COFFIE: CONSENSUS-FIRST FINE-TUNING FOR MOE

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) models are increasingly used as base models for downstream tasks, but how best to fine-tune them remains an open question. Full fine-tuning stores optimiser state for every expert, even though only a few are active for each token and each expert sees only a small fraction of the fine-tuning data. Parameter-efficient methods reduce this cost, but most inherit designs from dense models: each expert or adapter computes its update independently, whereas the output of an MoE layer depends on how its active experts work together. We propose **CoFFiE** (**Co**nsensus-**F**irst **Fi**ne-tuning for Mixture-of-**E**xperts), which keeps the experts and the router frozen and adapts this coordination instead. A small shared network refines each expert's output using its own state and the route-weighted consensus of the active experts. Because refining the experts also changes this consensus, we define the refined outputs implicitly, as their joint fixed point. A few iterations suffice in practice, and training memory does not grow with the iteration count. We further extend the method by propagating consensus information across layers, which also improves accuracy significantly. On OLMoE-1B-7B, Mixtral-8×7B and Qwen3-30B-A3B, CoFFiE outperforms the strongest MoE-specific PEFT baselines with the same or fewer trainable parameters, by up to 3.3 points on commonsense and arithmetic reasoning benchmarks. Code: `https://anonymous.4open.science/r/caft-ano-DD26/`.

1 INTRODUCTION

Sparse Mixture-of-Experts (MoE) layers scale model capacity by activating only a small subset of experts for each token (Jacobs et al., 1991; Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022), and have become a common component of open large language models (Jiang et al., 2024; Dai et al., 2024; Muennighoff et al., 2025). Yet fine-tuning sparse MoEs remains less developed than fine-tuning dense models. Standard full-parameter fine-tuning largely loses the efficiency advantage of sparse activation: parameter updates and optimiser states scale with the *total* model size rather than the number of parameters activated per token. For example, OLMoE-1B-7B contains 7B parameters but activates only about 1B for each token (Muennighoff et al., 2025).

Parameter-efficient fine-tuning (PEFT) reduces this cost, but most existing methods inherit designs developed for dense models. Attention-only LoRA leaves the MoE feed-forward blocks, where much of the model capacity resides, unchanged. Adapting the expert blocks directly often requires separate modules for individual experts, even though each expert sees only a fraction of the fine-tuning data. Routed methods such as PERFT (Liu et al., 2024) instead introduce a small set of adapter experts; its PERFT-R variant learns an additional router to select among them. Despite these different designs, the adaptation of an expert typically does not depend on the outputs of the other experts selected for the same token. Experts, or the adapters attached to them, are adapted individually, but their coordination is not explicitly adapted.

Yet what a token receives from an MoE layer is not the output of any single expert, but the combination of several experts that are each computed without seeing the others. We therefore take a different view: *expert coordination itself can be the target of fine-tuning*. For a token routed to several experts, we refine each expert using information from the others through their route-weighted consensus. This creates a circular dependency: the refined experts depend on the consensus, while the consensus itself depends on the refined experts. A single refinement step therefore conditions on a consensus that changes immediately after the experts are refined.

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

We resolve this circularity by seeking a self-consistent solution. Each active expert is anchored at its frozen pretrained output and receives a small correction computed from the consensus of the active experts. The corrected expert states then define a new consensus, and the desired state is the fixed point at which these quantities are jointly consistent. This leads naturally to a deep-equilibrium formulation (Bai et al., 2019): the refined state is defined as a fixed point, which we approximate with a few solver iterations. Training uses an implicit-gradient approximation whose memory cost does not grow with the number of forward solver iterations (Bai et al., 2019; Geng et al., 2021; Fung et al., 2022).

Based on this formulation, we introduce **CoFFiE** (**Co**nsensus-**F**irst **Fi**ne-tuning for Mixture-of-**E**xperts). CoFFiE keeps the backbone, router, and all pretrained experts frozen. Within each MoE layer, a small bottleneck network shared across the active experts learns to refine their outputs using their route-weighted consensus; because its output layer is zero-initialised, CoFFiE starts exactly from the pretrained MoE function. Since a token passes through a sequence of MoE layers, we further let the consensus reached at one layer inform the next, through a cross-layer variant that propagates consensus information between consecutive MoE layers.

Our contributions are as follows.

- **We introduce expert coordination as a target for MoE adaptation.** Rather than adapting each selected expert independently, CoFFiE lets its adaptation explicitly depend on the outputs of the other active experts, while keeping the pretrained experts and router frozen.
- **We formulate cross-expert adaptation as a coupled fixed-point problem.** Each active expert is anchored at its pretrained output and refined using the route-weighted consensus of the active experts. The resulting coupled state is solved as a fixed point, with training memory independent of the number of forward solver iterations. We further extend the formulation to propagate consensus across layers.
- **We show consistent gains under matched trainable budgets.** Across three MoE backbones on commonsense reasoning and two on arithmetic reasoning, CoFFiE outperforms the strongest routed-PEFT baselines we evaluate using the same or fewer trainable parameters. Parameter-matched ablations further isolate the gains from cross-expert consensus and cross-layer coordination.

2 Related Work

Mixture-of-Experts and expert interaction. Existing work introduces expert interaction in several ways. Chain-of-Experts (Wang et al., 2025) and SDG-MoE (Kulibaba et al., 2026) build interaction into expert layers trained from scratch, and DAG-MoE (Feng et al., 2026) learns an acyclic aggregation structure over the selected experts, likewise trained into the architecture from pretraining and fine-tuned with all parameters. HyperMoE (Zhao et al., 2024) draws on unselected experts while fine-tuning the full model; SymphonySMoE (Nguyen-Nhat et al., 2025) modifies routing rather than expert outputs; and GraphMoE (Lv et al., 2025) and GMoE (Bai et al., 2024) connect newly introduced LoRA experts on dense backbones. In contrast, CoFFiE adapts the coordination of an existing pretrained MoE as parameter-efficient fine-tuning: it is inserted after pretraining, starts exactly from the pretrained function, and keeps the pretrained experts and router frozen.

Parameter-efficient fine-tuning of MoE models. LoRA (Hu et al., 2022) and related adapters were originally developed for dense backbones, and several methods introduce mixture structure into the adapters themselves by placing multiple LoRA modules under a learned gate (Zadouri et al., 2024; Li et al., 2024; Dou et al., 2024). For pretrained MoE backbones, ESFT (Wang et al., 2024) fine-tunes only the experts most activated by the target task, while PERFT (Liu et al., 2024) systematises the design of PEFT modules within the MoE block, with PERFT-R as its strongest routed variant. Across these approaches, adaptation focuses on which modules are selected or where they are inserted, while each module computes its correction independently. CoFFiE instead adapts how the selected experts coordinate through a shared refinement operator.

Implicit models and deep equilibrium networks. Deep equilibrium models (DEQ) (Bai et al., 2019) define a layer's output as the fixed point of a weight-tied map rather than the result of a fixed

computation graph, and backpropagate through the equilibrium using the implicit function theorem. This implicit view is shared with neural ODEs (Chen et al., 2018), while multiscale DEQs extend the framework to large vision models (Bai et al., 2020). Practical training often replaces the exact implicit gradient with inexact or one-step approximations (Geng et al., 2021; Fung et al., 2022), and Anderson acceleration (Anderson, 1965) is a common forward solver. In standard DEQs, the fixed point represents a token state propagated through a weight-tied backbone. In CoFFiE, it instead represents the joint state of the experts selected within a pretrained MoE layer, coupled through the routing weights and anchored at their frozen outputs.

3 METHOD

3.1 PRELIMINARIES: THE STANDARD MOE LAYER

We consider a Mixture-of-Experts (MoE) layer with N expert sub-networks $\{E_i\}_{i=1}^{N}$, each $E_i : \mathbb{R}^d \to \mathbb{R}^d$, and a router G_ϕ. Given a token representation $x \in \mathbb{R}^d$, the router selects a set S of k experts with weights $\{w_i\}_{i\in S}$, and the layer returns their weighted combination,

$$(w, S) \;=\; G_\phi(x), \qquad y \;=\; x \;+\; \sum_{i\in S} w_i\, E_i(x). \tag{1}$$

The exact form of G_ϕ (softmax then top-k, top-k then softmax, with or without renormalisation over the selected set) differs across MoE families;[1] we inherit it unchanged from the pretrained model and keep both G_ϕ and $\{E_i\}$ frozen.

