Specialist Mediators: Causal Localization of Dense Fine-Tuning

Amir Asiaee
Department of Biostatistics
Vanderbilt University Medical Center
Nashville, TN 37203, USA
amir.asiaeetaheri@vumc.org

Abstract

Dense fine-tuning can turn a base language model into a domain specialist, but task accuracy alone does not reveal which internal computations mediate the new behavior. We introduce *specialist mediators*, the internal components through which a dense specialist's lift over its base causally flows. We identify these mediators by patching specialist activations into the base model on the same input and asking which patches recover the specialist's in-domain score while minimally perturbing off-domain behavior. This separates where the specialist behavior is localized, how much of it can be recovered by activation patching, and whether a sparse trainable update can implement the same behavior, three questions that are often conflated. We formalize this with an exact recovery result, a first-order approximation for joint patching, and a conditional bound showing when patching recovery can transfer to localized training. In synthetic transformers and structural causal models with planted mediators, patching recovers most planted mediators even when activation-magnitude baselines are misled by low-output distractors. In Pythia-410M specialists for math (GSM8K), multitask knowledge (MMLU), and biomedical QA (PubMedQA), single-patch localization ranks first on every domain and seed by area under the recovery curve. Five components recover more than 90% of the dense specialist's lift in answer-sequence likelihood on MMLU and PubMedQA, and a Pythia-1.4B PubMedQA specialist shows the same concentration. A comparison of LoRA adapters at exactly matched parameter counts then separates localization from trainability, showing that a few components can carry the specialist's score while adapter performance depends on more than where the adapter is placed.

1 Introduction

Fine-tuning can turn a general language model into a domain specialist, but task accuracy does not reveal which internal computations now carry the domain-specific lift. Did fine-tuning change a few routing components, redistribute the computation broadly, or mainly alter output-side calibration? This distinction matters for interpretability, auditing, and sparse adaptation, because deciding where to edit, monitor, or retrain a specialist requires a causal map of where its extra behavior flows.

We study this as a *model-difference localization* problem. By localization, we mean identifying a small set of internal components, such as attention heads or MLP blocks, that explain a target behavior. We call heads and MLP blocks *components* and reserve *mechanism* for a computational subgraph. Here the target is the *difference* between two trained models, a base f_0 and a dense specialist f_e. We ask which components causally mediate the performance gap $f_e - f_0$.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Our basic intervention is *activation patching*, which runs two forward passes on the same input and replaces selected internal activations in one model with the corresponding activations from another. We use cross-model activation patching (CMAP) [1], in which both passes share an input and the activations come from a fine-tuned model. We patch specialist activations into the base and measure how much of the dense-specialist lift returns while off-domain behavior stays close to base. A good *specialist mediator set* is one through which the in-domain lift causally flows. The definition is operational, so a mediator set is characterized by the effect of this intervention and need not be unique.

This mediator set supports three related but distinct goals. The first is *localization*, finding the components through which the specialist's extra behavior flows. The second is *patching recovery*, checking whether inserting only those specialist activations into the base recovers the specialist's behavior. The third is *localized training*, updating only parameters near the mediator set instead of fine-tuning the whole model. The third goal requires more, since patching produces a chimera, base weights with specialist activations inserted, and a small trainable parameter class must be able to implement the same computation. Our transfer theorem makes this realizability requirement explicit, and our experiments measure all three goals and show that they can come apart.

The theory follows this sequence from localization to recovery to training. We start with the ideal case, in which fine-tuning changed the computations of only a few components and left the rest of the network as it was. The unchanged components then compute the specialist's values from the specialist's inputs, so patching the changed components reproduces the specialist on every input. Dense fine-tuning usually spreads its changes more widely, so a small patch recovers the specialist only approximately. For this case we write the recovery of a patched set as a sum over its components, where each term is the score's sensitivity to that component times the change in its activation, plus a remainder controlled by the curvature of the score. A component's sensitivity depends on what else is patched, since a patch on a downstream component overwrites what an upstream component sends through it. Scores measured one component at a time can therefore count an effect twice, so we use them to rank candidates and measure the recovery of the resulting sets directly. Training raises a different question, because a patch inserts specialist activations at run time while an adapter must produce them through its own parameters. Our transfer bound limits a localized adapter's excess risk over the dense specialist by two gaps, the distance from the patched model's outputs to the specialist's and the distance from the trainable class to the patched model. Localization speaks to the first gap, and the choice of trainable class governs the second.

