ACNA: Decentralized Active-Cell Routing for Sparse and Dynamically Expandable Neural Networks

Anonymous Authors
English content-review mirror (not official ICLR style)

August 19, 2026

Abstract

Sparse neural networks activate only a subset of computation, but conventional Mixture-of-Experts (MoE) systems typically rely on a learned centralized router coupled to the expert pool. We propose the Active Cell Neural Architecture (ACNA), in which independently parameterized Expert Cells maintain local activation evidence through a receptor and resting bias, while low-rank Social Intervention provides lightweight inter-cell modulation. A residual transition branch enables function-preserving insertion into pretrained Transformers.

Experiments provide a proof of feasibility. A controlled CPU ablation shows lower latency when inactive experts are skipped. In Qwen2.5-Coder-32B, two-cell Python/SQL routing reaches 97.5%; a Math cell can be inserted and later removed without retraining the remaining cells; and mixed-domain prompts place both relevant cells in the Top-2 in 100% of 60 cases. Sequential expansion to eight domains reaches 159/160 (99.38%) in a post-calibration routing evaluation, with 80/80 (100%) accuracy on the original four domains; because this 160-prompt pool overlaps routing-calibration material, we do not label this result blind or held-out. An isolated eight-expert NVIDIA GB10 benchmark yields 1.09× speedup and 8.44% lower latency, while energy/token is 3.32% higher under the current PyTorch execution path. Complementary RTL simulations verify one-hot cell gating, dynamic cell insertion, and a reduction in registered compute activity from 300 dense events to 100 sparse events (66.7%). The RTL result is an activity-level hardware feasibility result, not a direct board-power measurement; together with the GPU result it motivates future cell-aware accelerators with grouped dispatch and fine-grained clock/power gating.

1 Introduction

Conditional computation seeks to increase neural-network capacity without increasing active computation in direct proportion to parameter count. Sparsely-gated MoE layers established this principle at large scale (Shazeer et al., 2017), followed by systems such as GShard (Lepikhin et al., 2020) and Switch Transformer (Fedus et al., 2022). However, conventional MoE systems typically use a learned router that scores a shared expert pool and coordinates dispatch.

ACNAexplores a different organizational principle. Instead of learning a separate global gating network, each Expert Cell contains its own activation evidence through a local receptor and resting bias. A globally broadcast representation is evaluated by all cells, while a low-rank Social Intervention term permits structured competition or cooperation. The goal is *not* to eliminate routing computation; it is to remove the need for a separately parameterized centralized router whose learned output dimensionality is tied to the available expert population.

We emphasize an implementation distinction. The general ACNAformulation supports local threshold-based activation, but our fixed-budget 32B experiments additionally use a global Top-1 comparison/dispatch primitive to keep heavy expert execution controlled. Thus, the current implementation is best described as *router-network-free*, not as eliminating every global synchronization or selection operation.

Terminology. Throughout this paper, *decentralized* means that learned activation evidence and activation parameters are owned by individual Expert Cells and no separate learned centralized router network is used; the fixed-budget implementation may still perform a global Top-1 comparison for dispatch. We use *hot-plug* to mean adding (in software) or enabling (in RTL) a new Expert Cell after prior cells have been established while keeping previously installed heavy-expert parameters frozen; the new cell may still require adaptation/calibration, and the RTL prototype does not implement FPGA partial reconfiguration.

The central scaling hypothesis is that representational capacity and heavy expert computation can be separated. Let N denote available cells and K the number of active heavy experts. ACNAdoes not imply $O(1)$ total inference time because receptor and interaction costs grow with N. Instead, it aims to keep the expensive expert term approximately bounded when K remains small.

Contributions. We make four main contributions:

- We formulate a cell-local activation architecture with Resting Potentials, low-rank Social Intervention, and hard expert skipping.
- We introduce a function-preserving transition mechanism for inserting ACNAinto a pretrained Transformer without changing the original layer at initialization.
- We experimentally evaluate reversible hot-plug/hot-unplug behavior, mixed-domain routing, and sequential expansion from two to eight expert domains in a 32B pretrained model.
- We separate *logical sparsity* from *hardware energy proportionality* using a GPU benchmark and RTL cell-gating simulations, showing where present software/hardware loses the theoretical sparsity benefit.