3.2 CROSS-EXPERT REFINEMENT AS A COUPLED FIXED POINT

The property of (1) that concerns us is not the router but the sum. The selected experts are evaluated in parallel and then combined, so each $E_i(x)$ is produced without any information about what the other active experts returned; whatever complementary structure the selected set carries is left entirely to the fixed weights w to exploit. We make this absent interaction, rather than the routing decision or the expert parameters, the object of adaptation: each expert's representation is refined in the context of the others, and the others are the very representations being refined. These quantities depend on one another, so we define the refined state *implicitly*, as the solution of a coupled system (Figure 1).

Relabelling the selected experts as $1, \dots, k$, let $E = (E_1, \dots, E_k) \in \mathbb{R}^{kd}$ denote their stacked (refined) representations, with $E_i(x)$ the frozen output of the i-th selected expert, and let $\bar{w} \in \Delta^{k-1}$ denote the routing weights renormalised to sum to one over the selected set, computed by the frozen router and not trained. The representations are defined implicitly, as the state that is consistent with the consensus it induces:

$$E_i^\star \;=\; E_i(x) \;+\; \mathcal{T}_\theta\Big(E_i^\star,\; c(E^\star)\Big), \qquad c(E) \;=\; \sum_j \bar{w}_j\, E_j, \tag{2}$$

the same for every i. Here $E_i(x)$ is the *frozen* expert output, which serves as a fixed anchor (an input injection); $\mathcal{T}_\theta$ is the *refinement operator*, a small trainable network shared by the active experts of the layer and specified in Section 3.3; and $c(E)$ is the route-weighted consensus of the active experts, through which expert i perceives the others. Stacking (2) over the active experts leaves a *single* fixed-point equation in the one unknown E,

$$E^\star \;=\; \mathcal{F}_\theta\big(E^\star\big), \qquad \mathcal{F}_\theta(E)_i \;=\; E_i(x) + \mathcal{T}_\theta\Big(E_i,\; \textstyle\sum_j \bar{w}_j\, E_j\Big), \tag{3}$$

or equivalently the root-finding form

$$\mathcal{R}_\theta(E^\star) \;=\; E^\star - \mathcal{F}_\theta(E^\star) \;=\; 0. \tag{4}$$

The dependence runs in a circle (each expert is refined against the consensus, and the consensus is formed from the refined experts), but the circle closes through the single variable E. Solving such a self-referential equation, and differentiating through it, is what the deep-equilibrium (DEQ) framework provides (Bai et al., 2019).

[1] OLMoE applies softmax over all experts and keeps the top-k weights unnormalised; Mixtral takes the top-k logits and applies softmax over them; Qwen3-MoE (Yang et al., 2025) applies softmax, takes the top-k, and renormalises.

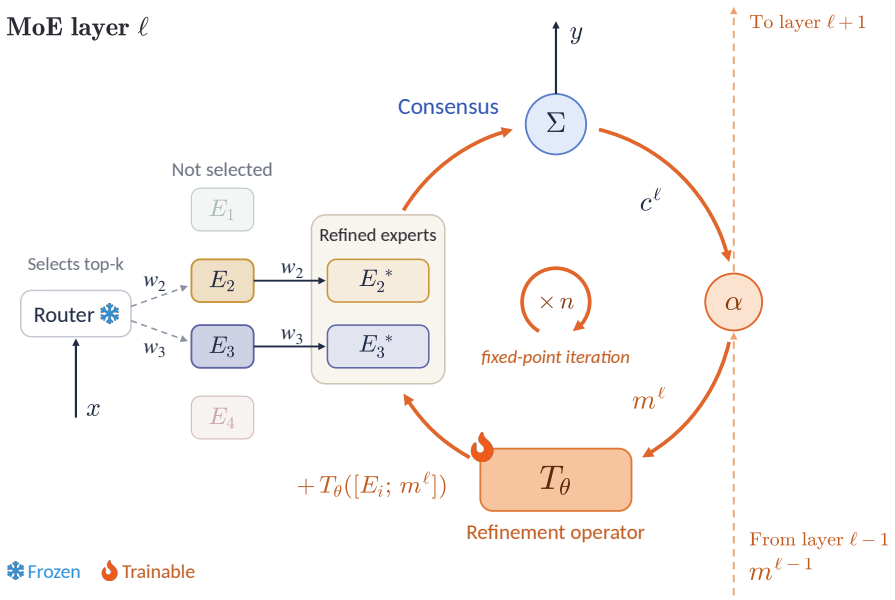

Figure 1: **CoFFiE.** The pretrained router selects the active experts, and their frozen outputs $E_i(x)$ serve as anchors. The refinement operator $\mathcal{T}_\theta$, a small trainable network, adds a correction to each active expert computed from its own state and the route-weighted consensus c^ℓ; because the correction changes the consensus, the refined states are solved as a fixed point in n anchored steps. In the cross-layer variant $\mathcal{T}_\theta$ reads a learned affine combination m^ℓ of c^ℓ and the previous layer's $m^{\ell-1}$ (dashed). The layer output is $y = x + \sum_i w_i E_i^\star$, with w the pretrained routing weights and $\bar{w}$ their renormalisation over the selected set.

Why the fixed point. A single application of $\mathcal{F}_\theta$ refines every expert against the consensus of *unrefined* neighbours, which is stale once the experts move; only the fixed point (3) makes each refined representation and the consensus it is conditioned on agree.

3.3 Instantiating the refinement operator

The refinement operator $\mathcal{T}_\theta$ is a small bottleneck network; the experts, the backbone, and the router remain frozen. A minimal instantiation uses

$$\mathcal{T}_\theta(E_i,\, c) \;=\; \mathrm{MLP}_\theta\big([\,E_i\,;\,c\,]\big), \tag{5}$$

where $[\cdot\,;\cdot]$ denotes concatenation. The operator reads both the expert's own state and the route-weighted consensus c, realising the coupling in (3). The refinement is applied only over the top-k active experts, so the state has size kd rather than Nd. In a *cross-layer* variant, the consensus fed to $\mathcal{T}_\theta$ at layer ℓ is replaced by a learned affine combination $m^\ell = \alpha\, m^{\ell-1} + (1-\alpha)\, c^\ell$ with a per-layer scalar $\alpha = \tanh(\tilde{\alpha})$, with $\tilde{\alpha}$ trainable and initialised at α_0; $\alpha \equiv 0$ held fixed recovers the in-layer method exactly. We set $m^0 = 0$ at the first MoE layer; m^ℓ is computed from the consensus at the solved state, $c^\ell(E^\star)$, and passed to the next layer without gradient. The initial value is the one hyperparameter the variant introduces (Section 4.2). Apart from these per-layer scalars, only θ is trained, so the trainable footprint is one bottleneck network per layer, shared by the active experts.

Remark 1 (Identity initialisation)**.** *With the output projection of* MLP_θ *set to zero in* (5)*, the representation collapses to its pretrained value,* $E^\star = E(x)$*, and the layer reduces* exactly *to the standard MoE of* (1) *at the start of training. The refinement is then learned as a departure from the pretrained behaviour, which protects the pretrained model from being disrupted at initialisation.*

3.4 Forward solve

The fixed point (3) is computed by Picard iteration from the anchor,

$$E^{(0)} = E(x), \qquad E^{(t+1)} = \mathcal{F}_\theta\big(E^{(t)}\big), \tag{6}$$

run for a fixed number n of steps without acceleration or early stopping. Every iterate is re-anchored at the frozen expert outputs, so the state never accumulates its own corrections; an accumulating variant, $E \leftarrow E + \mathcal{T}_\theta(\cdot)$, destabilised in preliminary runs. The forward cost is $n+1$ evaluations of $\mathcal{T}_\theta$ on the k active experts (n without gradient tracking and one differentiable application, Section 3.5) and is deterministic given n. Accelerated solvers were not needed: in fp32 the plain iteration reaches a median residual of 0.0165 at $n{=}3$ and 0.0043 at $n{=}5$ (Section 3.6).