The experiments follow this progression. Synthetic transformers and structural causal models test localization when the true mediators are known, and patching stays closer to the planted set when activation magnitudes are confounded by distractors with attenuated outputs. Pythia-410M specialists [2] test patching recovery on math word problems (GSM8K [3]), broad multitask knowledge (MMLU [4]), and biomedical question answering (PubMedQA [5]), with a Pythia-1.4B check on PubMedQA. The localized quantity is the teacher-forced negative log-likelihood (NLL) of the answer sequence, the reference answer followed by an end-of-sequence (EOS) token. The results therefore describe fine-tuning-induced changes in answer-sequence likelihood. A localized low-rank adaptation (LoRA) comparison then probes the training step.

Contributions.

1. We define **specialist mediator sets** by interventional recovery and off-domain interference (Section 3).

2. We give theory for the localization–recovery–training sequence, with exact recovery under structural localization, a first-order expansion for joint patching, and a conditional transfer bound that separates patching error from localized-class realizability (Section 4).

3. We evaluate localization and patching recovery on synthetic planted mediators and three Pythia-410M LM domains. Single-patch localization ranks first on every (domain, seed) AURC, and the concentration persists at 1.4B (Sections 5.1–5.3).

4. We compare localized LoRA supports at exactly matched parameter counts and show that localization and trainability are distinct questions (Section 5.4).

Table 1: What is contrasted and what is intervened on. Patching tools share the activation-swap primitive, and this paper uses the cross-model contrast of CMAP.

Setting	Contrast	Intervention target	Question answered
Activation and path patching, causal tracing [7, 9, 19]	clean vs. corrupted input, same weights	activations or paths	Which sites carry an input-level effect?
Skill localization by grafting [20]	base vs. fine-tuned weights	parameter values	Which parameter deltas recover a task?
CMAP [1]	fine-tuned vs. base weights, same input	cross-model activations	Which circuit step explains a fine-tuning gain?
This paper	base vs. dense specialist, same input	cross-model activations	Which components carry the lift, and does adapting them help?

2 Related work

Mechanistic interventions and causal mediation. Interchange interventions, activation patching, and path patching are standard primitives for localizing computations in neural networks [6–10]. A related causal-mediation line treats internal components as mediators between inputs and model behavior, for example in analyses of gender bias, syntactic agreement, and arithmetic reasoning in language models [11–13]. Sparse intervention methods also search for small sets of states whose removal or replacement changes a behavior [14]. Attribution patching and AtP* approximate patch effects with gradients to scale circuit search [15–17]. Certified interventional fidelity adds anytime-valid confidence sequences for such estimates [18]. These tools hold the weights fixed and contrast inputs. Causal tracing restores clean activations into a run on a corrupted input [7], path patching restricts the same swap to chosen paths [9], and attribution patching linearizes it [15]. Cross-model activation patching (CMAP) [1] instead fixes the input and takes activations from a fine-tuned model, and it attributed a fine-tuning gain on entity tracking to a step of an existing circuit. We use the same operator (Table 1) and add a recovery-and-interference selection objective over all candidate components, a conditional analysis of when patching transfers to localized training, and an empirical comparison of the two.

Skill localization and model differences. The closest work in problem formulation asks where newly acquired skills live in a fine-tuned model by *grafting* fine-tuned parameter values into the base and measuring recovered task performance [20]. That work localizes sparse parameter deltas, whereas we localize activation-level mediators of the performance gap.

Localized representation tuning and PEFT. Adapters, LoRA, and DoRA specialize a frozen or partially frozen backbone through small trainable parameter subspaces [21–23]. LoFiT is especially close in spirit, selecting task-specific attention heads and training representation offsets there [24]. ReFT/LoReFT learns interventions on hidden representations as a parameter-efficient alternative to standard fine-tuning [25]. These methods are adaptation mechanisms. Our method is a selection and diagnostic criterion derived from a dense teacher. It first asks where the dense specialist's extra behavior is mediated, then separately asks whether a chosen localized parameter class can realize the patched computation.

