KALAVAI: When Does Independent Specialist Fusion Work? Conditions for Post-Hoc Cooperative LLM Training

Ramchand Kumaresan
Murai Labs

Abstract

We study the conditions under which independently trained domain specialists, fused post-hoc via a mixture-of-experts router, outperform both individual specialists and equal-compute monolithic training. In the KALAVAI[1] protocol, contributors each fine-tune a copy of a shared base checkpoint on their own data without communication, then submit checkpoints for lightweight MoE routing (500 gradient steps on mixed data).

We identify three governing conditions: (1) *Shared initialisation is necessary*: specialists initialised from different checkpoints are not fusible; the shared starting point preserves representational compatibility. (2) *Frozen layers become necessary beyond ≈10,000 training steps*: at short horizons (≤5,000 steps) freezing is optional; beyond ≈10,000 steps, it prevents over-specialisation that makes fusion degrade. (3) *Joint inference is required*: single-specialist dispatch with 99.3%-accurate domain classification degrades performance by -21.1%; all specialists must run in parallel.

Supporting evidence: the fused model beats the best individual specialist by +14.2% (410M, 3 seeds), +14.8% (1B), and +2.4% (6.9B), and outperforms equal-compute monolithic training by +14.5%.

Boundary conditions: Qwen-1.5B at full training shows -1.0% (model-family-specific); 6.9B improvement is reduced due to a $\sim 34\times$ weaker per-parameter training signal relative to 410M. We do not claim inference efficiency (the fused model is $\approx N\times$ more expensive than a single specialist), universal architecture generality, or that downstream benchmarks reliably improve.

1 Introduction

Training competitive language models requires centralised compute at a scale most researchers and institutions cannot access. A single 70B-parameter training run requires hundreds of A100 GPUs operating synchronously for weeks. This creates a structural barrier: the organisations that can train frontier models are those that can pay for frontier compute, and the rest must fine-tune what is already available.

The core insight. A different path exists. If multiple contributors each train a *specialist* copy of a shared base checkpoint on their own domain, and if those specialists are subsequently fused via a learned router, the resulting model captures complementary knowledge that no single contributor could build alone. The training step requires no communication: contributors work independently, asynchronously, on their own hardware, using their own data. The only coordination is the shared starting point. The fused model requires all N specialists to run at inference, increasing inference

[1]KALAVAI (*kalaVAI*) is the ISO 15919 romanisation of the Tamil word for “fusion” or “mixing”.

compute by a factor of N relative to any individual specialist. We view this as a training-time democratisation that trades inference efficiency for training accessibility.

This observation is not entirely new—the branch-train-mix (BTX) paradigm [Sukhbaatar et al., 2024] demonstrates that MoE fusion of independently trained models is feasible. What is missing from the literature is an empirical characterisation of *the conditions under which independent specialist fusion succeeds or fails*: when shared initialisation alone is sufficient, when frozen layers become necessary, and what routing architecture drives improvement.

The protocol. KALAVAI[2] operationalises cooperative LLM training as a four-step protocol: (1) a coordinator distributes a shared base checkpoint; (2) each contributor fine-tunes independently on their domain for a fixed number of steps; (3) contributors submit their checkpoints; (4) a lightweight router is trained on a small mixed-domain dataset and used for inference. Contributors never share gradients, intermediate activations, or data. The only shared artefact is the initial checkpoint.

Key results. We demonstrate:

1. **Consistent improvement at scale.** Post-hoc MoE fusion beats the best individual specialist by +14.2% at 410M (3 seeds, $\pm 0.016\%$), +14.8% at 1B (3 seeds, $\pm 0.003\%$), and +2.4% at 6.9B parameters (3 seeds, $\pm 0.00\%$). Improvement is stable across Pythia training maturities from 3.5% to 100% of training.
2. **Training duration crossover.** Without frozen layers, fusion improvement peaks at approximately 5,000 specialist training steps (+16.4%) then degrades to +13.6% at 20,000 steps as specialists overfit their domains. With four frozen layers, improvement plateaus at +14.8% through 20,000 steps. The crossover occurs at approximately 10,000 steps.
3. **Specialists must run jointly.** Routing to a single specialist based on a near-perfect (99.3% accurate) domain classifier *degrades* performance by 21.1%. Joint inference with all specialists and softmax-weighted combination improves it by 14.2%. Improvement requires specialists to collectively process each token, not classification.
4. **Beats equal-compute monolithic training.** A single model trained on mixed data for the same total compute as three specialists achieves +6.7% over base. KALAVAI achieves +14.5% over the same monolithic model.

Contributions. This paper provides: (i) an empirical account of why post-hoc MoE fusion works (shared initialisation preserves representational compatibility; individual specialists exhibit catastrophic forgetting on out-of-domain tokens; joint inference restores coverage); (ii) a practical guideline for when frozen layers are necessary; (iii) controlled capacity comparisons ruling out parameter count as the mechanism; and (iv) the KALAVAI protocol, which requires zero communication during training.

2 Related Work

Branch-Train-Mix (BTX). The closest prior work is BTX [Sukhbaatar et al., 2024], which demonstrates that models branched from a shared checkpoint, independently trained, and mixed via MoE routing form a better model than any individual branch. Specifically, BTX does not characterise: (i) when shared initialisation alone is sufficient versus when frozen layers become necessary (our training-duration crossover, Section 4.4); (ii) whether the improvement persists against a compute-matched monolithic baseline (Section 4.3); (iii) why single-specialist dispatch fails despite near-perfect domain classification (Section 4.5); or (iv) whether the improvement is explained by increased parameter count (Section 4.6). These four questions define our empirical contribution. We note that BTX demonstrates fusion at comparable model scales with substantially longer specialist training budgets; our contribution is the empirical characterisation of when and why fusion succeeds or fails, not a performance claim over BTX.

MoErging and PHATGOOSE. The MoErging survey [Yadav et al., 2024] taxonomises approaches for recycling and routing among independently trained experts. PHATGOOSE [Muqeeth et al., 2024]

[2]Code and all experiment scripts: `https://github.com/mechramc/Kalavai`

achieves +11% zero-shot generalisation improvement via learned routing among fine-tuned models, compared to KALAVAI's +14.2% held-out improvement with task-specific training. KALAVAI adds a monolithic baseline, training duration analysis, and explicit capacity controls not present in PHATGOOSE.

Pari thesis. Pari [2025] provides a theoretical analysis using Centered Kernel Alignment (CKA) of why weight averaging of independently trained models fails: divergent representations produce destructive interference when merged by linear interpolation. KALAVAI provides the empirical complement, demonstrating that MoE routing avoids this interference (+14.2% improvement versus weight averaging's +4.0%).

Weight interpolation methods. Model soups [Wortsman et al., 2022], TIES-Merging [Yadav et al., 2023], and DARE [Yu et al., 2024] combine fine-tuned models via weight interpolation. These methods require specialised merging procedures and typically produce smaller gains than routing-based approaches. In our experiments, simple weight averaging achieves +4.0% versus +14.2% for learned MoE routing.

Federated learning. Federated approaches [McMahan et al., 2017] distribute training with periodic gradient synchronisation. KALAVAI requires *zero communication* during training; contributors are never synchronised until the fusion step, making the protocol fully asynchronous.

FuseLLM. FuseLLM [Wan et al., 2024] fuses LLMs via knowledge distillation into a modified single model; KALAVAI preserves all specialist parameters intact.

Sparse Upcycling. Komatsuzaki et al. [2023] initialise MoE models from dense checkpoints and continue training jointly; KALAVAI trains specialists *independently* with no shared computation.

STAR and related concurrent work. Qin et al. [2025] demonstrate modular composition over frozen foundations for multimodal learning; KALAVAI provides the language modelling instantiation with analysis of the crossover point where freezing transitions from optional to required.

3 Method

The KALAVAI protocol consists of four phases.

Phase 1: Shared initialisation. A coordinator selects a publicly available base checkpoint θ_0 and distributes it to all contributors. All specialists begin from *identical* weights. This shared initialisation is the core structural guarantee that enables post-hoc fusion: specialists diverge in representation space, but their representational geometry remains compatible because they begin from the same point. Hash verification ensures all contributors use exactly the same checkpoint.