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.

2 Related Work

2.1 Mixture-of-Experts and Sparse Routing

The sparsely-gated MoE layer uses a trainable gating network to select a sparse subset of experts (Shazeer et al., 2017). GShard demonstrated large-scale conditional computation with automatic sharding (Lepikhin et al., 2020), while Switch Transformer simplified expert routing to a Top-1 policy (Fedus et al., 2022). Expert Choice Routing reverses the assignment direction by allowing experts to select tokens, addressing load balancing and adaptive compute allocation (Zhou et al., 2022).

Recent Routing-Free Mixture-of-Experts internalizes activation decisions within experts and removes external routers, Softmax, Top-K, and hard-coded load-balancing rules (Liu et al., 2026). This is closer to ACNA's cell-local premise than conventional router-based MoE; however, our evaluated fixed-budget variant is less routing-free because it retains a global Top-1 dispatch comparison, while this work emphasizes Resting Potential, low-rank Social Intervention, reversible post-training cell insertion/removal, and hardware-visible cell gating in a pretrained 32B setting.

2.2 Modular and Continual Expansion

Progressive Neural Networks expand capacity while preserving previously learned components (Rusu et al., 2016). PathNet learns reusable pathways through a larger network and can freeze useful pathways during transfer (Fernando et al., 2017). AdapterFusion separates task-specific knowledge acquisition from later non-destructive composition (Pfeiffer et al., 2021). These works motivate modularity and interference control; ACNAdiffers by treating each added capability as a conditionally executed Expert Cell inside a sparse computation layer.

Adaptive MoEs for Expanding Task Sets explicitly grows the expert population as new tasks arrive and reports simulations on growing task streams (Candocia et al., 2025). Our study addresses the same expansion pressure from a different mechanism and additionally measures preservation of frozen heavy experts, reversible removal, routing-boundary interference/calibration, and a hardware-gating path; this comparison does not imply that either method subsumes the other.

2.3 Sparsity-Aware Hardware

Algorithmic sparsity does not automatically produce proportional hardware savings. EIE and SCNN demonstrate that sparse neural computation becomes substantially more efficient when compressed representation, memory traffic, scheduling, and compute skipping are co-designed with hardware (Han et al., 2016; Parashar et al., 2017). Commercial GPUs likewise exploit selected forms of structure: NVIDIA Ampere Sparse Tensor Cores support a fixed 2:4 sparsity pattern (NVIDIA, 2023). These precedents motivate a separate hardware question for ACNA: whether cell-level activation can be mapped to execution and power-management mechanisms at a granularity that avoids paying for inactive expert regions.

3 Active Cell Neural Architecture

3.1 Cell-Local Receptor and Resting Potential

For token representation $x \in \mathbb{R}^d$, cell i computes

$$p_i(x) = r_i(x) + b_i, \tag{1}$$

where r_i is a learnable receptor and b_i is the Resting Potential. In the 32B implementation, $r_i(x) = w_i^\top x$ and b_i is implemented as the receptor bias. A negative or sufficiently small bias makes the cell less likely to activate in the absence of matching evidence.

Let

$$P = [p_1, \ldots, p_N]^\top. \tag{2}$$

3.2 Low-Rank Social Intervention

A dense pairwise interaction matrix would require $O(N^2)$ parameters and computation. Instead, ACNAuses $U, V \in \mathbb{R}^{N \times r}$ with $r \ll N$:

$$\Delta P = U(V^\top P), \qquad S = P + \Delta P. \tag{3}$$

The low-rank state requires $O(Nr)$ storage. Rows associated with newly inserted cells can be initialized independently. In several hot-plug experiments we set the new cell's V row to zero at insertion, producing one-way isolation during initial adaptation.