3.5 Backward pass via implicit differentiation

Let θ denote the trainable parameters of the operator. Differentiating the equilibrium condition (4) at $E^\star$ and applying the implicit function theorem gives

$$\frac{\partial E^\star}{\partial \theta} \;=\; -\Big(I - \tfrac{\partial \mathcal{F}_\theta}{\partial E}\Big)^{-1} \frac{\partial \mathcal{R}_\theta}{\partial \theta}. \tag{7}$$

The corresponding vector–Jacobian products can be computed matrix-free without backpropagating through the forward solver (Bai et al., 2019). In practice, we use the cheaper one-step (phantom) gradient (Geng et al., 2021; Fung et al., 2022). Let $\widehat{E} = E^{(n)}$ denote the final forward iterate. The n solver iterations are run without gradient tracking, followed by one differentiable application of $\mathcal{F}_\theta$ at $\widehat{E}$ (Algorithm 1, Appendix A). This retains the leading term of the implicit-gradient inverse while avoiding backpropagation through the solver trajectory. Since the pretrained experts are frozen, gradients update only the refinement operator, and training memory is independent of the number of forward solver iterations.

3.6 Convergence

Write the operator of (5) as $\mathcal{T}_\theta(u, c) = W_2\, \sigma(W_1^u u + W_1^c c)$, with $W_1 = [\, W_1^u \; W_1^c \,]$ split by input block, σ the SiLU nonlinearity and $L_\sigma = \sup_z |\sigma'(z)| \approx 1.10$. On $\mathbb{R}^{kd}$ take the norm $\|E\|_\infty = \max_i \|E_i\|_2$. In the cross-layer variant the operator reads $\alpha\, m^{\ell-1} + (1-\alpha)\, c(E)$ with $m^{\ell-1}$ fixed for the layer; $\alpha{=}0$ is the in-layer case.

Proposition 1 (Contraction of the coupling map)**.** *For all $E, E' \in \mathbb{R}^{kd}$,*

$$\|\mathcal{F}_\theta(E) - \mathcal{F}_\theta(E')\|_\infty \;\le\; \kappa\, \|E - E'\|_\infty, \qquad \kappa \;=\; L_\sigma\, \|W_2\|_2 \big(\, \|W_1^u\|_2 + |1-\alpha|\, \|W_1^c\|_2 \big). \tag{8}$$

If $\kappa < 1$, the fixed point $E^\star$ exists and is unique, and the iteration (6) *converges to it geometrically,*

$$\Big\| E^{(t)} - E^\star \Big\|_\infty \le \kappa^t \, \Big\| E^{(0)} - E^\star \Big\|_\infty .$$

The proof is in Appendix B. Its key step is that the consensus is a convex combination, $\bar{w} \in \Delta^{k-1}$, and is therefore non-expansive: $\|c(E) - c(E')\|_2 \le \|E - E'\|_\infty$. Coupling k experts thus introduces no factor of k into the bound.

Proposition 1 gives a sufficient global condition for convergence. At initialisation, $W_2 = 0$ and hence $\kappa = 0$. After training, however, this global bound is no longer satisfied: for the $h{=}64$ in-layer operator, κ ranges from 2.3 to 5.9 across layers (Table 4). This does not determine the local behaviour near the solution, which is governed by the Jacobian $J = \partial \mathcal{F}_\theta / \partial E$.

We therefore measure J directly in fp32 on the trained $h{=}64$ in-layer model (seed 42), using 200 tokens at every MoE layer. For each layer we report the largest estimated spectral radius across tokens. Across the 16 layers, the median is $\rho(J) = 0.62$, and 15 of 16 layers satisfy $\rho(J) < 1$. The only exception is layer 10, with $\rho(J) = 1.002$, where the Picard residual correspondingly plateaus rather than continuing to decrease. In contrast, $\|J\|_2 > 1$ at every layer (median 2.62), showing that the trained operator is not a contraction in the Euclidean norm even where the iteration is locally stable. This gap between $\rho(J)$ and $\|J\|_2$ reflects the strong non-normality of the learned Jacobian.

The measured dynamics agree with the forward solver: the median residual decreases from 0.0165 after $n{=}3$ steps to 0.0043 after $n{=}5$ in fp32. Thus the fixed-point iteration is empirically locally stable at almost all measured in-layer states, despite the failure of the global contraction bound after training. The cross-layer variant shows the same behaviour more uniformly, with $\rho(J) < 1$ at all 16 measured layers (Table 5). We use $n{=}3$, at which downstream accuracy has already saturated (Section 4.2).

4 EXPERIMENTS

Setup. We evaluate on OLMoE-1B-7B (Muennighoff et al., 2025), a fully open MoE LLM with 64 experts per layer and top-8 routing, following the fine-tuning protocol of PERFT (Liu et al., 2024): models are trained on the Commonsense170K instruction set (Hu et al., 2023) and evaluated zero-shot on eight commonsense reasoning benchmarks (BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-easy and ARC-challenge (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018)), reporting per-task accuracy and the average. CoFFiE inserts the coupled refinement into every MoE layer; the experts, backbone, and router are frozen, and only the bottleneck operator MLP_θ (hidden width h) and the per-layer coefficients $\tilde{\alpha}$ are trained, with learning rate 2×10^{-4} shared across all configurations except where stated. Training uses AdamW in bf16, batch size 32 (micro-batch 16 on two GPUs with DDP), 3 epochs, sequence cutoff 256, and a held-out validation set of 120 examples; rows marked ±std aggregate 3 seeds. OLMoE has d=2048 and 16 MoE layers, and MLP_θ has no bias terms, so the $3dh$ count is exact. Baselines are re-run by us under the same data, epochs and evaluation harness; PERFT variants use the hyperparameters of Liu et al. (2024). All mechanism ablations (iteration count, coupling composition, operator capacity, cross-layer coefficient) are conducted on OLMoE-1B-7B; the larger backbones serve to test whether the main result transfers, not to repeat the analysis.

Trainable is the total number of parameters held in gradient and optimiser state, i.e. all stored adapter parameters regardless of how many are activated per token. Unless stated otherwise, all CoFFiE rows use the cross-layer variant with $\tilde{\alpha}$ initialised at -0.5 and n=3 solver iterations with the one-step gradient of Algorithm 1; rows marked "in-layer" set α=0 and do not train it.

Baselines. We compare against representative PEFT strategies under the same protocol: LoRA applied to the attention matrices (qvLoRA), which leaves the MoE blocks unadapted, and the PERFT family (Liu et al., 2024), whose routed variant PERFT-R (an independently routed mixture of adapter experts) is the strongest configuration in its design space. At h=88 our trainable budget (8.65M) exactly matches PERFT-R with LoRA rank 16 and Top8/8 adapter routing, enabling a comparison at an identical budget.

Method	# Trainable	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
OLMoE-1B-7B (64 experts, top-8)										
qvLoRA$_{64}$ (W_q, W_v@Attn)	8.39M	66.41	81.12	71.87	45.03	64.62	79.55	62.38	68.74	67.47
PERFT-E (LoRA$_2$, Top8/64)	8.4M	65.54	79.11	73.59	50.06	**73.24**	77.27	58.70	**72.80**	68.79
PERFT-R (LoRA$_{16}$, Top8/8)	8.65M	65.93	77.58	74.41	**55.14**	71.98	76.47	57.59	71.40	68.81
CoFFiE (h=64)	6.29M	68.59	81.50	73.69	49.47	68.51	80.98	62.29	71.00	69.50
CoFFiE (h=88)	8.65M	**69.02**	**81.66**	**75.54**	49.42	68.03	**81.23**	**64.51**	71.00	**70.05**
Mixtral-8×7B (8 experts, top-2)										
LoRA$_8$ (W_q, W_v@Attn)	3.41M	72.31	90.88	79.96	88.92	82.60	92.14	84.61	84.95	84.55
PERFT-S (LoRA$_{16}$)	4.19M	75.11	90.26	**81.63**	94.26	84.85	92.85	81.40	87.60	85.99
PERFT-R (LoRA$_8$, Top2/2)	4.46M	74.68	89.77	81.47	94.33	86.27	92.05	81.48	89.80	86.23
PERFT-R (LoRA$_{16}$, Top1/4)	18.9M	72.84	89.12	80.40	92.69	84.37	91.84	82.25	85.80	84.91
PERFT-R (LoRA$_8$, Top2/4)	9.4M	74.71	90.10	79.38	94.18	85.71	92.09	81.31	85.80	85.41
PERFT-R (LoRA$_8$, Top2/8)	21.0M	73.76	89.12	**81.63**	94.51	85.16	91.67	80.20	87.80	85.48
PERFT-E (LoRA$_8$, Top2/8)	16.8M	74.13	90.21	80.81	91.36	86.42	92.21	81.06	88.60	85.60
CoFFiE (h=8)	3.15M	76.88	91.24	81.12	94.89	86.90	94.49	**86.01**	89.80	87.66
CoFFiE (h=16)	6.29M	**77.92**	**92.27**	80.86	**95.31**	**88.71**	**94.78**	**86.01**	**90.40**	**88.28**
Qwen3-30B-A3B (128 experts, top-8)										
PERFT-S (LoRA$_{44}$, shared)	8.65M	73.70	90.21	81.53	94.36	86.82	96.34	89.42	92.40	88.10
PERFT-R (LoRA$_5$, Top8/8)	8.65M	75.14	91.35	**82.86**	95.78	87.21	97.81	92.83	94.20	89.65
CoFFiE (h=29)	8.55M	**76.64**	**92.71**	82.14	**95.96**	**89.03**	**98.40**	**94.28**	**94.60**	**90.47**