Phase 2: Optional freezing. Optionally, the first K transformer layers are frozen during specialist training. Frozen layers guarantee that lower-level representations remain shared across specialists, providing a structural anchor that is robust to extended training. Our experiments show freezing is unnecessary at short training horizons ($\leq$5,000 steps) but becomes beneficial beyond approximately 10,000 steps (Section 4.4).

Phase 3: Independent specialist training. Each contributor trains their copy of θ_0 on a single knowledge domain using standard next-token prediction loss. Training is fully independent: contributors never share data, gradients, or activations. Any training infrastructure, hardware, or optimiser the contributor prefers may be used, provided the architecture and freeze configuration match the coordinator's specification. We use full fine-tuning of unfrozen layers rather than low-rank adaptation (LoRA); preliminary experiments with LoRA showed insufficient specialist divergence to produce fusion gains.

Formally, contributor i trains specialist θ_i by minimising:

$$\mathcal{L}_i = -\mathbb{E}_{x\sim\mathcal{D}_i}\left[\sum_t \log p_{\theta_i}(x_t \mid x_{<t})\right]$$

where $\mathcal{D}_i$ is contributor i's domain-specific dataset and θ_i shares the first K frozen layers with all other specialists.

Phase 4: Post-hoc MoE fusion. After all specialists submit their checkpoints, a lightweight router is trained on a small mixed-domain dataset (500 gradient steps in our experiments). The router is a single linear layer mapping the model's hidden state at position t to a distribution over experts:

$$g_t = \mathrm{softmax}\left(W_r \cdot h_t\right), \quad W_r \in \mathbb{R}^{N \times d}$$

where h_t is the hidden state at layer $K+1$ (first non-frozen layer)[3], N is the number of specialists, and d is the hidden dimension. At inference, all N specialists process each token in parallel and the output is a weighted combination of their logit distributions:

$$p_{\text{fused}}(x_t \mid x_{<t}) = \sum_{i=1}^{N} g_t^{(i)} \cdot p_{\theta_i}(x_t \mid x_{<t})$$

Design decisions—LoRA vs. full fine-tuning, softmax vs. argmax, linear vs. MLP router—are discussed in Appendix C; none meaningfully affects the core results.

4 Experiments

4.1 Experimental Setup

Models. We run experiments at three scales: Pythia-410M (24 layers, hidden size 1024), Pythia-1B (16 layers, hidden size 2048), and Pythia-6.9B (32 layers, hidden size 4096) [Biderman et al., 2023]. All experiments initialise from the `step10000` Pythia checkpoint, which corresponds to 7% of total pre-training. We use Pythia because it releases checkpoints at multiple training stages, enabling the maturity sweep analysis.

Domains. Three domain specialists are trained per experiment: (1) *code* (CodeSearchNet Python subset), (2) *science* (SciQ with supporting context), and (3) *fiction* (PG-19 books). For each domain, 90% of samples are used for specialist training, 10% are held out and never seen during training or router training.

Training configuration. All 410M and 1B experiments: 2,000 specialist training steps (effective batch size 8, sequence length 512), 500 router training steps. 6.9B experiments: 1,000 specialist training steps, 500 router steps. Freeze depth $K = 4$ for 410M (4/24 = 17%), $K = 4$ for 1B (4/16 = 25%), $K = 6$ for 6.9B (6/32 = 19%). Optimiser: AdamW, lr $= 2 \times 10^{-5}$, weight decay 0.1, linear warmup over 10% of steps.

Evaluation metric. All improvement percentages are computed as:

$$\Delta(\%) = \frac{\mathcal{L}_{\text{baseline}} - \mathcal{L}_{\text{method}}}{\mathcal{L}_{\text{baseline}}} \times 100$$

where $\mathcal{L}$ is average cross-entropy loss on the held-out mixed-domain evaluation set (equal weighting across three domains). Lower loss is better; positive Δ indicates improvement over the baseline. The baseline for the main result is the best individual specialist on mixed evaluation; the baseline for the monolithic comparison is the monolithic model.

Seeds. All main results are reported across 3 random seeds (42, 137, 2026).

Evaluation consistency. Base model loss varies slightly across experimental setups (e.g., 2.248 in Table 1 vs. 2.420 in the dispatch experiments) due to different evaluation batch compositions. All improvement percentages are computed within the same evaluation setup; cross-table loss values should not be compared directly.

Table 1: Main results across three model scales. Improvement is computed against the best individual specialist on held-out mixed-domain evaluation. All results use 3 random seeds; 6.9B uses corrected seeded evaluation (see "Note on 6.9B evaluation" paragraph in Section 4.2). Base losses: 410M 2.248, 1B 2.160, 6.9B 2.700.

Scale	Method	Mixed Loss	vs. Best Spec.	vs. Base	Seeds	Std
Pythia-410M	Base model	2.248	—	—	—	—
	Best specialist	2.089	—	+7.1%	3	±0.00%
	Weight averaging	2.158	−3.3%	+4.0%	3	±0.00%
	Monolithic baseline	2.098	—	+6.7%	3	±0.00%
	KALAVAI (MoE)	**1.793**	**+14.2%**	+20.2%	3	±0.016%
Pythia-1B	Base model	2.160	—	—	—	—
	Best specialist	1.992	—	+7.8%	3	±0.00%
	KALAVAI (MoE)	**1.696**	**+14.8%**	+21.5%	3	±0.003%
Pythia-6.9B	Base model	2.700	—	—	—	—
	Best specialist	2.634	—	+2.5%	3	±0.00%
	KALAVAI (MoE)	**2.570**	**+2.4%**	+4.8%	3	±0.00%

4.2 Core Results

Table 1 presents the main results. KALAVAI consistently outperforms the best individual specialist at all tested scales. The 410M improvement of +14.2% and 1B improvement of +14.8% are robust: variance across three random seeds is near-zero (±0.016% and ±0.003%, respectively), and results are stable across Pythia training maturities from step 5,000 to step 143,000 (see Appendix I).

The 6.9B improvement of +2.4% is smaller than at smaller scales. We attribute this primarily to the reduced specialist training budget: 1,000 steps at 6.9B versus 2,000 steps at 410M and 1B, while the model has 17× the parameters. The per-parameter specialist training signal is therefore substantially weaker, limiting the specialist divergence that enables fusion gains. The maturity analysis at 6.9B (step 10,000: +2.43%; step 143,000: +2.26%) confirms the effect is stable across training stages—the improvement does not collapse at higher maturity—suggesting the limiting factor is the training step budget rather than pre-training saturation (Appendix I).

Note on 6.9B evaluation. An initial 6.9B run using unseeded data shuffling produced inconsistent per-seed results (mean +2.72%, std ±8.17%) due to non-deterministic chunk selection. After fixing the evaluation to use a seeded shuffle, all three seeds converge to +2.43%–+2.44% (mean +2.43%, std ±0.00%). All 6.9B results in this paper use the corrected evaluation.

4.3 Comparison to Equal-Compute Monolithic Training

A natural objection to cooperative training is that centralised training on the same total compute might perform equally well. We test this directly. A single model is fine-tuned from the same base checkpoint for 6,000 steps (equal to three specialists × 2,000 steps) on a mixed dataset containing equal proportions of code, science, and fiction data. This monolithic baseline achieves +6.7% over the base model (loss: 2.098), reflecting gradient interference from mixed-domain training [Sukhbaatar et al., 2024].

KALAVAI with three 2,000-step specialists achieves +14.5% over this monolithic baseline (Figure 4, Table 2). The decomposition reveals two separable effects: (1) *Specialisation advantage*: the best individual specialist (+7.1% vs. base) marginally exceeds the monolithic model (+6.7% vs. base), contributing ≈0.4pp; (2) *Fusion advantage*: the router then selects the best specialist per token, contributing the remaining ≈7.1pp gap over the monolithic model.

[3]In our implementation, h_t is the mean-pooled final hidden state averaged across all specialists' forward passes; see Appendix K for implementation details.