3.3 Activation and Physical Expert Skipping

The general thresholded activation is

$$a_i = \mathrm{ReLU}(S_i - \tau), \qquad m_i = \mathbb{1}(a_i > 0). \tag{4}$$

Only experts with $m_i = 1$ should be evaluated. This distinction is essential: evaluating all experts and multiplying inactive outputs by zero gives mathematical sparsity but not physical compute skipping.

For the fixed-budget 32B routing experiments, we use

$$j = \arg\max_i S_i, \qquad m_i = \mathbb{1}(i = j), \tag{5}$$

and diagnostic routing probabilities

$$q_i = \mathrm{softmax}(S)_i. \tag{6}$$

The expert branch is

$$F_{\mathrm{ACNA}}(x) = \sum_{i=1}^{N} m_i \, q_i \, E_i(x). \tag{7}$$

The global $\arg\max$ in Eq. 5 is a lightweight dispatch primitive rather than a learned router network; removing this synchronization is not claimed in the present implementation.

3.4 Function-Preserving Transition

Directly replacing a pretrained MLP can cause severe representation drift. The 32B implementation therefore uses a residual transition branch:

$$F(x) = F_{\mathrm{base}}(x) + g F_{\mathrm{ACNA}}(x), \tag{8}$$

with $g = 0$ at insertion. Thus insertion is exactly function-preserving at initialization. In the reported checkpoints, $g \approx 0.01062$, so the experiments establish routing and modularity while the pretrained MLP remains the dominant functional path.

3.5 Cell-Local Expansion and Boundary Calibration

When inserting a new cell m, existing expert parameters are frozen. For positive examples of the new domain we use a margin objective

$$\mathcal{L}_{\mathrm{pos}} = \max\left(0,\, \gamma_{\mathrm{pos}} - \left[S_m - \max_{j<m} S_j\right]\right), \tag{9}$$

and for old-domain negatives

$$\mathcal{L}_{\mathrm{neg}} = \max\left(0,\, \gamma_{\mathrm{neg}} - \left[\max_{j<m} S_j - S_m\right]\right). \tag{10}$$

This directly trains the new cell to win on its own domain and yield on old domains.

At eight cells, sequential insertion alone produces routing-boundary interference. We therefore perform a lightweight global boundary calibration with all heavy expert weights frozen:

$$\mathcal{L}_{\text{cal}} = \text{CE}(S, y) + \lambda \max\left(0,\, \gamma - \left[S_y - \max_{j \neq y} S_j\right]\right). \quad (11)$$

Only routing parameters are adjusted in this stage.

4 Computational Structure

Let N be the number of available cells, d hidden dimension, r social rank, K active cells, and C the cost of one heavy expert. The computational structure is

$$T_{\text{ACNA}} = O(Nd) + O(Nr) + O(KC). \quad (12)$$

ACNAtherefore does *not* claim constant total inference complexity. Its intended scaling property is that the expensive KC term can remain approximately bounded when K is controlled, while lightweight receptor and interaction terms still grow with N.

5 Experimental Setup

Backbone. Large-model experiments use Qwen2.5-Coder-32B-Instruct (Hui et al., 2024), with hidden size 5120. ACNAis inserted at the final Transformer MLP location. Each experimental expert uses an intermediate dimension of 10240. The experiments use bfloat16 execution on an NVIDIA GB10 system.

CPU benchmark scope. Table 1 reports one FP32 CPU microbenchmark configuration with batch size 128. The manuscript uses only the 0.77/0.93/2.64 ms measurements from that single setup and does not combine them with timing numbers from other runs or configurations.

Domain adaptation. The initial cells specialize to Python and SQL. Subsequent hot-plug experiments add Math, Linux, C++, Java, Network, and Git cells. Small synthetic "golden" adaptation sets are used for the added domains. The eight-cell sequential stage uses 38 C++ examples, 41 Java examples, 39 Network examples, and 41 Git examples. Where an evaluation below is explicitly called held-out, its prompts are separated from the corresponding adaptation data. The 160-prompt eight-cell result is an exception: its prompt pool overlaps the routing-calibration material, so it is reported only as a post-calibration diagnostic rather than untouched blind evidence.