Table 1: **Commonsense reasoning across three MoE backbones.** All rows are single-seed; three-seed statistics for the OLMoE CoFFiE rows are in Table 3 and Appendix I.

Arch.	Strategy	# Trainable	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Avg.
OLMoE-1B-7B									
LoRA_{16}	W_q, W_v@Attn	2.10M	19.67	8.41	47.36	**23.55**	51.02	28.44	29.74
LoRA_{16}	PERFT-R (Top2/2)	1.11M	26.50	8.49	52.15	20.87	56.69	32.30	32.83
LoRA_{32}	PERFT-R (Top1/2)	2.22M	27.17	9.33	50.89	20.87	57.09	32.00	32.89
LoRA_{32}	PERFT-R (Top2/4)	8.66M	28.00	9.10	49.37	19.29	57.09	33.20	32.67
LoRA_{8}	PERFT-E (Top8/64)	33.5M	**28.33**	7.81	43.80	21.26	57.28	32.60	31.85
CoFFiE_{8}	fixed-point refine	0.79M	27.00	8.42	53.42	18.11	58.07	35.40	33.40
CoFFiE_{32}	fixed-point refine	3.15M	24.33	8.95	**55.95**	18.50	56.89	33.90	33.09
CoFFiE_{64}	fixed-point refine	6.29M	26.83	**9.70**	54.18	20.47	57.28	**37.00**	**34.25**
CoFFiE_{88}	fixed-point refine	8.65M	26.33	9.02	54.43	17.72	**59.84**	34.10	33.57
Mixtral-8×7B									
LoRA_{8}	W_q, W_v@Attn	3.41M	59.12	51.44	89.05	29.31	88.63	68.36	64.32
LoRA_{8}	PERFT-R (Top2/2)	4.46M	82.83	55.80	87.59	29.92	89.76	68.30	69.04
LoRA_{8}	PERFT-E (Top2/8)	20.12M	79.00	54.06	87.34	29.13	88.98	70.30	68.13
CoFFiE_{8}	fixed-point refine	3.15M	85.50	**61.49**	**89.87**	**31.50**	**91.14**	74.40	**72.32**
CoFFiE_{16}	fixed-point refine	6.29M	**88.17**	59.36	87.09	30.71	90.75	**74.70**	71.80
CoFFiE_{22}	fixed-point refine	8.65M	85.33	59.14	88.35	27.95	**91.14**	73.40	70.89

Table 2: **Arithmetic reasoning on OLMoE-1B-7B and Mixtral-8×7B.** # Trainable as in Table 1. On OLMoE, CoFFiE_{64} reaches 34.25 with 6.29M parameters, $+1.36$ over the strongest baseline (32.89); GSM8K is a floor task on this backbone (below 10 everywhere). On Mixtral, CoFFiE_{8} improves the average by $+3.28$ over the strongest baseline with the smallest budget in the block, with the largest gains on the two most multi-step benchmarks (GSM8K $+5.69$, SVAMP $+4.10$). Several test sets in this suite are small, giving the six-task average a sampling standard error of ~ 0.7 points, so differences among CoFFiE widths carry no signal.

4.1 Main results: comparison with PEFT baselines

Table 1 presents the main comparison. At an exactly matched trainable budget of 8.65M parameters our method outperforms every baseline outright (70.05 vs. 68.81 for PERFT-R), and at 73% of the budget it is already on par with PERFT-R (69.50 at 6.29M); the corresponding three-seed means are 69.64 ± 0.36 and 68.78 ± 0.63. It does so through a structurally simpler mechanism: a single bottleneck operator shared by the active experts of each layer, rather than a mixture of per-layer adapter experts with an additional router trained from scratch. We read this as evidence that the adaptation budget is better spent on coordinating the frozen experts than on adding routed weight-space capacity.

Larger backbones. Because one operator per layer serves all active experts, the trainable count of CoFFiE does not grow with the number of experts. On Mixtral-8×7B, CoFFiE at 3.15M reaches 87.66, above the best PERFT configuration (86.23 at 4.46M), and at 6.29M reaches 88.28, above every PERFT variant up to 21.0M. On Qwen3-30B-A3B (Yang et al., 2025) (48 layers, 128 experts, top-8) the comparison is at a near-matched budget (8.55M vs. 8.65M; with 48 layers no integer h lands exactly on 8.65M),[2] and despite a much stronger base (the routed baseline already averages 89.65), CoFFiE improves on PERFT-R by $+0.8$.

Arithmetic reasoning. To test the mechanism beyond commonsense reasoning, we repeat the protocol on the Math50K instruction set (Hu et al., 2023) with six arithmetic benchmarks (MultiArith (Roy & Roth, 2015), GSM8K (Cobbe et al., 2021), AddSub (Hosseini et al., 2014), AQuA (Ling et al., 2017), SingleEq (Koncel-Kedziorski et al., 2015), SVAMP (Patel et al., 2021)). In Table 2, CoFFiE at h=64 (6.29M) reaches the best OLMoE average of 34.25, against 32.89 for the strongest PERFT configuration (PERFT-R, LoRA_{32}, Top1/2, 2.22M) and 29.74 for attention-only qvLoRA_{16} (2.10M); even the smallest operator (h=8, 0.79M) attains 33.40. On Mixtral the smallest operator improves the average by $+3.28$ over the strongest baseline. The remainder of this section dissects where this performance comes from.

[2] PERFT-E cannot be instantiated at this budget on this backbone: one adapter per expert at rank r=1 already requires 25.2M trainable parameters, $2.9\times$ the budget. Per-expert schemes do not scale with the number of experts times layers, which leaves PERFT-R as the only common baseline that can be parameter-matched across all three backbones.

Table 3: **Cumulative ablation on OLMoE-1B-7B** (h=64, 6.29M trainable, commonsense average, three-seed mean±std). Components are added top to bottom. The expert-only operator reads $[E_i]$ and is given width h'=$1.5h$ so that its trainable count equals that of the consensus operator ($2d \cdot 1.5h = 3dh$). With no feedback the consensus is computed once from the frozen anchors (n=0); with feedback, the state is solved toward self-consistency (n=3). The cross-layer row is the default configuration. Per-seed values in Appendix I.

Input to $\mathcal{T}_\theta$	Feedback	Avg.	Δ
$[E_i]$, h'=$1.5h$	–	67.26±0.25	–
$[E_i;\, c(E^{(0)})]$	n=0	67.56±0.51	+0.30
$[E_i;\, c(E)]$	n=3	68.06±0.09	+0.80
$[E_i;\, m^\ell]$ (cross-layer)	n=3	**68.78**±0.63	+1.52

4.2 Where does the gain come from? A cumulative ablation

Table 3 adds the three components of CoFFiE one at a time, at a fixed trainable budget, starting from an operator that reads only the expert's own state.

Consensus. The first and third rows together isolate the contribution of cross-expert information: a parameter-matched expert-only operator reaches 67.26, whereas the full in-layer operator reaches 68.06, a $+0.80$ gain that cannot be attributed to additional capacity; as the next paragraph shows, most of this gain materialises only once the consensus is iterated. Feeding the operator only the difference $[c - E_i]$ performs clearly worse (Appendix F), showing that the operator benefits from both the individual expert state and the consensus.