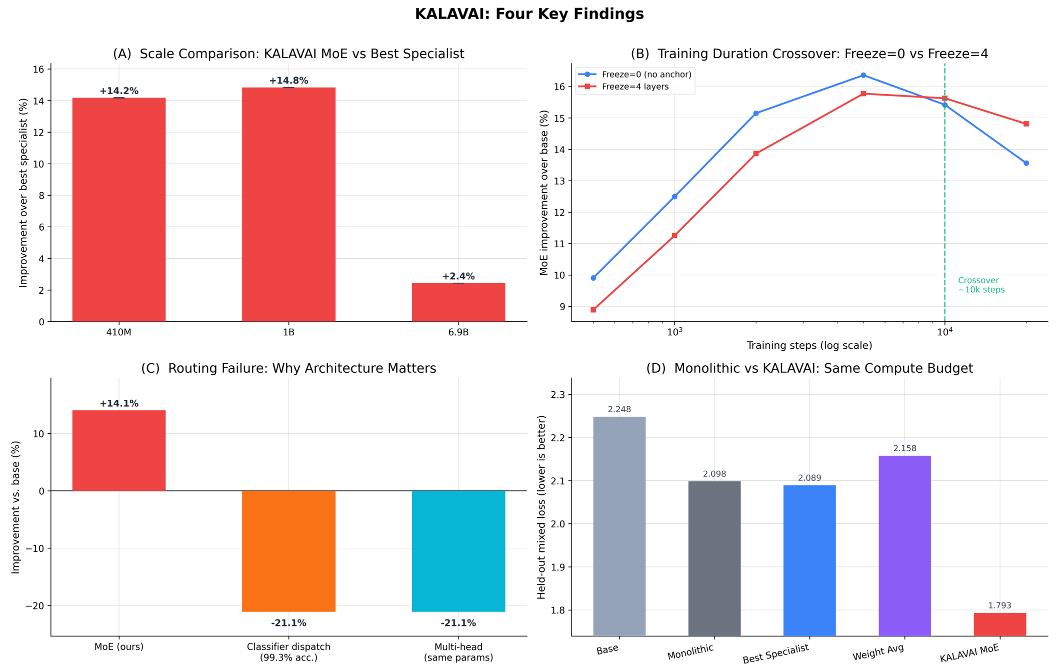

Figure 1: **KALAVAI core results. (A)** Fusion improvement over the best individual specialist across model scales: +14.2% at 410M, +14.8% at 1B, +2.4% at 6.9B. **(B)** Training duration crossover: freeze=0 peaks at 5k steps then degrades; freeze=4 plateaus through 20k steps; crossover at $\approx$10k steps. **(C)** Routing failure modes: learned MoE routing achieves +14.1%; single-specialist classifier dispatch degrades by -21.1% (negative values indicate degradation below base). **(D)** KALAVAI vs. equal-compute alternatives at 410M: MoE achieves +14.5% over the equal-compute monolithic baseline. All results are means over 3 seeds.

Table 2: Equal-compute comparison at Pythia-410M. Monolithic is trained for 6,000 steps on mixed data. KALAVAI uses three specialists trained for 2,000 steps each. All 3 seeds; compute matched. Improvement columns use the base model and monolithic model as baselines respectively.

Method	**Loss**	**vs. Base**	**vs. Monolithic**
Base model	2.248	—	—
Monolithic (6000 steps, mixed)	2.098	+6.7%	—
Best specialist (code, 2000 steps)	2.089	+7.1%	+0.4%
KALAVAI MoE	**1.793**	+20.2%	**+14.5%**

4.4 Training Duration and the Role of Frozen Layers

A key design question is whether frozen layers are necessary. The freeze depth sweep (Appendix E) shows only 2.5 percentage points of variation across freeze depths from 0 to 50% of layers, suggesting freezing is largely optional at 2,000-step training horizons. However, this masks a training duration dependence.

Table 3 (and Figure 3 in Appendix D) show fusion improvement as a function of specialist training duration. Without frozen layers, improvement peaks at 5,000 steps (+16.4%) and then degrades monotonically to +13.6% at 20,000 steps. With four frozen layers, improvement plateaus at +15.8% through 20,000 steps. The crossover occurs at approximately 10,000 training steps: above this threshold, frozen layers prevent over-specialisation that degrades post-hoc fusion.

Practical guideline: For specialist training up to 5,000 steps, frozen layers are unnecessary. For training horizons beyond 5,000 steps (and certainly beyond 10,000), freezing the first K layers is recommended. The optimal K is not sensitive: the freeze depth sweep shows only 2.5pp variation across $K \in \{0, 2, 4, 6, 8, 12\}$ at 2,000 steps (Appendix E).

Table 3: Fusion improvement vs. best specialist as a function of training duration, with and without frozen layers. Pythia-410M, seed 42. Bold entries indicate the better configuration at each step count; † marks the crossover point.

Steps	Freeze=0	Freeze=4	Leader
500	+9.9%	+8.9%	**Freeze=0**
1,000	+12.5%	+11.3%	**Freeze=0**
2,000	+15.1%	+13.9%	**Freeze=0**
5,000	+16.4%	+15.8%	**Freeze=0**
10,000†	+15.4%	+15.6%	**Freeze=4**
20,000	+13.6%	+14.8%	**Freeze=4**

4.5 All Specialists Must Run: Single-Specialist Dispatch Fails Catastrophically

A natural simplification would route each input to a single specialist. We test this with a logistic regression classifier achieving 99.3% accuracy on held-out inputs. The result is catastrophic: single-specialist dispatch degrades performance by -21.1% relative to base, compared to +14.1% for joint MoE inference. A multi-head baseline with identical parameter count produces the same -21.1% degradation. The mechanism is catastrophic forgetting: each specialist loses accuracy on non-specialist domains, and single-specialist dispatch provides no fallback. Hard routing (argmax, all experts run) achieves +14.1%—identical to soft routing—confirming that specialist participation, not weighting precision, drives improvement. Full results in Appendix H.

4.6 Capacity Controls: Parameter Count Is Not the Mechanism

The fused model has $3\times$ the unfrozen parameters of any individual specialist. To rule out parameter count as the mechanism, we train Pythia-1.4B ($3.5\times$ total parameters of a 410M specialist) for 6,000 steps on mixed data; it achieves +5.9%—less than half the KALAVAI improvement. The multi-head baseline (same parameter count as MoE, classifier-selected head) achieves -21.1%. The improvement requires specialised representations combined via joint inference, not additional capacity. Full comparison in Appendix H.

5 Analysis

Router architecture does not matter. A uniform router (fixed weights of $1/N$ per expert, no training) achieves +6.7% over base—the same as the monolithic baseline. A linear router achieves +14.2%; a 2-layer MLP router achieves an identical +14.2%. The gap between uniform and learned routing (+7.5pp) is entirely explained by the router's ability to assign domain-appropriate weights; the specific function class used to compute those weights is irrelevant. Full results in Appendix G.

Improvement is robust across training maturities. Fusion improvement is consistent across Pythia checkpoints from step 5,000 to step 143,000 at both 410M (+13.4%–+15.0%) and 1B (+13.8%–+15.9%); the mechanism does not depend on the base model being under-trained (Appendix I).

Improvement is robust across specialist count. Scaling from 2 to 5 specialists shows no meaningful degradation: 3–5 specialists all achieve $\approx$+14.1% with near-zero variance across seeds (Appendix J).

Token-level routing confirms mid-sequence switching. On hybrid-domain prompts, the router produces 2.2 expert switches per prompt on average, assigning domain-appropriate weights within a single sentence—confirming the router operates at token granularity, not document level (Appendix N).

Representational divergence confirms specialisation. Figure 2 shows the cross-domain evaluation loss matrix at step 2,000. The pronounced diagonal structure confirms that each specialist has learned domain-specific representations: each specialist achieves its lowest loss on its own domain and highest loss on the furthest domain. The code specialist evaluates at 1.879 on code data and 2.909

on science data—a gap of 0.032 above base on science, confirming out-of-domain degradation. The off-diagonal pattern directly motivates MoE fusion: a router that dispatches each token to the appropriate diagonal entry recovers all specialist gains.

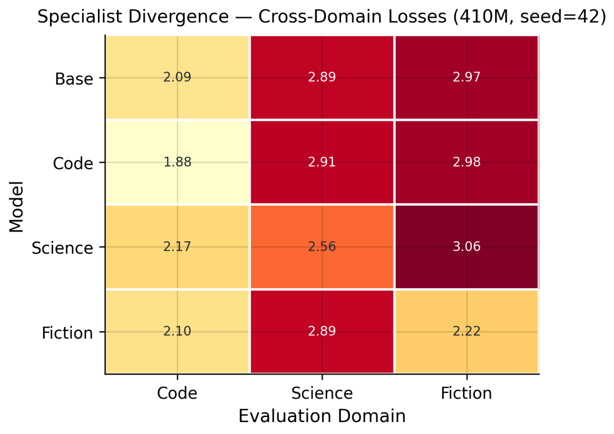

Figure 2: Cross-domain evaluation loss matrix at Pythia-410M, step 2,000 (freeze=4, seed=42). Rows are specialists; columns are evaluation domains. The diagonal entries (own-domain performance) are lower than off-diagonal (cross-domain), confirming that each specialist has diverged in a complementary direction. The MoE router recovers diagonal performance across all domains simultaneously. Color scale: green indicates lower loss (better performance), red indicates higher loss.