Localize-then-correct, editing, and editability of localization. Causal tracing localizes where factual associations appear to be represented, and rank-one or low-rank editing methods modify weights near those locations [7, 26]. However, prior work shows that localization by causal tracing does not necessarily predict the best layer for knowledge edits [27]. Attribution-guided pruning [28] uses layer-wise relevance propagation within one model to identify the parameters behind an undesired behavior and corrects the behavior by pruning that circuit. Our setting differs in both steps, since we localize a difference between two models and the downstream question is whether a trainable local class can reproduce a patched computation rather than whether deletion removes a behavior. The approaches are complementary. Our setting is broader than one factual edit, but the same lesson applies, since finding a causal site and implementing an update there are distinct claims. Theorem 4.3 makes this separation explicit by isolating the patching gap from the realizability gap of the localized parameter class.

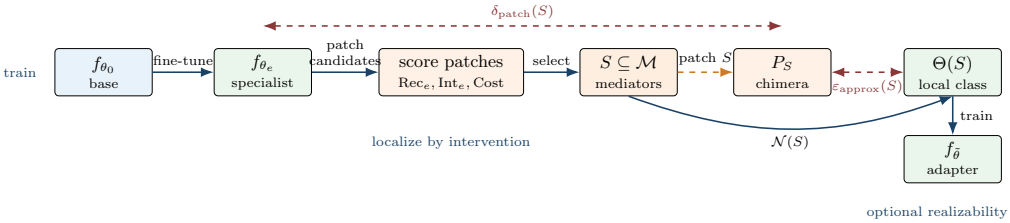

Figure 1: Specialist localization, patching, and realizability. We select mediators S by specialist-to-base patching scores, then separate the patching gap $\delta_{\text{patch}}(S)$ from the realizability gap $\varepsilon_{\text{approx}}(S)$ of the localized class $\Theta(S)$.

A broader survey of causal abstraction, structured pruning, parameter-efficient adaptation, and modularity appears in Appendix E.

3 Problem setup

3.1 Base model, dense specialist, and patching operator

Fix a set of environments $\mathcal{E}$ and one target environment $e \in \mathcal{E}$ with data distribution $\mathcal{D}_e$ and task loss ℓ_e. Let $f_0 := f_{\theta_0}$ be the base generalist and $f_e := f_{\theta_e}$ a *dense specialist teacher*, obtained for example by full fine-tuning or a sufficiently expressive PEFT method on environment e. We write $\mathcal{R}_e(\theta) := \mathbb{E}_{(x,y)\sim\mathcal{D}_e}[\ell_e(f_\theta(x), y)]$ for the risk induced by the loss.

Let $\mathcal{M}$ denote the candidate components to localize over, such as attention heads, MLP blocks, channels, neurons, or learned features. We treat the forward pass as a deterministic structural causal model (SCM) whose endogenous variables are the intermediate activations and whose structural equations are the layer computations [6, 29]. The same view of a trained network as an SCM over its internal activations appears in recent work [30]. For a component $m \in \mathcal{M}$, write $a_m^0(x)$ and $a_m^e(x)$ for the activation produced by the base and specialist on the same input x.

For a set $S \subseteq \mathcal{M}$, define the *specialist-to-base patching operator* $P_S(x) := f_{\theta_0}^{\text{do}(A_S := a_S^e(x))}(x)$, that is, the base model evaluated on x with the activations in S replaced by the specialist's activations on that same input.

3.2 Recovery, interference, and mediator selection

We use score-based quantities for selecting mediators and risk-based quantities for the transfer theorem. Let $\psi_e(f; x, y)$ be a task score where larger is better. For the theoretical results in Section 4 (which require differentiability), we take ψ_e to be a differentiable surrogate such as negative loss or log-probability of the correct answer. In the language-model experiments, ψ_e is the negative teacher-forced NLL of the target sequence (Section 5.2). Define the dense-specialist lift $G_e := \mathbb{E}_{(x,y)\sim\mathcal{D}_e}[\psi_e(f_e; x, y) - \psi_e(f_0; x, y)]$, the quantity we want the mediator set to explain.