Feedback. The second row reads the consensus once from the frozen anchors and never revises it; the third iterates so that the refined states and their consensus move toward self-consistency. This change gives a positive gain and holds for every seed. The dependence is a property of the trained operator rather than of the training configuration: taking n=5-trained checkpoints and removing feedback only at inference costs 3.8 points, while a single refinement step recovers the loss substantially (Table 7, Appendix E). This behaviour is consistent with the rapid residual decay observed in Section 3.6. Beyond one refinement round, downstream accuracy saturates: n=$1, 3, 5$ are statistically indistinguishable. We therefore use n=3 without further tuning.

Cross-layer consensus. The last row lets each layer read its own consensus together with the state carried from the previous layer. The gain holds at every operator width we test, from 0.79M to 12.58M trainable parameters on OLMoE (Table 10, Appendix H) and at every width on Mixtral (Appendix J). In three-seed paired comparisons (h=64 at $n \in \{0, 1, 3, 5\}$ and h=88 at n=3, Appendix I), the cross-layer variant wins 15 of 15 comparisons with mean $+0.95$ (one-sided sign test $p < 10^{-4}$). The initialisation of α determines whether the variant helps (Appendix K): from $\alpha_0 \in \{+0.5,\, 0\}$ it remains at the in-layer level (68.07, 67.81 vs. 68.06 ± 0.09), whereas from $\alpha_0 \in \{-0.5,\, -1.0\}$ it reaches 69.50 and 69.59. In every run, α moves only in the negative direction and does not settle; within this negative regime, the exact value matters little (0.2 points across final $\alpha \in [-0.98, -0.76]$). We therefore treat α_0 as a hyperparameter with a sign prior and use -0.5 throughout. Zeroing the carried state $m^{\ell-1}$ at evaluation returns the model to in-layer accuracy (68.01 vs. 68.03), showing that the gain comes from cross-layer information rather than the $(1 - \alpha)$ rescaling.

With the cross-layer state, n=0 reaches 69.01 (three-seed mean; Appendix I). We conjecture that the carried state already supplies much of the cross-expert interaction that the in-layer solve would otherwise provide.

Operator capacity. Table 9 (Appendix G) scales the bottleneck width h. Accuracy increases smoothly from h=8 (0.79M) to h=88 (8.65M); already at h=64 (0.491% of activated parameters), the method recovers most of the accuracy of the largest configuration. The h=88 configuration is compared with PERFT-R at a matched trainable budget in Table 1.

4.3 COST OF THE MECHANISM

The preceding ablations attribute the gains to the consensus and the cross-layer state; this section reports what they cost. On OLMoE-1B-7B (2×H200, bf16, 3 epochs) the h=64 in-layer operator at n=3 trains in 2.71 h and the default cross-layer configuration in 2.82 h, against 2.67 h for PERFT-R at the matched budget and 5.06 h for PERFT-E, which evaluates an adapter on every one of the 64 experts. The run-to-run ranges of PERFT-R and CoFFiE overlap. Each solver step costs about 3% at constant peak memory, and the cross-layer term adds 4% (Appendix D).

Limitations. Three observations bound our claims. With the cross-layer state, accuracy no longer depends on the number of solver steps (Appendix I); the cross-layer gain appears only for negative initialisations of α (Appendix K); and the Mixtral and Qwen3 results are single-seed. Why the two routes to cross-expert information do not add, and why the sign of α_0 matters, remains to be explained; we conjecture that $m^{\ell-1}$ already carries a consensus refined by the previous layer, leaving the in-layer solve little to correct, but we have not tested this.

5 CONCLUSION

We presented CoFFiE, a parameter-efficient fine-tuning method that adapts how the experts selected by a pretrained MoE coordinate while keeping the backbone, router, and experts frozen. The refined expert states are defined implicitly as a fixed point in which each expert is refined using the route-weighted consensus of the active experts, and training uses a one-step implicit gradient whose memory cost is independent of the number of solver iterations. Across three MoE backbones, CoFFiE outperforms the strongest routed-PEFT baselines we evaluate under matched or smaller trainable budgets, and ablations show that both cross-expert consensus and cross-layer information contribute to the gains. Future work will study stronger convergence guarantees and the cross-layer mechanism across a broader range of MoE models.

REPRODUCIBILITY STATEMENT

All experiments use publicly available checkpoints (OLMoE-1B-7B, Mixtral-8×7B, Qwen3-30B-A3B) and public instruction sets and benchmarks (Commonsense170K and Math50K with their standard evaluation suites, Section 4). The method is fully specified in Section 3: the coupled fixed point in (3), the operator instantiation in (5), the cross-layer term in Section 3.3, the forward solve in Section 3.4 and the one-step gradient in Section 3.5; pseudocode is given in Algorithm 1 (Appendix A) and the proof of Proposition 1 in Appendix B. Training hyperparameters are stated in the Setup paragraph of Section 4 (AdamW, bf16, learning rate 2×10^{-4}, batch size 32, 3 epochs, sequence cutoff 256, n=3, $\tilde{\alpha}$ initialised at -0.5) and are shared across all CoFFiE configurations; baselines are re-run under the same data, epochs and evaluation harness with the hyperparameters of Liu et al. (2024). The trainable-parameter accounting used in every table is defined in Section 4. Per-seed results for all three-seed comparisons are in Appendix I, and hardware and wall-clock costs in Section 4.3 and Appendix D. Code and configuration files are available at `https://anonymous.4open.science/r/CoFFiE-ano-DD26/`.

ETHICS STATEMENT

This work proposes a parameter-efficient fine-tuning method for pretrained Mixture-of-Experts language models. It introduces no new datasets and involves no human subjects; all training data and benchmarks are existing public resources used under their original licences. The method adapts frozen base models and does not by itself alter their safety properties: any biases or harmful capabilities present in the underlying checkpoints are inherited unchanged, and the same mitigations that apply to the base models apply here. We do not foresee ethical concerns specific to this contribution.

AI USE STATEMENT

We used generative AI tools (Claude, Anthropic, and AI-assisted coding tools) to assist in implementing parts of the training and evaluation code, to interpret ablation results, and to give feedback on the presentation of experiments. We also used them to draft and edit text, draft the layout of

Figure 1, search literature, check bibliography entries, and re-compute table statistics from per-seed values. All AI-assisted code, text and numbers were verified by the authors, who take responsibility for the final content.

REFERENCES

Donald G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12 (4):547–560, 1965.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Multiscale deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Ting Bai, Yue Yu, Le Huang, Zenan Xu, Zhe Zhao, and Chuan Shi. GMoE: Empowering LLMs fine-tuning via MoE graph collaboration. *arXiv preprint arXiv:2412.16216*, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, 2024.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, 2024.

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

Jiarui Feng, Hanqing Zeng, Karish Grover, Ruizhong Qiu, Yinglong Xia, Qiang Zhang, Qifan Wang, Ren Chen, Dongqi Fu, Jiayi Liu, et al. Dag-moe: From simple mixture to structural aggregation in mixture-of-experts. *arXiv preprint arXiv:2606.01062*, 2026.

Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley Osher, and Wotao Yin. JFB: Jacobian-free backpropagation for implicit networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 2022.

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. LLM-Adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

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

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015.

Stepan Kulibaba, Kirill Labzin, Artem Dzhalilov, Roman Pakhomov, Oleg Svidchenko, Alexander Gasnikov, and Aleksei Shpilman. SDG-MoE: Signed debate graph mixture-of-experts. *arXiv preprint arXiv:2605.08322*, 2026.

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. In *International Conference on Learning Representations*, 2021.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. MixLoRA: Enhancing large language models fine-tuning with LoRA-based mixture of experts. *arXiv preprint arXiv:2404.15159*, 2024.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

Yilun Liu, Yunpu Ma, Shuo Chen, Zifeng Ding, Bailan He, Zhen Han, and Volker Tresp. PERFT: Parameter-efficient routed fine-tuning for mixture-of-expert model. *arXiv preprint arXiv:2411.08212*, 2024.