Downstream benchmarks. At 1B scale, MoE leads on HellaSwag (35.0% vs. 34.4% for the base) and best individual specialist (34.2–34.4%). Monolithic training produces the worst HellaSwag score (33.4%). Overall average accuracy: base 50.6%, MoE 49.6%, monolithic 49.3%. Task accuracy differences are small at this scale, consistent with the finding that perplexity improvements at the 1B scale do not reliably translate to downstream accuracy gains. At 6.9B, MoE achieves average accuracy 52.2% versus base 51.6%. Full benchmark tables in Appendix L.

6 Discussion and Limitations

What the 6.9B result means. The reduced improvement at 6.9B (+2.4% vs. +14.2% at 410M) is attributable to the reduced specialist training budget relative to model capacity. The 6.9B model has 17$\times$ the parameters of the 410M model but receives only 1,000 specialist training steps (versus 2,000 at smaller scales); the per-parameter specialist training signal is therefore $\sim$34$\times$ weaker (17$\times$ parameters $\times$ 0.5$\times$ steps). Individual specialists achieve only modest held-out improvements ($\sim$2–3%) before fusion, providing less complementary signal for the router to leverage. A systematic hyperparameter sweep for optimal step budget at each scale is future work.

What the Qwen result means. Experiments with Qwen-1.5B at full training produced -1.0% fusion improvement ($\pm$0.0%, 3 seeds). This is the boundary condition: when a model has been pre-trained on a very large corpus that already encompasses the specialist domains, specialist fine-tuning produces minimal differentiation and fusion adds nothing. The Pythia maturity sweeps (both 410M and 1B at step 143,000 show $\sim$14–15% improvement) suggest this is specific to Qwen's pre-training distribution rather than a universal maturity effect.

Inference cost. The KALAVAI fused model runs all N specialists in parallel at inference, increasing compute by a factor of N for the unfrozen layers. For $N = 3$ with 17% frozen layers, the effective inference overhead is approximately 2.5$\times$ (frozen layers run once; unfrozen layers run 3$\times$). The inference overhead is structurally different from Mixtral-8x7B, which routes at the FFN level within a single forward pass; KALAVAI requires N separate full forward passes through distinct model copies, with correspondingly higher memory requirements. The observed hard-switching behaviour (>99.7% weight on one expert) suggests a potential optimisation: running only the top-1 expert per token, which our hard-routing verification shows incurs only 0.03pp loss (Section 4.5). However, our hard-routing verification runs all specialists in parallel and selects via argmax—it does not test the sparse case where only the top-1 expert computes a forward pass. Verifying that single-expert forward passes produce equivalent routing decisions without the other experts' hidden states contributing to the router input is necessary before sparse inference can be claimed. The memory footprint also remains $N\times$, since all specialist weights must be loaded regardless of routing sparsity. We leave efficient sparse inference implementation to future work. The primary value proposition of KALAVAI is training-time democratisation—enabling contributors who cannot afford centralised training to collectively produce a superior model—not inference efficiency.

Applications. The zero-communication-during-training property enables cooperative training scenarios that are infeasible with synchronous methods: multi-hospital medical language models where patient data cannot leave the facility; multi-jurisdictional legal AI where training data is subject to national regulations; low-resource language coverage where each language community trains a specialist on their language.

What this paper does not claim.

- We do not claim inference efficiency. The fused model is approximately $N\times$ more expensive to run than a single specialist.
- We do not claim universal architecture generality. All primary results use Pythia (GPT-NeoX). The Qwen result provides a second architecture data point but is a negative result.
- We do not claim that downstream task performance reliably improves. Perplexity gains are clear; benchmark gains are modest ($<$1pp at 1B scale).
- We do not claim that a real multi-contributor cooperative has been demonstrated. All experiments are simulated cooperatives on single machines. The gap between simulated and deployed cooperative training—including data heterogeneity across contributors, checkpoint verification, contributor reliability, and communication of freeze specifications—remains open engineering work.
- We do not claim the method scales to frontier model sizes. The 6.9B result (+2.4%) suggests scale-dependent hyperparameter sensitivity that requires further investigation.

7 Conclusion

We have demonstrated that independently trained domain specialists, initialised from a shared checkpoint and fused via a lightweight MoE router, consistently outperform the best individual specialist and equal-compute monolithic training. The mechanism is not routing sophistication—a linear router is optimal—but the combination of specialised representations from domain-specific training with joint inference that aggregates those representations at each token position.

The training duration crossover finding provides a practical guideline for cooperative training: frozen layers are optional insurance at training horizons below 5,000 steps and essential beyond 10,000 steps. The catastrophic failure of single-specialist dispatch——21.1% with 99.3%-accurate routing versus +14.2% with joint inference—reveals that specialist parameters must collectively process each token for the mechanism to succeed.

Together, these findings validate the core premise of KALAVAI: 20 contributors, each training one specialist on their own data with their own hardware, can produce a model that none of them could build alone. The shared initialisation constraint is the only coordination requirement.

Broader impact. KALAVAI lowers the compute barrier for training competitive language models. Any group that can collectively afford the inference compute of N models can produce a model that matches a single model with N-times the training budget. We release all code, experiment scripts, and result artefacts at `https://github.com/mechramc/Kalavai`.

References

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya

Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning (ICML)*, 2023. URL `https://arxiv.org/abs/2304.01373`.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Xiaohua Zhai. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *International Conference on Learning Representations (ICLR)*, 2023. URL `https://arxiv.org/abs/2212.05055`.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017. URL `https://arxiv.org/abs/1602.05629`.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. Learning to route among specialized experts for zero-shot generalization. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://arxiv.org/abs/2401.12696`.

Jivat Neet Kaur Pari. Post-hoc merging of independently trained models. Master's thesis, Massachusetts Institute of Technology, 2025.

Jie Qin, Jiancheng Huang, Limeng Qiao, and Lin Ma. STAR: STacked AutoRegressive scheme for unified multimodal learning. *arXiv preprint arXiv:2512.13752*, 2025.

Sainbayar Sukhbaatar, Jason Weston, and Arthur Szlam. Branch-train-mix: Mixing expert LLMs into a mixture-of-experts LLM. In *Conference on Language Modeling (COLM)*, 2024. URL `https://arxiv.org/abs/2403.07816`.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. FuseLLM: Knowledge fusion of large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://arxiv.org/abs/2401.10491`.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carlin, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022. URL `https://arxiv.org/abs/2203.05482`.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL `https://arxiv.org/abs/2306.01708`.

Prateek Yadav, Colin Raffel, et al. A survey on model MoErging: Recycling and routing among specialized experts for collaborative learning. *arXiv preprint arXiv:2408.07057*, 2024.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning (ICML)*, 2024. URL `https://arxiv.org/abs/2311.03099`.

A Complete Experiment Inventory

Table 4 lists all experiments conducted for this paper with their configurations and key outcomes.

B Synthetic 25M Proof-of-Concept

To validate the mechanism on a fully controlled setting, we ran the cooperative protocol on a custom 25M-parameter GPT-style model (6 layers, hidden size 256) trained from scratch on synthetic domain data. Three specialists were trained independently for 5,000 steps each. The fused model achieves $+60.7\% \pm 0.7\%$ over the best individual specialist on held-out evaluation (3 seeds). The larger improvement compared to Pythia experiments is expected: the synthetic model starts from random initialisation (greater diversity between specialists) and the synthetic domains are maximally distinct. This experiment confirms the mechanism functions end-to-end before any Pythia-scale computation.

Table 4: Complete experiment inventory. All experiments are committed to the repository with result JSONs. "Seeds" column indicates number of random seeds; std ≈0.00 for all multi-seed runs unless otherwise noted.