Boundary calibration. The eight-cell global calibration uses 189 routing-calibration prompts and 41,000 trainable routing parameters, approximately 0.000128% of a nominal 32B-parameter backbone. All heavy expert parameters are frozen during this stage.

Metrics. We report routing accuracy, routing margin, old-domain retention, new-domain acquisition, wall-clock latency, throughput, peak allocated CUDA memory, sampled GPU power, and derived energy per token. Routing accuracy measures expert selection, not downstream answer correctness.

Table 1: CPU conditional-execution ablation.

Configuration	Active Cells	Latency (ms)
Full ACNA	1.00/5	0.77
w/o Social Intervention	1.00/5	0.93
w/o Resting Potential	5.00/5	2.64

Table 2: Capacity scaling with one active heavy expert.

Total Cells N	Active Cells K	Latency (ms)
5	1.00	0.902
10	1.00	0.961
20	1.00	1.221
50	1.00	2.294
100	1.00	5.269

RTL feasibility prototype. Using Icarus Verilog, we simulate three independently gated compute cells, one-hot task-to-cell selection, and a fourth hot-plug cell. The activity benchmark compares 100 tokens under dense three-cell execution against one-active-cell sparse execution. Reported "activity" is a registered compute-event count used as a switching proxy; it is not measured electrical power.

6 Results

6.1 Conditional-Execution CPU Ablation

Removing the Resting Potential yields five active cells and increases latency from 0.77 to 2.64 ms. Relative to the fully active ablation, the sparse configuration reduces measured wall-clock latency by 70.8%. This is a compute-skipping result, not an electrical-energy measurement.

6.2 Capacity Scaling

Across a 20× increase in capacity, $K = 1$ throughout. Total latency still rises because the $O(Nd)$ receptor and $O(Nr)$ interaction terms grow with N, consistent with Eq. 12.

6.3 32B Two-Domain Routing

A two-cell ACNAbranch is inserted into Qwen2.5-Coder-32B-Instruct. On 40 held-out domain prompts, Python routing is 19/20, SQL routing is 20/20, and overall routing accuracy is 39/40 (97.5%). Because $g \approx 0.01062$, this result demonstrates large-model routing specialization but not complete replacement of the pretrained MLP.

6.4 Reversible Math Hot-Plug and Hot-Unplug

A third Math cell is appended while the Python and SQL cells are frozen. After cell-local rejection calibration, a 60-prompt held-out evaluation reaches 20/20 Python, 20/20 SQL, and 20/20 Math routing. The Math cell is then physically removed with no retraining. The remaining two-cell system returns 20/20 Python and 20/20 SQL routing (40/40 overall). This provides direct evidence for reversible module insertion/removal at the routing and expert-block level.

6.5 Mixed-Domain Routing

All 60 mixed-domain prompts place the winner inside the intended pair, and both intended cells occupy the Top-2 in every case. This result is consistent with multi-domain score structure rather than purely one-label memorization.

6.6 Sequential Expansion to Four and Eight Cells

Adding Linux as a fourth cell initially creates behavioral interference: the new cell wins many Python and SQL prompts even though old parameters have not changed. Before calibration, old-domain retention falls to 61.67%. A cell-local Linux boundary calibration restores Python, SQL, Math, and

Table 3: Mixed-domain routing. "Pair winner" means the winning cell belongs to the intended domain pair.

Pair	Pair Winner	Pair in Top-2	Avg. Margin
Python + Math	100%	100%	0.2767
SQL + Math	100%	100%	0.5951
Python + SQL	100%	100%	0.1939

Table 4: Eight-cell post-calibration routing evaluation.