Bo Lv, Chen Tang, Zifan Zheng, Bohao Yang, Kun Zhao, Ning Liao, Xiaoxing Wang, Feiyu Xiong, Zhiyu Li, Nayu Liu, and Jingchi Jiang. GraphMoE: Amplifying cognitive depth of mixture-of-experts network via introducing self-rethinking mechanism. *arXiv preprint arXiv:2501.07890*, 2025.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoE: Open mixture-of-experts language models. In *International Conference on Learning Representations*, 2025.

Minh-Khoi Nguyen-Nhat, Rachel S. Y. Teo, Laziz Abdullaev, Maurice Mok, Viet-Hoang Tran, and Tan Minh Nguyen. Modeling expert interactions in sparse mixture of experts via graph structures. *arXiv preprint arXiv:2510.16411*, 2025.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. SocialIQA: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Zihan Wang, Deli Chen, Damai Dai, Runxin Xu, Zhuoshu Li, and Y. Wu. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 784–801, 2024.

Zihan Wang, Rui Pan, Jiarui Yao, Robert Csordas, Linjie Li, Lu Yin, Jiajun Wu, Tong Zhang, Manling Li, and Shiwei Liu. Chain-of-experts: Unlocking the communication power of mixture-of-experts models. *arXiv preprint arXiv:2506.18945*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient MoE for instruction tuning. In *International Conference on Learning Representations*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Hao Zhao, Zihan Qiu, Huijia Wu, Zili Wang, Zhaofeng He, and Jie Fu. HyperMoE: Towards better mixture of experts via transferring among experts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10605–10618, 2024.

A SOLVER DETAILS

Algorithm 1 gives the complete forward and backward procedure for one MoE layer and one token. The solver runs the anchored Picard iteration of Eq. (6) for a fixed number n of steps entirely without gradient tracking, with no acceleration and no early stopping. A single differentiable application of the map at the resulting state then implements the one-step (phantom) gradient of Eq. (7): memory is constant in the number of solver steps, and because the expert outputs $E_i(x)$ enter only as frozen anchors, no gradient reaches the expert or router weights.

Algorithm 1 CoFFiE refinement of MoE experts (one layer, one token). Shown for the in-layer operator; the cross-layer variant replaces c by $\alpha\, m^{\ell-1} + (1-\alpha)\, c$ inside $\mathcal{F}_\theta$ (Section 3.3).

Require: token state $x \in \mathbb{R}^d$; frozen router G_ϕ; frozen experts $\{E_e\}$; top-k; operator $\mathcal{T}_\theta$; steps n

Sparse routing with the frozen pretrained router:

$(w, \mathcal{S}) \leftarrow G_\phi(x)$ ▷ pretrained weights $w \in \mathbb{R}^k$, selected experts $\mathcal{S}$

$E_i^0 \leftarrow E_{\mathcal{S}_i}(x), \quad i = 1, \dots, k$ ▷ anchor: frozen expert outputs

Fixed-point solve (no gradient; memory independent of n):

$E \leftarrow E^0$

for $t = 1$ **to** n **do**

  $E \leftarrow \mathcal{F}_\theta(E)$

end for

Single differentiable step (one-step gradient):

$\widetilde{E} \leftarrow \mathcal{F}_\theta(\mathrm{stopgrad}(E))$

return $y \leftarrow x + \sum_{i=1}^k w_i\, \widetilde{E}_i$ ▷ pretrained routing weights are preserved

function $\mathcal{F}_\theta(E)$ ▷ coupling map, Eq. (3)

  $\bar{w}_j \leftarrow w_j \big/ \sum_{j'} w_{j'}$ ▷ renormalise so that $\sum_j \bar{w}_j = 1$

  $c \leftarrow \sum_{j=1}^k \bar{w}_j\, E_j$ ▷ route-weighted consensus

  $\Delta_i \leftarrow \mathcal{T}_\theta\big([\, E_i \,;\, c\,]\big)$ ▷ bottleneck MLP, $\mathbb{R}^{2d} \to \mathbb{R}^d$

  return $E_i^0 + \Delta_i$ ▷ re-anchored at the frozen expert outputs

end function

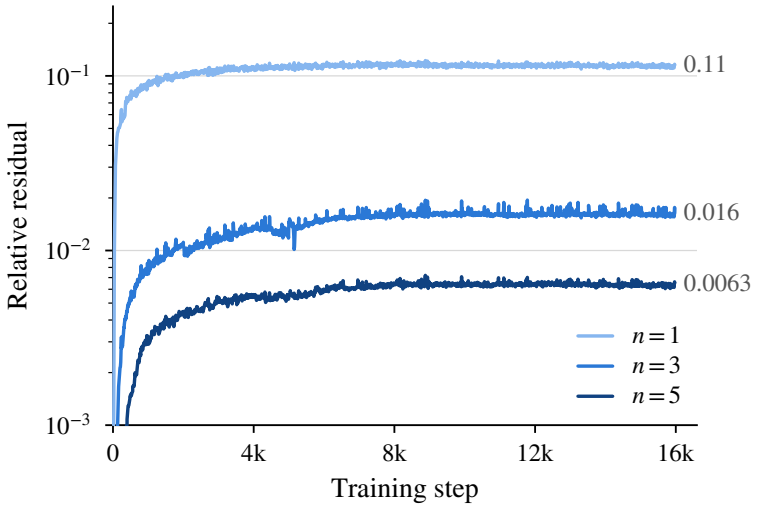

Figure 2: **The solver residual remains stable throughout training.** Layer-averaged relative residual of the final solver step, $\|E^{(n)} - E^{(n-1)}\| \,/\, \|E^{(n-1)}\|$, logged during training on OLMoE-1B-7B (h=64, in-layer) for runs trained with $n \in \{1, 3, 5\}$.

B CONTRACTION: PROOF AND MEASURED CONSTANTS

Proof of Proposition 1. Fix $E, E' \in \mathbb{R}^{kd}$ and let $\delta = \|E - E'\|_\infty$. Since $\bar{w} \in \Delta^{k-1}$,

$$\|c(E) - c(E')\|_2 \;=\; \Big\| \textstyle\sum_j \bar{w}_j \,(E_j - E'_j) \Big\|_2 \;\le\; \textstyle\sum_j \bar{w}_j \,\|E_j - E'_j\|_2 \;\le\; \delta.$$

In the cross-layer variant the operator reads $\alpha\, m^{\ell-1} + (1-\alpha)\, c(E)$ with $m^{\ell-1}$ fixed for the layer, so the difference of consensus inputs is bounded by $|1-\alpha|\,\delta$. Since σ is L_σ-Lipschitz, for every i,

$$\|\mathcal{T}_\theta(E_i, c) - \mathcal{T}_\theta(E'_i, c')\|_2 \;\le\; L_\sigma \,\|W_2\|_2 \,\big(\, \|W_1^u\|_2 \,\|E_i - E'_i\|_2 + \|W_1^c\|_2 \,\|c - c'\|_2 \,\big) \;\le\; \kappa\,\delta.$$

The anchor $E_i(x)$ cancels in $\mathcal{F}_\theta(E)_i - \mathcal{F}_\theta(E')_i$, and taking the maximum over i gives (8). For $\kappa < 1$, existence, uniqueness and the geometric rate are the Banach fixed-point theorem on $(\mathbb{R}^{kd}, \|\cdot\|_\infty)$. □

Measured constants. Table 4 lists the spectral norms entering (8) for the h=64 in-layer operator (n=3, seed 42) at the end of training, with $L_\sigma = 1.0998$ for SiLU and α=0. κ exceeds one in every layer, so the global condition of Proposition 1 does not hold for the trained operator; we therefore measure the local Jacobian directly below.

Layer	$\|W_2\|_2$	$\|W_1^u\|_2$	$\|W_1^c\|_2$	κ
0	1.455	1.734	1.786	5.64
1	0.786	1.354	1.354	2.34
2	1.182	1.917	1.904	4.97
3	1.380	1.912	1.985	5.92
4	1.057	1.835	1.794	4.22
5	1.360	1.925	1.945	5.79
6	1.339	1.838	1.843	5.42
7	1.288	1.802	1.821	5.13
8	1.314	1.836	1.856	5.33
9	1.159	1.701	1.724	4.36
10	1.106	1.746	1.755	4.26
11	1.036	1.744	1.754	3.98
12	1.082	1.531	1.526	3.64
13	1.022	1.549	1.465	3.39
14	1.144	1.553	1.538	3.89
15	0.975	1.345	1.338	2.88
mean	1.168	1.708	1.712	4.45
max	1.455	1.925	1.985	5.92

Table 4: **Spectral norms of the trained operator and the resulting constant κ of Proposition 1.** OLMoE-1B-7B, h=64, in-layer, n=3, seed 42; $L_\sigma = 1.0998$. The global contraction condition $\kappa < 1$ holds at initialisation ($W_2 = 0$) and is exceeded in every layer after training.