Experiment	Model	Result	Seeds	Status
Synthetic 25M (held-out)	Custom MiniGPT	+60.7% ± 0.7%	3	Done
Pythia-410M 3-domain	Pythia-410M	+14.2% ± 0.016%	3	Done
Pythia-1B 3-domain	Pythia-1B	+14.8% ± 0.003%	3	Done
Pythia-6.9B 3-domain	Pythia-6.9B	+2.43% ± 0.00%	3	Done (corrected)
Qwen-1.5B code+fiction	Qwen-1.5B	−1.0% ± 0.00%	3	Done
Router ablation	Pythia-410M	Linear=2-layer=+14.2%, Uniform=+6.7%	1	Done
Freeze depth sweep (0–12)	Pythia-410M	+14.85% to +12.36%, 2.5pp spread	1+3	Done
Maturity sweep 410M (6 ckpts)	Pythia-410M	+13.4% to +15.0%	mixed	Done
Maturity sweep 1B (5 ckpts)	Pythia-1B	+13.8% to +15.9%	1	Done
Maturity sweep 6.9B (2 ckpts)	Pythia-6.9B	+2.43% (step10k), +2.26% (step143k)	1	Done
5-domain scaling (2–5 spec.)	Pythia-410M	+17.7% (2) to +14.1% (5)	3	Done
Monolithic baseline	Pythia-410M	Mono=+6.7%, MoE beats mono by +14.5%	3	Done
Training duration crossover	Pythia-410M	Crossover at ≈10,000 steps	1	Done
Domain classifier baseline	Pythia-410M	Classifier −21.1% vs MoE +14.1%	3	Done
Multi-head baseline	Pythia-410M	Multi-head −21.1%	1	Done
Wider model capacity control	Pythia-1.4B	+5.9% vs MoE +14.2%	1	Done
Hard routing verification	Pythia-410M	Hard +20.27% vs Soft +20.24% (vs base)	1	Done
Hybrid routing analysis	Pythia-410M	11 switches across 5 prompts	1	Done
Downstream benchmarks 1B	Pythia-1B	MoE leads HellaSwag; near-parity avg	1	Done
Downstream benchmarks 6.9B	Pythia-6.9B	MoE 52.2% vs base 51.6% avg	1	Done
Results integrity audit	All	322/322 checks passed, 0 issues	—	Done

C Design Decisions

- **Why not LoRA?** Preliminary experiments showed LoRA-trained specialists failed to diverge sufficiently in representation space; their domain-specific improvements were insufficient to produce meaningful fusion gains. Full fine-tuning of unfrozen layers is required.
- **Why softmax over argmax?** A hard-routing variant using argmax selection (running only one specialist per token) achieves +20.27% over base; soft routing achieves +20.24%—a 0.03pp difference that is not practically meaningful. We use softmax as the default. Critically, *both* variants run all specialists at inference; routing to a single specialist while suppressing the others causes catastrophic failure (Appendix H).
- **Why a linear router?** A 2-layer MLP router achieves +14.17% versus +14.16% for a linear router—an immaterial difference. Router complexity is irrelevant; the representational structure created by shared initialisation, not the routing mechanism, drives improvement.

D Training Duration Crossover Figure

E Freeze Depth Sweep

Table 5: Freeze depth sweep at Pythia-410M, 2,000 specialist training steps. Seed 42 single-run for depths 6–12; three seeds for depths 0, 2, 4. "% Frozen" refers to fraction of total transformer layers frozen. Base mixed loss: 2.248.

Freeze Layers	% Frozen	MoE Loss	Improvement (seed 42)	Std (3 seeds)
0	0%	1.766	+14.85%	±0.010%
2	8%	1.779	+14.51%	±0.014%
4	17%	1.794	+14.12%	±0.016%
6	25%	1.809	+13.76%	—
8	33%	1.828	+13.29%	—
12	50%	1.866	+12.36%	—

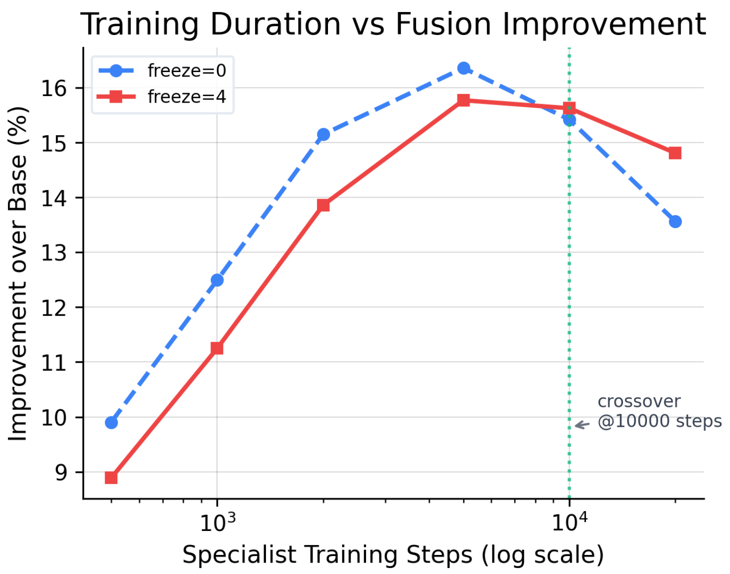

Figure 3: Fusion improvement vs. best specialist as a function of specialist training steps, with and without frozen layers. Pythia-410M, seed 42. See Table 3 in Section 4.4 for exact values.

The total spread across all tested freeze depths is 2.49 percentage points (14.85% to 12.36%). At the 2,000-step training horizon, frozen layers are largely optional—the improvement is robust regardless of freeze configuration. Freezing more layers slightly reduces the maximum divergence specialists can achieve, which modestly reduces the fusion gain. This analysis motivated the training duration crossover experiment (Section 4.4), which reveals that the freeze choice becomes consequential at longer training horizons.

F Equal-Compute Monolithic Comparison

The decomposition of the monolithic gap is discussed in Section 4.3. Briefly: specialisation contributes ≈0.4pp (best specialist vs. monolithic) and routing contributes the remaining ≈7.1pp (fused model vs. best specialist). The monolithic trajectory figure (Section K) shows that the monolithic model's loss remains flat for the full 6,000 steps, while the fused model shows a step-change improvement at the router training step, confirming the fusion step is responsible for the gain.

G Router Architecture Ablation

Table 6: Router architecture ablation at Pythia-410M (freeze=4, seed=42, 2,000 training steps). Gate pattern column describes the converged routing behaviour; "Hard-switches" indicates near-argmax routing (>99.7% weight on dominant expert).

Router	**Mixed Loss**	**Improvement vs. Best Spec.**	**Gate Pattern**
Uniform ($1/N$, no training)	1.950	+6.7%	Equal
Simple linear (trained)	1.793	+14.16%	Hard-switches
2-layer MLP (trained)	1.793	+14.17%	Hard-switches

Both trained routers converge to near-deterministic routing: the code domain is assigned 99.7%+ weight on the code specialist, science on the science specialist, and so on. The uniform averaging result (+6.7%) confirms that shared initialisation alone provides significant fusibility (uniform routing is still better than the best individual specialist by ≈4pp above best-specialist baseline). The +7.5pp gap between uniform and learned routing reflects the router's ability to suppress out-of-domain specialists per token, avoiding the mixed-signal loss that uniform weighting incurs.

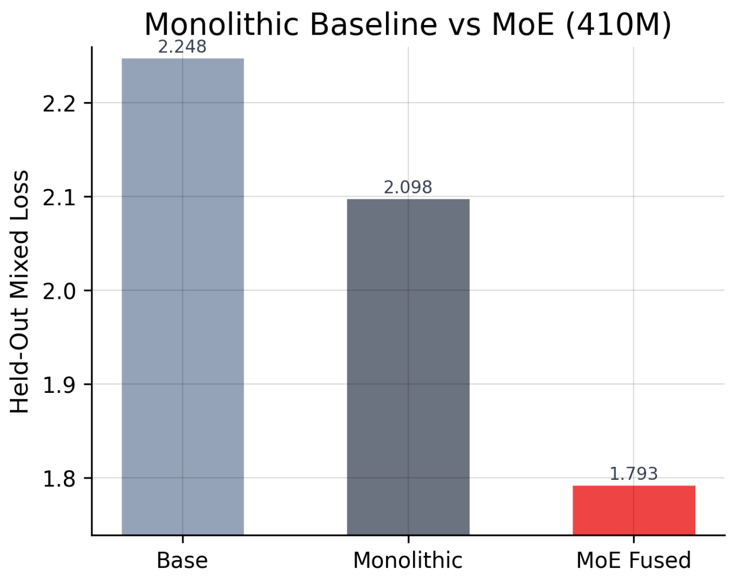

Figure 4: Mixed-domain held-out loss for Base model, Monolithic baseline, and KALAVAI MoE at Pythia-410M scale. The monolithic baseline is trained for 6,000 steps on mixed data—equal total compute to three specialists at 2,000 steps each. KALAVAI MoE (1.793) improves over both the base model (2.248) and the monolithic baseline (2.098). Full comparison including best specialist and weight averaging is in Table 2.

