 latency). (3) Add training-side ablations for routing temperature, bottleneck size, and quantization granularity under fixed compute budgets.

Model	N	Accuracy	ECE	Brier	AUROC	Confident-Wrong@0.8
`ar_baseline`	100	0.5000	0.0773	0.2277	0.3308	0.0000
`diffusion_baseline`	100	0.4100	0.5892	0.5890	0.6372	0.5900
`self_consistency`	100	0.7800	0.1913	0.1748	0.0760	0.0000

Table 5: External-style MCQA harness snapshot (`mmlu-tiny`, seed 424242) from `artifacts/reports/external_eval/summary.md`.

Limitation	Current Impact	Required Upgrade
No matched dense baseline under equal budget	Throughput claims are benchmark-local, not fair end-to-end superiority	Add dense baseline with same parameter/FLOP envelope
No external task-level benchmark table	No public quality leaderboard claim is supported	Add fixed evaluation harness and report external metrics
Robustness seeds only shuffle test order	Limited signal on model/data stochastic stability	Add randomness controls for model init/data order
n=5 CI with normal approximation	Variance estimates may be optimistic on noisy hardware	Increase sample count and/or use small-sample robust reporting
Hardware-state controls not pinned	Bench variance can drift with thermal and scheduler noise	Add governor/thermal pinning and multi-time-window repeats

Table 6: Known gaps and mitigation path before broader claims.

6 Conclusion

This paper validates a specific sparse-modeling thesis: routing by reconstruction error, paired with manifold-constrained shared-anchor quantization, can produce an auditable and efficient MoE stack. In the current MCQSMoE snapshot, the method is internally consistent (deterministic and parity-verified), faster than matched dense baselines in the tested settings, and measurably faster than serial execution under repeated runs.

The next publication step is clear: keep the same claim discipline while adding large external benchmarks and serving-level evaluations. If those results hold, the method transitions from a strong internal technical result to a broader competitive claim.

Reproducibility Statement

All commands, logs, and summary artifacts are included in this bundle under `scripts/`, `protocol/`, `configs/`, and `artifacts/`. The canonical runner is:

```
./foundation_models/paper_mcsqoe/scripts/40_run_all.sh
```

References

[1] DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. URL https://arxiv.org/abs/2412.19437.

[2] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021. URL https://arxiv.org/abs/2101.03961.

[3] Claus Huang, Han Chi, Chenyang Luo, Xia Wei, Bolin An, Wei Guo, Xin Xin, Qian Xu, and Lei Zhang. mhc: Memory-safe hierarchical caching reduces meta data and eliminates load peaks for dynamic moe serving. *arXiv preprint arXiv:2512.24880*, 2025. URL https://arxiv.org/abs/2512.24880.

[4] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020. URL https://arxiv.org/abs/2006.16668.

[5] 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. *arXiv preprint arXiv:1701.06538*, 2017. URL https://arxiv.org/abs/1701.06538.