Local Jacobian measurements. We additionally measure the Jacobian $J = \partial\mathcal{F}_\theta/\partial E$ directly in fp32 on the trained h=64 model (seed 42), using 200 tokens at every MoE layer. For each layer, we report the largest estimated spectral radius across the sampled tokens.

Variant	median $\rho(J)$	max $\rho(J)$	$\rho(J) \ge 1$	median $\|J\|_2$	range $\|J\|_2$	residual n=3/5
in-layer	0.62	1.002	1/16	2.62	[1.03, 3.65]	0.0165 / 0.0043
cross-layer	0.48	0.90	0/16	3.19	[1.52, 4.49]	0.0152 / 0.0029

Table 5: **Post-training local Jacobian measurements.** OLMoE-1B-7B, h=64, seed 42, fp32, 200 tokens per layer. For the in-layer operator, 15 of 16 layerwise maxima satisfy $\rho(J) < 1$; the exception is layer 10 with $\rho(J) = 1.002$. For the cross-layer operator, all 16 layerwise maxima satisfy $\rho(J) < 1$. In both variants, $\|J\|_2 > 1$ at every layer.

These measurements distinguish local stability from a global contraction guarantee: the latter is not certified after training, while the measured spectral radii are below one for 15 of 16 in-layer layers and all 16 cross-layer layers.

C TRUNCATION ERROR OF THE ONE-STEP GRADIENT

The exact gradient (7) applies $(I-J)^{-1} = \sum_{k\geq 0} J^k$, which converges when $\rho(J) < 1$; the one-step gradient of Section 3.5 keeps the k=0 term. A geometric operator-norm bound on the truncation error, $\left\|(I-J)^{-1} - I\right\| \leq \|J\| / (1 - \|J\|)$, requires $\|J\| < 1$. Since the measured post-training $\|J\|_2$ exceeds one, this bound does not apply to our trained operator. Conditions under which the truncated gradient remains a descent direction are given by Fung et al. (2022).

D TRAINING COST

Table 6 reports wall-clock training time on OLMoE-1B-7B (2×H200, bf16, DDP, batch 32, 3 epochs, 15,966 steps); median over three runs, range in brackets. The per-step cost of the solver is roughly constant ($+3\%$ per additional solver step), and peak memory does not change with n: the iterates are computed without gradient tracking and only the final differentiable application is stored.

Configuration	Time (h)	vs. reference
(a) Against PERFT baselines		
PERFT-R (LoRA_{16}, Top8/8)	2.67 [2.58, 2.71]	–
PERFT-E (LoRA_2, Top8/64)	5.06 [4.73, 5.40]	$+90\%$
CoFFiE, h=64, in-layer, n=3	2.71 [2.68, 2.72]	$+1.5\%$
CoFFiE, h=88, in-layer, n=3	2.70 [2.69, 2.72]	$+1.1\%$
CoFFiE, h=64, cross-layer, n=3	2.82 [2.77, 2.89]	$+5.6\%$
(b) Solver steps (h=64, in-layer)		
n=0	2.47 [2.45, 2.51]	–
n=1	2.54 [2.54, 2.62]	$+2.8\%$
n=3	2.71 [2.68, 2.72]	$+9.7\%$
n=5	2.87 [2.85, 2.89]	$+16.2\%$

Table 6: **Training wall-clock time on OLMoE-1B-7B.** Median over three runs, range in brackets. Block (a) compares against the PEFT baselines at the matched budget; block (b) isolates the cost of the solver. The ranges of PERFT-R and in-layer CoFFiE overlap, so the $+1.5\%$ is within run-to-run variation.

E SOLVER STEPS

Table 7 separates whether feedback is necessary from how many solver steps the solve needs, complementing the cumulative ablation of Section 4.2. The first row trains and evaluates at each n: feedback (n=1) beats no feedback (n=0) for every seed ($+0.58$), and $n=1, 3, 5$ are statistically indistinguishable. The second row takes the n=5-trained checkpoints and changes only the number of solver steps at inference: removing feedback costs 3.8 points and a single step recovers it, consistent with the local Jacobian measurements in Table 5.

	n=0	n=1	n=3	n=5
trained at n (3 seeds)	67.56±0.51	**68.14**±0.62	68.06±0.09	68.06±0.40
n=5 checkpoint, eval-time n	64.50±0.22	68.16±0.41	68.30±0.19	**68.32**±0.22

Table 7: **Solver-step ablation for the in-layer operator.** OLMoE-1B-7B, h=64, lr 2×10^{-4}; values are three-seed mean±std. The top row trains and evaluates at each n; the bottom row changes n only at inference for checkpoints trained with n=5. Per-seed values are in Appendix I.

F COMPOSITION OF THE COUPLING TERM

Table 8 reports the full ablation on what the refinement operator reads, complementing the summary in Section 4.2. % Train. denotes # Trainable as a fraction of the 1.28B activated backbone parameters. Block (a) is the parameter-matched control: the expert-only variant $[E_i]$ is widened to $h'{=}1.5h$ so that its trainable count exactly equals that of the full operator $[E_i; c]$ ($2d \cdot 1.5h = 3dh$), and the consensus input still contributes $+0.80$ average accuracy at identical capacity (68.06 ± 0.09 vs. 67.26 ± 0.25, three seeds). The difference input $[c - E_i]$ at matched parameters averages 59.82, clearly worse (-8.24). Block (b) removes the width compensation: at fixed $h{=}64$, dropping the consensus costs 2.41 points relative to the full operator, and replacing the concatenated input with the difference costs a further 1.96, indicating that the operator needs the expert's own state and the consensus separately rather than any single fused signal.

Input to $\mathcal{T}_\theta$	Arch.	# Trainable	% Train.	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
(a) Parameter-matched (three-seed mean ± std)												
$[E_i; c]$	CoFFiE$_{64}$	6.29M	0.491	67.34	**80.70**	**72.62**	**47.53**	**64.72**	**81.07**	**60.84**	**69.67**	**68.06**$\pm$0.09
$[E_i]$	CoFFiE$_{96}$	6.29M	0.491	**67.74**	79.89	72.13	46.22	63.56	80.39	60.81	67.33	67.26$\pm$0.25
$[c - E_i]$	CoFFiE$_{96}$	6.29M	0.491	59.27	75.68	66.07	28.84	56.75	78.28	57.51	56.20	59.82$\pm$0.78
(b) Fixed h, unmatched parameters (single seed)												
$[E_i]$	CoFFiE$_{64}$	4.19M	0.327	**67.06**	**79.49**	**72.36**	**41.97**	**60.30**	**79.92**	**59.73**	**65.00**	**65.65**
$[c - E_i]$	CoFFiE$_{64}$	4.19M	0.327	64.22	77.48	70.06	41.39	58.88	78.87	57.00	61.60	63.69

Table 8: **Ablation on the composition of the cross-expert coupling term c.** Block (a) is the primary evidence: widening the expert-only variant to $h'{=}1.5h$ equalises the trainable parameter count exactly, and admitting c still yields $+0.80$ average accuracy, so the gain stems from cross-expert information rather than added capacity. In block (b) the width is held fixed instead, so the gain is confounded with parameter count; these rows are reported for qualitative reference only.

G OPERATOR CAPACITY

Table 9 sweeps the bottleneck width h for the in-layer operator. Average accuracy increases from $h{=}8$ (0.79M trainable parameters) through $h{=}128$ (12.58M) without saturating, and the $h{=}88$ point (8.65M) provides the exact budget match to PERFT-R used in Table 1. Because a single shared operator serves all active experts, the trainable and activated counts coincide at every width.