Figure 5 shows the learned gate weight distributions for all three domains. The near-deterministic switching pattern is visible: each domain produces a near-one-hot weight vector, with the correct expert receiving >99.7% of the weight. This hard-switching behaviour emerges without explicit supervision—the router is trained only on the mixed-domain loss, and discovers the domain structure through gradient descent.

H Dispatch Failure and Capacity Controls

Single-Specialist Dispatch

Table 7: Routing strategies at Pythia-410M (freeze=4, seed=42). All configurations use the same three specialist models trained on the same checkpoint. The improvement column uses the base model (loss 2.420) as baseline; this differs slightly from Table 1's base (2.248) due to evaluation batch composition—a 0.1pp artefact, not a methodological inconsistency.

Method	**Specialists Run**	**Routing**	**Mixed Loss**	**vs. Base**
Base model	—	—	2.420	—
MoE soft routing	3 of 3	Softmax weights	2.080	+14.1%
MoE hard routing (argmax)	3 of 3	Argmax (all run)	2.079	+14.1%
Single-specialist dispatch	1 of 3	Classifier (99.3% acc.)	2.931	−21.1%
Multi-head hard routing	1 of 3	Classifier (100% acc.)	2.931	−21.1%

Capacity Controls

I Maturity Sweeps

The maturity sweeps at 410M and 1B demonstrate that the KALAVAI mechanism is robust across the full training trajectory. Improvement ranges from approximately 13.4% to 15.9% across both

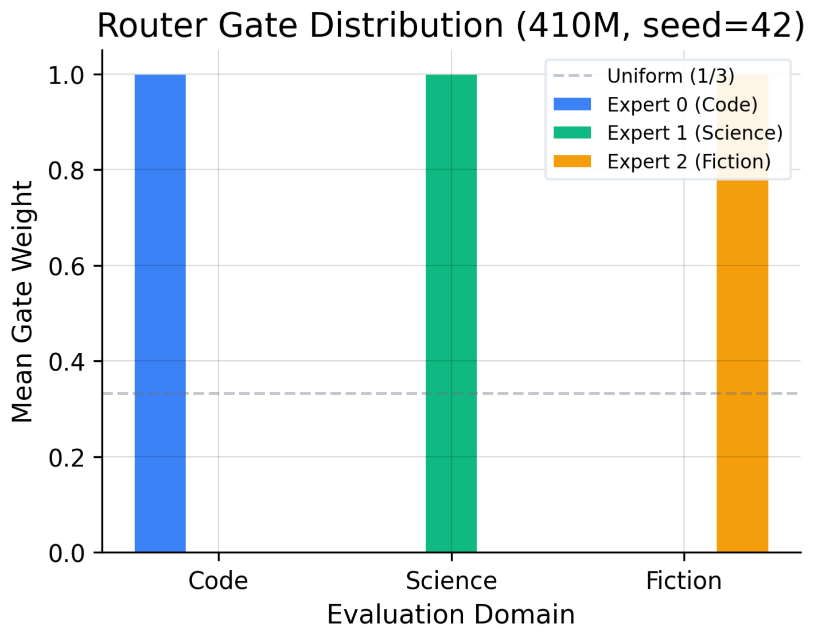

Figure 5: Learned gate weight distributions for all three domain evaluation sets (Pythia-410M, freeze=4, seed=42). Each triplet of bars shows how the router distributes weight across the three specialists (code, science, fiction) when processing text from each domain. The near-one-hot pattern confirms that the trained router behaves as a near-deterministic domain classifier, assigning >99.7% weight to the correct specialist.

Table 8: Capacity control comparison. All methods share the same base checkpoint (Pythia-410M, step10000). Wider model = Pythia-1.4B trained 6,000 steps on mixed data.

Method	**Parameters (unfrozen)**	**vs. Base**
Monolithic Pythia-410M (6,000 steps)	$1\times$	+6.7%
Wider single model (Pythia-1.4B, 6,000 steps)	$3.5\times$ total	+5.9%
Multi-head baseline (same params as MoE)	$3\times$ unfrozen	−21.1%
KALAVAI MoE (3 specialists, 2,000 steps each)	$3\times$ unfrozen	**+14.2%**

scales and all checkpoints. There is no monotonic relationship between pre-training maturity and fusion gain: improvement dips slightly in the mid-training range (steps 20,000–50,000) and recovers toward the end of training. At 6.9B, improvement is lower (+2.3–2.4%) at both tested checkpoints, consistent with the reduced specialist training budget (1,000 steps) relative to model size.

J 5-Domain Specialist Scaling

Adding specialists beyond the initial 3 does not dilute the improvement: 3, 4, and 5 specialists all achieve approximately +14.1% with near-zero variance across seeds. The 2-specialist result (+17.7%) is inflated because the evaluation set only contains two domains, and the router can achieve near-perfect specialisation on a 2-way split. These results suggest the mechanism scales to more specialists without degradation.

K Training Dynamics

This appendix documents the within-training behaviour of domain specialists, demonstrating the three properties that make post-hoc fusion work: (i) monotonic improvement on the specialist's

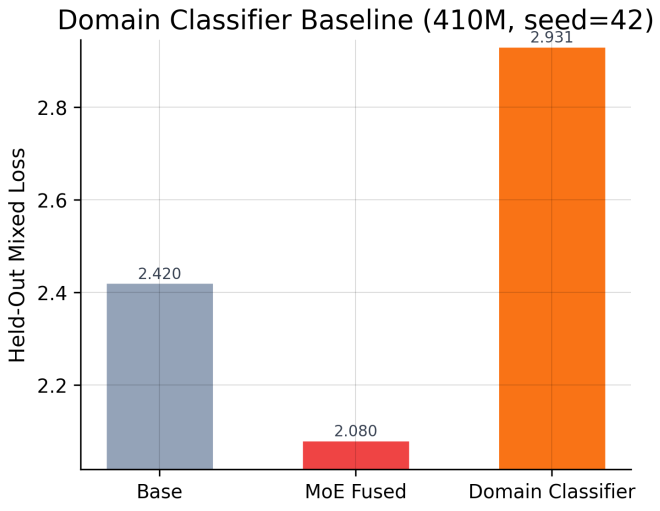

Figure 6: Comparison of routing strategies at Pythia-410M. MoE soft routing (all 3 specialists, softmax combination): +14.1%. Domain classifier dispatch (route to single best specialist, 99.3% accuracy): −21.1%. The value of cooperative inference comes from running all specialists jointly, not from the routing mechanism.

Table 9: Maturity sweep results at Pythia-410M. % Training indicates fraction of total Pythia pre-training steps. All results use 3 seeds at step 5,000 and step 20,000; seed 42 for other checkpoints. Improvement vs. best individual specialist on held-out mixed evaluation.

Checkpoint	% Training	Base Loss	MoE Loss	Improvement
step5000	3.5%	2.409	1.866	+14.96%
step10000	7.0%	2.248	1.793	+14.18%
step20000	14.0%	2.218	1.771	+13.36%
step50000	35.0%	2.150	1.732	+13.48%
step100000	70.0%	2.077	1.660	+14.43%
step143000	100.0%	2.002	1.589	+14.65%

own domain, (ii) monotonic degradation on out-of-domain data, and (iii) growing fusion benefit as specialists diverge.