Domain	Correct	Accuracy	Avg. Margin
Python	20/20	100.00%	0.6099
SQL	20/20	100.00%	0.6872
Math	20/20	100.00%	0.9581
Linux	20/20	100.00%	0.6934
C++	19/20	95.00%	0.6525
Java	20/20	100.00%	0.7367
Network	20/20	100.00%	0.7363
Git	20/20	100.00%	0.7109
Overall	159/160	99.38%	–

Linux to 20/20 each, while previously installed expert modules remain unchanged.

At eight cells, the same phenomenon reappears more strongly. Sequential insertion alone achieves 137/160 (85.62%) overall routing accuracy, 60/80 (75.0%) retention on the original four cells, and 77/80 (96.25%) acquisition on the new four. This shows that "parameter freezing" does not imply "behavioral non-interference."

Global routing-boundary calibration then adjusts 41,000 routing parameters while keeping all heavy experts frozen. The resulting system reaches the performance in Table 4.

Because the 160-prompt evaluation pool overlaps material used by the global routing calibration, these numbers quantify post-calibration fit/retention rather than untouched generalization; we therefore do not call this evaluation blind or held-out.

The original four domains retain 80/80 (100%) accuracy, while the four new domains achieve 79/80 (98.75%). All eight domains are at or above 95%. The single remaining error is a C++ prompt routed to Java.

6.7 GPU Physical-Sparsity Microbenchmark

To isolate heavy expert execution from the transition MLP, we benchmark the eight trained expert blocks outside end-to-end 32B generation. This is therefore an *expert-block microbenchmark*, not a whole-model speedup claim. The input shape is $[1, 32, 5120]$, with eight experts and Top-1 per-token execution. We compare dense execution of all eight experts against ACNAsparse dispatch. Results are shown in Table 5.

Routing-only latency is 0.1213 ms. Per token, only one of eight experts executes, corresponding to 12.5% active-expert density and 87.5% logical expert sparsity. However, the 32-token batch collectively touches seven of eight experts. The implementation therefore launches work for most expert identities at the batch level despite Top-1 per-token sparsity.

6.8 RTL Cell-Gating and Hot-Plug Feasibility

A compact RTL prototype maps ACNAactivation to independently enabled compute cells. One-hot routing correctly activates exactly one of three cells for task IDs 0–2. A logical hot-plug test begins with activity counts $[10, 10, 10, 0]$ for three enabled cells plus a pre-instantiated dormant fourth cell; after enabling the fourth cell and routing a new task to it, counts become $[20, 20, 10, 10]$, showing that the newly available cell accepts work while existing modules remain available. This RTL test validates runtime availability gating, not FPGA partial reconfiguration.

Table 5: NVIDIA GB10 expert-block benchmark (PyTorch 2.13.0+cu130, CUDA 13.0).

Metric	Dense-8	ACNATop-1	Change
Mean latency	8.1450 ms	7.4574 ms	-8.44%
Throughput	3928.79 tok/s	4291.03 tok/s	$+9.22\%$
Peak CUDA memory	1633.96 MB	1633.12 MB	≈ 0
Average power	23.45 W	26.46 W	$+12.84\%$
Energy/token	0.00596876 J	0.00616693 J	$+3.32\%$

For equal 100-token workloads, dense execution activates all three cells per token (300 registered compute events), whereas sparse one-cell execution produces 100 events ($34 + 33 + 33$), a 66.7% activity reduction. This verifies the hardware control mechanism required for energy-proportional execution, but does not itself measure watts or joules.

7 Hardware Interpretation

The GPU and RTL results answer different questions. On GB10, ACNAreduces expert-block latency by 8.44% but increases measured energy/token by 3.32%. The 32-token batch touches seven of eight experts, so token-wise sparsity is fragmented into masks, gathers/scatters, small GEMMs, memory traffic, and multiple kernel launches. A software-skipped expert is also not equivalent to an independently power-gated hardware island.