Arch.	Strategy	# Trainable	% Train.	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
CoFFiE$_8$	in-layer ($n{=}3$)	0.79M	0.061	63.21	74.76	65.10	33.48	53.91	76.64	53.07	54.80	59.37
CoFFiE$_{64}$	in-layer ($n{=}3$)	6.29M	0.491	67.74	81.12	72.47	48.20	64.56	**81.40**	59.98	68.80	68.03
CoFFiE$_{88}$	in-layer ($n{=}3$)	8.65M	0.675	68.29	81.72	73.90	47.62	66.46	81.36	62.88	**71.20**	69.18
CoFFiE$_{128}$	in-layer ($n{=}3$)	12.58M	0.982	**68.32**	**82.86**	**74.31**	**48.05**	**70.24**	**81.69**	**63.48**	71.00	**69.99**

Table 9: **Ablation on the operator width h.** The $h{=}88$ configuration (8.65M trainable parameters) is the one used for the budget-matched comparison with PERFT-R in Table 1.

H CROSS-LAYER CONSENSUS ON OLMOE-1B-7B ACROSS OPERATOR CAPACITY

Table 10 pairs the in-layer and cross-layer operators at four widths, and Table 11 gives the per-task numbers behind each pair.

# Trainable	0.79M	6.29M	8.65M	12.58M
in-layer	59.37	68.03	69.18	69.99
cross-layer	**61.55**	**69.50**	**70.05**	**70.76**
Δ	+2.18	+1.47	+0.87	+0.77

Table 10: **Cross-layer consensus on OLMoE-1B-7B across operator capacity** (commonsense average over eight benchmarks, n=3, single seed; per-task numbers in Table 11). Each pair differs only in the cross-layer mixture; the cross-layer row is the default configuration used throughout. Three-seed statistics are in Appendix I.

Method	# Trainable	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
CoFFiE$_8$	0.79M	63.21	74.76	65.10	33.48	53.91	76.64	53.07	54.80	59.37
+ cross-layer	0.79M	**65.87**	**77.80**	**67.09**	**33.95**	**55.96**	**78.03**	**55.46**	**58.20**	**61.55**
CoFFiE$_{64}$	6.29M	67.74	81.12	72.47	48.20	64.56	**81.40**	59.98	68.80	68.03
+ cross-layer	6.29M	**68.59**	**81.50**	**73.69**	**49.47**	**68.51**	80.98	**62.29**	**71.00**	**69.50**
CoFFiE$_{88}$	8.65M	68.29	**81.72**	73.90	47.62	66.46	**81.36**	62.88	**71.20**	69.18
+ cross-layer	8.65M	**69.02**	81.66	**75.54**	**49.42**	**68.03**	81.23	**64.51**	71.00	**70.05**
CoFFiE$_{128}$	12.58M	68.32	**82.86**	74.31	48.05	70.24	**81.69**	63.48	71.00	69.99
+ cross-layer	12.58M	**68.65**	81.45	**75.59**	**51.98**	**71.82**	81.23	**63.99**	**71.40**	**70.76**

Table 11: **Cross-layer consensus on OLMoE-1B-7B across operator capacity: per-task results** (commonsense reasoning, n=3, lr 2×10^{-4}, single seed). Each pair differs only in the cross-layer term; the unmarked row is the in-layer operator. Task-level comparisons favour the cross-layer variant in 26 of 32 cases; three-seed statistics are in Appendix I.

I PER-SEED RESULTS

Table 12 lists the per-seed averages behind the three-seed statistics in the main text: the paired cross-layer test (15 of 15 positive, mean $+0.95$, one-sided sign test $p < 10^{-4}$), the cumulative ablation of Table 3, and the n-ablation of Table 7.

	in-layer			cross-layer			
Config	s42	s1	s2	s42	s1	s2	Δ mean
h=64, n=0	67.83	66.98	67.88	69.12	68.99	68.91	+1.45
h=64, n=1	67.91	67.66	68.84	69.02	68.82	69.17	+0.86
h=64, n=3	68.03	67.99	68.16	69.50	68.31	68.53	+0.72
h=64, n=5	68.32	67.60	68.27	69.54	68.57	69.50	+1.14
h=88, n=3	69.18	68.84	69.22	70.05	69.45	69.42	+0.56

Table 12: **Per-seed commonsense averages** (OLMoE-1B-7B, lr 2×10^{-4}). All 15 paired differences are positive.

J CROSS-LAYER CONSENSUS ON MIXTRAL-8×7B

Table 13 repeats the paired cross-layer comparison on Mixtral-8×7B. The setting is less favourable to the mechanism than OLMoE: with top-2 routing the consensus mixes only two experts, and every configuration already exceeds 87 average accuracy, leaving little headroom. The cross-layer variant nevertheless improves every capacity pair ($+0.25$, $+0.46$, $+0.59$), consistent in direction with the OLMoE result.

Method	# Trainable	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
CoFFiE$_8$	3.15M	**77.13**	**91.29**	80.96	**95.16**	86.42	94.36	85.32	88.60	87.41
+ cross-layer	3.15M	76.88	91.24	**81.12**	94.89	**86.90**	**94.49**	**86.01**	**89.80**	**87.66**
CoFFiE$_{16}$	6.29M	**78.17**	91.24	**81.83**	93.88	87.92	94.57	84.73	90.20	87.82
+ cross-layer	6.29M	77.92	**92.27**	80.86	**95.31**	**88.71**	**94.78**	**86.01**	**90.40**	**88.28**
CoFFiE$_{22}$	8.65M	**77.74**	91.24	80.96	93.08	**88.08**	94.49	85.67	**89.40**	87.58
+ cross-layer	8.65M	77.65	**92.22**	**81.37**	**95.95**	88.00	94.49	**86.52**	89.20	**88.17**

Table 13: **Cross-layer consensus on Mixtral-8×7B across operator capacity** (commonsense reasoning, n=3, lr 2×10^{-4}, single seed). Each pair differs only in the cross-layer term; the unmarked row is the in-layer operator.

K CROSS-LAYER COEFFICIENT INITIALISATION

Table 14 varies the initialisation α_0 of the cross-layer coefficient. In every run the trained coefficients move in the negative direction and do not settle; the gain over the in-layer operator (68.06 ± 0.09, three seeds) appears only for $\alpha_0 < 0$, and within the negative regime the final value matters little. The main text uses $\alpha_0{=}{-}0.5$ throughout. For that run the trained coefficients give $|1-\alpha| \in [1.6,\ 1.9]$ across layers, the factor on $\|W_1^c\|_2$ in Proposition 1. Zeroing the carried state $m^{\ell-1}$ at evaluation returns the default checkpoint to in-layer accuracy (68.01 vs. 68.03 for the in-layer checkpoint of the same seed).

Arch.	Init	# Trainable	final α	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
CoFFiE$_{64}$	$\alpha_0{=}0$	6.29M	-0.03	67.92	80.52	71.70	46.31	64.33	81.06	61.86	68.80	67.81
CoFFiE$_{64}$	$\alpha_0{=}{+}0.5$	6.29M	$+0.07$	66.79	80.41	72.26	47.89	64.56	**81.78**	60.67	70.20	68.07
CoFFiE$_{64}$	$\alpha_0{=}{-}0.5$	6.29M	-0.76	68.59	**81.50**	73.69	**49.47**	68.51	80.98	62.29	71.00	69.50
CoFFiE$_{64}$	$\alpha_0{=}{-}1.0$	6.29M	-0.91	**69.63**	81.12	**74.36**	48.11	**68.67**	81.14	**62.46**	**71.20**	**69.59**
CoFFiE$_{88}$	$\alpha_0{=}{-}0.5$	8.65M	-0.77	69.02	81.66	75.54	**49.42**	68.03	**81.23**	**64.51**	71.00	70.05
CoFFiE$_{88}$	$\alpha_0{=}{-}0.5$, α-lr 10^{-3}	8.65M	-0.98	**69.39**	**82.75**	**76.10**	47.62	**70.32**	80.93	62.97	**71.60**	**70.21**

Table 14: **Initialisation of the cross-layer coefficient** (cross-layer, n=3, seed 42). α_0 is the initial value of $\tilde{\alpha}$ (so $\alpha = \tanh\alpha_0$ at initialisation); final α is the mean over layers of the trained coefficient. The last row raises the learning rate of α alone by $5\times$; α then travels further but accuracy does not change beyond seed noise.