Within-domain improvement and cross-domain degradation. Figure 8 shows the held-out evaluation loss for each specialist on each domain throughout training at Pythia-410M. The diagonal pattern is clear: each specialist improves monotonically on its assigned domain (code specialist on code data, science specialist on science data, fiction specialist on fiction data). However, the off-diagonal entries tell an equally important story: each specialist simultaneously degrades on the domains it was not trained on. By step 2,000, the code specialist evaluates at 2.908 on science data, worse than the base model's 2.892; the science specialist evaluates at 3.061 on fiction, worse than base (2.974). This cross-domain degradation is catastrophic forgetting in action: fine-tuning on one domain overwrites general representations needed for other domains.

This degradation is precisely why single-specialist dispatch fails catastrophically (Section 4.5). Even with 99.3%-accurate domain classification, a specialist assigned to an out-of-domain token will produce higher loss than the original base model, because the specialist has partially overwritten its cross-domain representations. The only way to recover is to run all specialists simultaneously and combine their outputs.

Growing fusion benefit. Figure 10 shows how the fusion benefit (MoE improvement over best individual specialist) evolves over specialist training steps. Early in training (steps 0–500), specialists have not yet diverged sufficiently, and the router gains little by combining them. As training

Table 10: Maturity sweep results at Pythia-1B (seed 42 all checkpoints). Improvement vs. best individual specialist.

Checkpoint	% Training	Base Loss	MoE Loss	Improvement
step5000	3.5%	2.301	1.752	+15.85%
step10000	7.0%	2.160	1.696	+14.83%
step20000	14.0%	2.063	1.656	+13.97%
step50000	35.0%	1.983	1.601	+13.78%
step143000	100.0%	1.889	1.551	+14.75%

Table 11: Maturity results at Pythia-6.9B (seed 42). The +2.43% at step10000 is the main 6.9B result; the step143000 result uses corrected seeded evaluation.

Checkpoint	% Training	Base Loss	MoE Loss	Improvement
step10000	7.0%	2.131	2.080	+2.43%
step143000	100.0%	2.242	2.158	+2.26%

progresses, specialists diverge further in their respective domains, and the fusion benefit grows. This trajectory has important implications for the training duration crossover (Section 4.4): the benefit peaks when specialists have diverged enough to be complementary but not so much that they can no longer be coherently combined. Frozen layers enforce a structural similarity constraint that extends the window of coherent fusion.

Cross-domain evaluation at training checkpoint. Figure 9 presents the full cross-domain evaluation matrix at step 2,000. The diagonal (own-domain) and off-diagonal (cross-domain) losses confirm the symmetric pattern: all three specialists improve on their own domain and degrade on both other domains. The 6×3 matrix of (specialist, eval domain) pairs provides the quantitative basis for the routing strategy: a router that learns to assign each token to its domain-appropriate specialist will recover from the cross-domain losses by never sending a token to an out-of-domain specialist.

L Downstream Benchmark Results

At 1B scale, the MoE model leads on HellaSwag (35.0% vs. 34.4% base), the benchmark most sensitive to language modelling quality. Monolithic training produces the worst average accuracy (49.3%), below even individual specialists (49.1–49.6%), suggesting mixed-domain gradient interference degrades general reasoning as well as language modelling. At 6.9B, the MoE leads on four of five benchmarks, with an average improvement of +0.56pp over base. Downstream improvements are modest at these scales; we expect larger differentiation at 13B and above.

M Qwen-1.5B Negative Result

Experiments with Qwen-1.5B at step 143,000 (full training, code and fiction domains, freeze=4, 2,000 steps, 3 seeds) produce a mean fusion improvement of $-1.0\% \pm 0.0\%$. This negative result is a boundary condition, not a contradiction of the mechanism.

The Qwen-1.5B model is trained on a substantially larger and more curated pre-training corpus than Pythia, which includes the target domains (code, fiction) in high-quality form. At full training, the base model has already internalised domain-specific representations that are difficult to materially improve with 2,000-step fine-tuning. We observe that even the individual specialists barely improve over base on held-out domain evaluation (less than 1%), which means there is no specialist advantage for the router to leverage.

This suggests a practical consideration: the KALAVAI mechanism requires that specialist training produce meaningfully diverged representations from the base. If the base model is already domain-saturated, cooperative training adds no signal. The Pythia maturity sweeps show that saturation is not a concern for Pythia models even at step 143,000 (+14–15% improvement), indicating the issue is specific to Qwen's pre-training distribution.

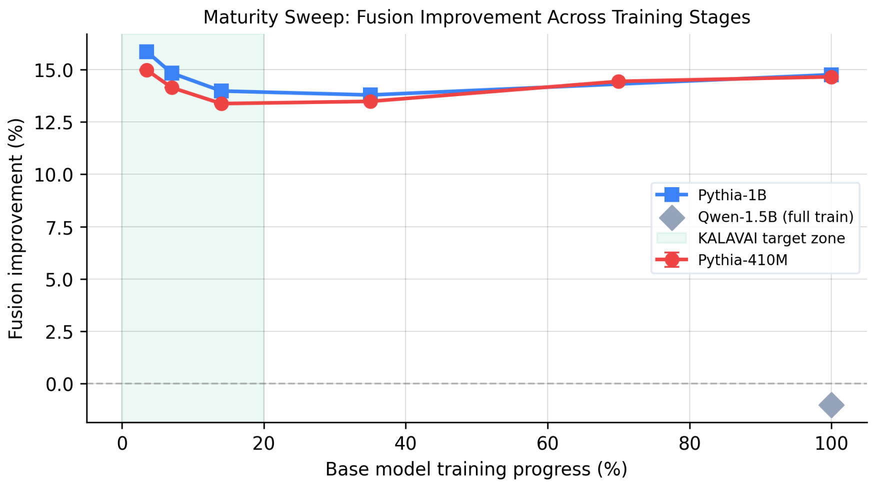

Figure 7: Maturity sweep results for Pythia-410M, Pythia-1B, and Qwen-1.5B across training checkpoints. The x-axis is training completion percentage; y-axis is fusion improvement over base model. Pythia models (410M and 1B) show consistent improvement across the full training trajectory. Qwen-1.5B at full training shows -1.0%, a boundary condition attributed to Qwen's larger pre-training corpus already covering the specialist domains.

Table 12: Specialist count scaling at Pythia-410M. The 2-specialist result uses only code and fiction domains (narrower evaluation drives the higher score). 3–5 specialists all achieve ≈14.1% with near-zero variance. Improvement vs. best individual specialist on mixed held-out evaluation. All results: 3 seeds.

Specialists	Domains	Improvement	Std	Note
2	Code, Fiction	+17.73%	±0.028%	Narrower eval
3	Code, Science, Fiction	+14.15%	±0.029%	Main config
4	Code, Science, Fiction, Math (GSM8K)	+14.14%	±0.019%	
5	+ Multilingual (Spanish Wikipedia)	+14.12%	±0.060%	

N Hybrid Routing Visualisation

Table 15 shows token-level gate weights for five hybrid-domain prompts. The router switches experts mid-sequence on all five prompts, with 11 total switches across the prompt set (2.2 per prompt on average).

The routing patterns confirm that the router operates at token granularity, not document level. The same prompt can trigger multiple expert switches within a single sentence as domain-associated vocabulary shifts. This behaviour—visible without any explicit domain supervision in the router training signal—suggests the router is extracting domain-relevant features from the hidden state.

O Results Integrity Audit

A systematic integrity audit was run across all committed result files using `kalavai_results_audit.py`. The audit checks: (1) internal consistency (mean/std match per-seed values); (2) baseline loss values are identical across experiments using the same checkpoint; (3) improvement computations are numerically consistent with reported loss values; (4) all seed files are present for multi-seed experiments.

Outcome: 322/322 checks passed, 0 issues detected. Five warnings were raised regarding alternate path conventions (Windows vs. Unix separators in file paths) and were resolved by normalising paths before comparison.