The RTL prototype shows that the activation decision can directly control hardware activity: with the same 100-token workload, registered compute activity falls from 300 dense events to 100 sparse events. This does *not* prove a 66.7% reduction in electrical energy; static power, routing, memories, clock trees, and control logic remain. It instead verifies the mechanism by which a cell-aware FPGA/ASIC/NPU could suppress inactive computation. Prior sparse accelerators show that energy benefits become larger when sparsity, memory movement, scheduling, and compute hardware are co-designed (Han et al., 2016; Parashar et al., 2017). Thus, ACNAprovides an explicit active-cell signal that future hardware can map to grouped dispatch, local memories, clock enables, and—where practical—fine-grained power gating.

8 Limitations

The present evidence is intentionally scoped as a proof of feasibility. First, routing accuracy is not equivalent to downstream task accuracy. Where held-out routing is reported, it measures whether the intended Expert Cell wins and does not constitute a standard language-model benchmark result. The eight-cell 159/160 result is specifically a post-calibration diagnostic because that prompt pool overlaps routing-calibration material; a fresh untouched eight-domain evaluation remains required for a generalization claim. Second, the 32B transition gate remains near $g = 0.01062$, so the pretrained MLP still dominates the modified layer. Complete replacement requires experiments in which g approaches one while perplexity, generation quality, and task performance remain stable.

Third, the current fixed-budget implementation still uses a global Top-1 comparison for dispatch. Thus, ACNAremoves a separately learned centralized router network but not every global selection/synchronization primitive. Fourth, the eight-domain expansion requires routing-boundary calibration after sequential insertion. This is evidence that modular expert freezing does not automatically solve behavioral interference.

Finally, the GPU measurement is an isolated microbenchmark with short power-sampling windows, and the RTL experiment measures activity rather than electrical power. Neither is a full-chip or end-to-end 32B energy study.

9 Conclusion

We introduced ACNA, a sparse conditional-computation architecture that places activation evidence inside independently parameterized Expert Cells and combines local receptors, Resting Potentials, low-rank Social Intervention, hard expert skipping, and a function-preserving transition path for pretrained Transformers.

The experimental sequence supports the architectural feasibility of modular active-cell computation. Sparse execution reduces CPU wall-clock work; active heavy-expert count remains fixed while available capacity grows; a 32B model develops domain-correlated routing; a Math cell can be inserted and later removed without retraining the remaining cells; mixed-domain prompts preserve the intended expert pair; and sequential expansion to eight domains reaches 99.38% post-calibration routing accuracy with 100% accuracy on the original four domains after lightweight routing calibration. Because the eight-cell evaluation pool overlaps calibration material, this number is not presented as untouched blind accuracy.

On GPU, Top-1 expert execution yields real latency and throughput gains but not present energy savings. RTL simulation independently shows that the same sparse activation semantics can reduce registered compute activity by 66.7% when inactive cells are explicitly gated. Together, these results distinguish algorithmic sparsity from electrical energy proportionality: ACNAsupplies the active-cell control signal and avoids heavy computation, while future cell-aware hardware must translate that signal into efficient dispatch, memory locality, clock gating, and, where practical, power gating. The present work therefore establishes feasibility and an experimentally grounded hardware path without claiming a measured FPGA/ASIC energy reduction.

Reproducibility Statement

The main text specifies the architecture equations, transition mechanism, expansion losses, model backbone, hidden dimensions, principal dataset sizes, calibration parameter count, and hardware benchmark configuration. For final submission, the supplementary archive should include exact training/calibration/evaluation scripts, the 189-prompt calibration set, a newly created untouched eight-domain blind set, synthetic adaptation datasets, checkpoint migration logic, random seeds, optimizer settings, and raw CPU/GPU/RTL logs. The accompanying internal package contains a completeness manifest and templates, but missing experimental source files are not fabricated; they must be copied from the experiment machine before submission.

AI Use Statement

Generative AI tools were used in this work to assist with conceptual refinement, feedback on experimental design, code drafting and debugging, generation of small synthetic domain-adaptation datasets, interpretation and organization of experimental results, literature discovery, and manuscript drafting/editing. All reported experiments were executed by the authors on local hardware; generated code and data were reviewed before use; reported numerical results were taken from executed experiment logs; and the authors reviewed all AI-assisted text and claims. The authors take responsibility for the final content of the work, including text, code, experimental design, and conclusions.