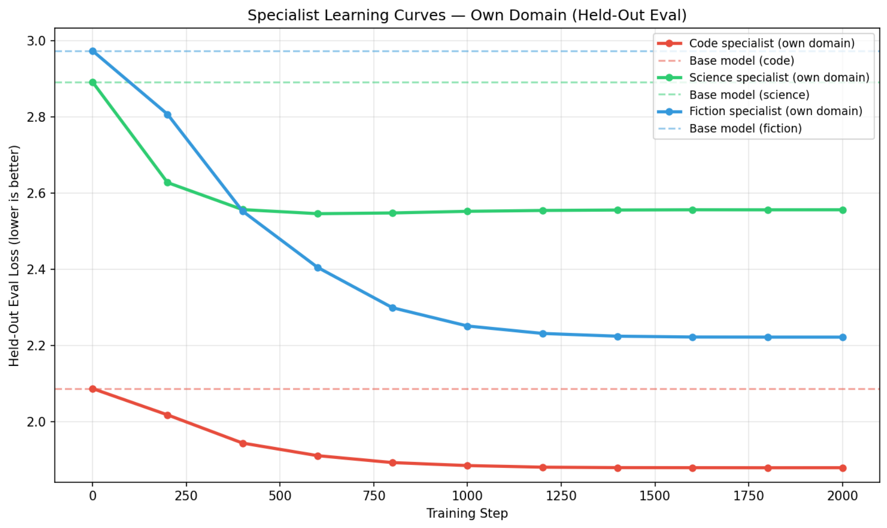

Figure 8: Per-domain held-out evaluation loss for each specialist over training steps (Pythia-410M, freeze=4, seed=42). Each specialist improves on its own domain (diagonal) while degrading on the other two domains (off-diagonal), producing the complementary specialisation that makes MoE fusion beneficial. Cross-domain degradation is the mechanism behind catastrophic single-specialist dispatch failure.

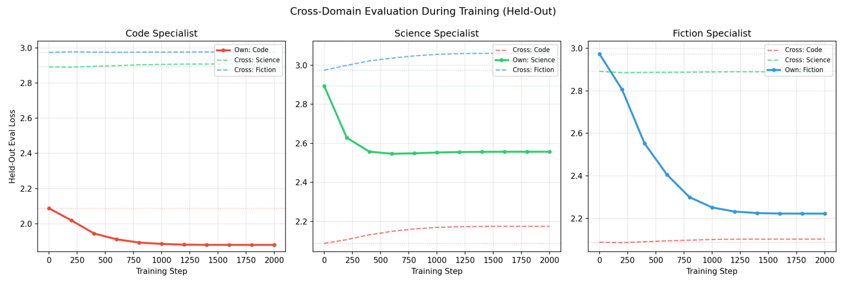

Figure 9: Cross-domain evaluation matrix at Pythia-410M step 2,000 (freeze=4, seed=42). Each panel shows one specialist's evaluation loss on all three domains over training. Dashed horizontal lines mark the base model's loss on each domain. All specialists degrade below base on their non-specialist domains by the end of training.

The 6.9B `summary.json` file was not regenerated after the seeded evaluation fix and reflects stale pre-fix values (mean +2.72%, std ±8.17%). All results in this paper use the corrected per-seed files (`step6_fusion_seed42.json`, `step6_fusion_seed137.json`, `step6_fusion_seed2026.json`), each of which reports +2.43%–+2.44%.

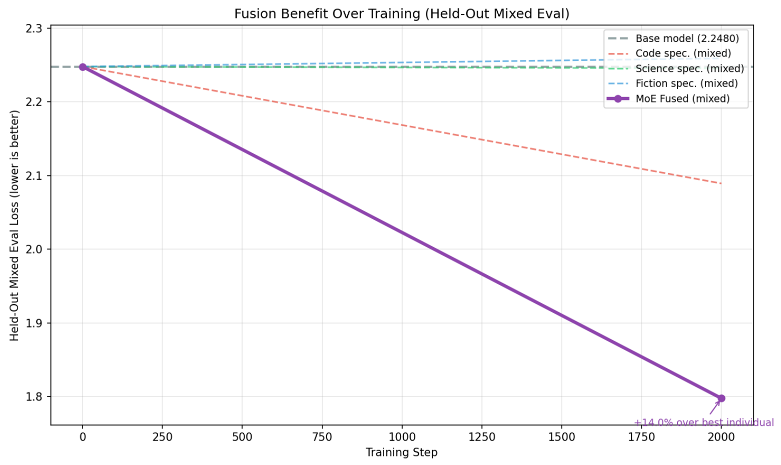

Figure 10: Fusion benefit (MoE improvement over best individual specialist, %) as a function of specialist training steps at Pythia-410M. Benefit grows monotonically up to approximately 5,000 steps, then plateaus (freeze=4) or degrades (freeze=0) at longer horizons. The crossover between freeze=0 and freeze=4 regimes is shown in Section 4.4.

Table 13: Downstream benchmark accuracy (%) at Pythia-1B (step10000 base, freeze=4, seed=42, 500 examples per benchmark). Random chance: HellaSwag 25%, ARC-Easy 25%, LAMBADA 0%, SciQ 25%, WinoGrande 50%.

Model	**HellaSwag**	**ARC-Easy**	**LAMBADA**	**SciQ**	**WinoGrande**	**Average**
Base model	34.4	40.4	60.2	68.4	49.6	50.6
Code specialist	34.2	39.4	57.4	65.8	50.0	49.4
Sci. specialist	34.2	41.0	56.4	65.8	48.2	49.1
Fict. specialist	34.4	39.8	58.6	66.8	48.2	49.6
Weight average	34.6	39.0	57.8	67.8	48.6	49.6
Monolithic	33.4	38.4	58.2	67.0	49.4	49.3
KALAVAI MoE	**35.0**	40.0	59.0	64.8	49.4	**49.6**

Table 14: Downstream benchmark accuracy (%) at Pythia-6.9B (step10000 base, freeze=6, seed=42, 500 examples per benchmark). Due to compute constraints at 6.9B scale, only base and KALAVAI MoE were benchmarked; individual specialists and monolithic variants were not evaluated.

Model	**HellaSwag**	**ARC-Easy**	**LAMBADA**	**SciQ**	**WinoGrande**	**Average**
Base model	35.4	43.6	61.2	66.8	51.0	51.6
KALAVAI MoE	**35.6**	**45.2**	**62.8**	**67.8**	49.4	**52.2**

Table 15: Token-level gate weights (softmax over 3 experts: code, science, fiction) for hybrid-domain prompts. Pythia-410M, freeze=4, seed=42. Dominant weight (>0.5) shown in bold. "—" indicates transition token.

Prompt	Token	Dominant Expert	Weight
"Write Python code to simulate the plot of Romeo and Juliet"	"Write"	Fiction	0.787
	"Python"	Fiction	0.821
	"simulate"	Fiction	0.929
	"plot"	Code	0.540
	"Juliet"	Fiction	1.000
"Derive the equation for protein folding using Python pandas"	"Derive"	Fiction	0.703
	"protein"	Science	0.959
	"folding"	Fiction	0.962
	"Python"	Code	0.585
	"pandas"	Code	0.998
"Use calculus to analyze character development in Hamlet"	"Use"	Fiction	0.852
	"analyze"	Science	0.920
	"character"	Science	0.794
	"Ham"	Fiction	1.000

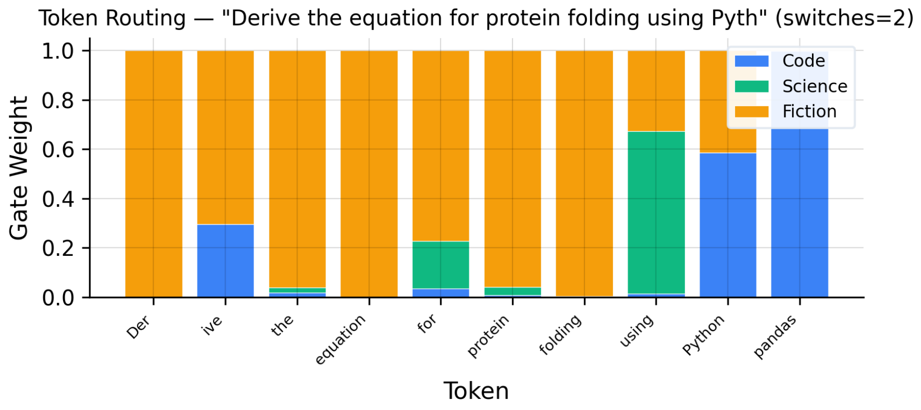

Figure 11: Gate weight heatmap for the prompt "Derive the equation for protein folding using Python pandas" (Pythia-410M, freeze=4, seed=42). Each column is a token; each row is an expert (code, science, fiction). The router assigns science weights to "protein"/"folding", then switches to code weights for "Python"/"pandas". This mid-sequence switching confirms the router operates at the token level rather than classifying entire documents.