References

N. Shazeer et al. *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*. ICLR, 2017.

D. Lepikhin et al. *GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding*. arXiv:2006.16668, 2020.

W. Fedus, B. Zoph, and N. Shazeer. *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*. JMLR 23(120):1–39, 2022.

Y. Zhou et al. *Mixture-of-Experts with Expert Choice Routing*. arXiv:2202.09368, 2022.

Y. Liu, J. Han, S. Yan, V. Tresp, and Y. Ma. *Routing-Free Mixture-of-Experts*. arXiv:2604.00801, 2026.

A. A. Rusu et al. *Progressive Neural Networks*. arXiv:1606.04671, 2016.

C. Fernando et al. *PathNet: Evolution Channels Gradient Descent in Super Neural Networks*. 2017.

J. Pfeiffer et al. *AdapterFusion: Non-Destructive Task Composition for Transfer Learning*. EACL, 2021.

A. Candocia, O. M. Inan, R. Agarwal, A. Varma, and M. A. Davenport. *Scalable Continual Learning: Adaptive MoEs for Expanding Task Sets*. OpenReview, 2025.

S. Han et al. *EIE: Efficient Inference Engine on Compressed Deep Neural Network*. ISCA, 2016.

A. Parashar et al. *SCNN: An Accelerator for Compressed-Sparse Convolutional Neural Networks*. ISCA, 2017.

NVIDIA. *Structured Sparsity in the NVIDIA Ampere Architecture and Applications in Search Engines*. NVIDIA Developer Blog, 2023.

B. Hui et al. *Qwen2.5-Coder Technical Report*. arXiv:2409.12186, 2024.

A Additional Experimental Details

A.1 Eight-Cell Expansion Sequence

The evaluated sequence is:

$$\begin{aligned}\text{Python} &\to +\text{SQL} \to +\text{Math} \to +\text{Linux} \\ &\to +\text{C++} \to +\text{Java} \to +\text{Network} \to +\text{Git}.\end{aligned}$$

The Math hot-plug stage freezes previously installed Python/SQL modules. The Linux stage uses cell-local calibration to prevent the new cell from stealing old-domain prompts. The final eight-cell stage freezes all heavy experts during global boundary calibration.

A.2 Pre- and Post-Calibration Eight-Cell Results

The "after calibration" column below uses the same 160-prompt pool discussed in the main text and is therefore a post-calibration diagnostic, not an untouched blind test. A final untouched blind set should be generated only after calibration is frozen and should never be used for subsequent tuning.

Table 6: Effect of global routing-boundary calibration.

Metric	Before Calibration	After Calibration
Overall routing	137/160 (85.62%)	159/160 (99.38%)
Original 4-cell retention	60/80 (75.00%)	80/80 (100.00%)
New 4-cell acquisition	77/80 (96.25%)	79/80 (98.75%)
Domains $\geq 90\%$	5/8	8/8

A.3 GPU Benchmark Notes

The GPU benchmark uses 30 warm-up iterations and 100 timed iterations. CUDA events are synchronized for per-iteration latency measurement. Power is sampled through `nvidia-smi` at approximately 50 ms intervals. Because the timed workload is short and power telemetry is sampled at a coarser timescale, the power/energy numbers should be interpreted as preliminary system-level evidence rather than precision electrical characterization.

A.4 RTL Gating Notes

The RTL prototype uses three compute cells plus a pre-instantiated fourth cell whose availability is controlled at runtime. Functional tests verify one-hot task routing and logical cell enablement; they do not implement FPGA partial reconfiguration. For the equal-work activity test, dense execution produces 300 registered compute events and sparse execution 100 events. These counts are switching/activity proxies only; actual FPGA board power should be measured in future work with rail- or board-level power instrumentation.