The *recovery* of a patched set S is $\text{Rec}_e(S) := \mathbb{E}_{(x,y)\sim\mathcal{D}_e}[\psi_e(P_S; x, y) - \psi_e(f_0; x, y)]$. There are two useful normalizations. The *dense-lift recovery* is $\rho_e^{\text{lift}}(S) := \text{Rec}_e(S)/G_e$, defined whenever $G_e > 0$ (i.e., the dense specialist improves over the base). When $G_e \approx 0$, the specialist provides no meaningful lift and localization is not applicable. The *patchable-gap recovery* is $\rho_e^{\text{patch}}(S) := \text{Rec}_e(S)/\text{Rec}_e(\mathcal{M})$, defined when $\text{Rec}_e(\mathcal{M}) > 0$. This second normalization is useful when the candidate set $\mathcal{M}$ does not include every component that differs between base and specialist. We call $c_e := \text{Rec}_e(\mathcal{M})/G_e$ the *patching ceiling*, the fraction of dense lift recovered by patching all components under consideration. The term denotes this full-set reference value rather than an upper bound over subsets, since recovery need not be monotone in S and either normalization can be negative or exceed one. When $\mathcal{M}$ contains every component that changed, $c_e = 1$ and the two normalizations coincide. In the language-model experiments, embeddings and the unembedding are left unpatched, so recovery curves are normalized by the patchable gap.

To discourage components that indiscriminately change behavior outside the specialty, define (for $|\mathcal{E}| \geq 2$) the off-domain interference $\text{Int}_e(S) := |\mathcal{E} \setminus \{e\}|^{-1} \sum_{e' \neq e} \mathbb{E}_{(x,y)\sim\mathcal{D}_{e'}} \big[|\psi_{e'}(P_S; x, y) -$

$\psi_{e'}(f_0; x, y)|]$, where $\psi_{e'}$ is the score function for environment e'. The absolute value penalizes any off-domain behavioral change, whether helpful or harmful, since the goal is a mediator set that is specific to the target domain.

For the risk bound in Section 4, we also track the output-space patching gap $\delta_{\text{patch}}(S) := \mathbb{E}_{X\sim\mathcal{D}_e}[\|P_S(X) - f_{\theta_e}(X)\|]$. It plays the role in the transfer bound that score recovery plays in the experiments. Outputs are a fixed representation with a specified norm, and for cross-entropy we use vocabulary logits with the Euclidean norm under the teacher-forced convention of Appendix D.

The ideal mediator set solves

$$S_e^\star \in \arg\max_{S\subseteq\mathcal{M}} \big\{\mathrm{Rec}_e(S) - \beta_{\text{int}}\,\mathrm{Int}_e(S) - \lambda\,\mathrm{Cost}(S)\big\}, \tag{1}$$

where $\mathrm{Cost}(S)$ may be cardinality, FLOPs, number of trainable parameters induced by S, or latency overhead.

4 Theory

This section states the steps from localization to training as three results about a set of patched components. Proposition 4.1 treats the ideal case, in which fine-tuning changes only a few components. Proposition 4.2 expands the recovery of any patched set when the changes are more widespread. Theorem 4.3 moves from patching to training and bounds the risk of a localized adapter. All proofs are in Appendix A.

4.1 Exact mediator sufficiency

Suppose fine-tuning changed the computations of only the components in a set T_e and left the rest of the network as it was. A patch at every component in T_e gives each later computation the same inputs it sees in the specialist. Those computations are unchanged, so they produce the specialist's values all the way to the output, and the patched model reproduces the specialist exactly.

Proposition 4.1 (Exact mediator sufficiency)**.** *Suppose the dense specialist f_e is obtained from the base model f_0 by changing only the structural equations of a set of components $T_e \subseteq \mathcal{M}$, while every other node of the computation graph, including input and output computations outside $\mathcal{M}$, is unchanged. If patching is applied at the outputs of every component in T_e, then $P_{T_e}(x) = f_e(x)$ for every input x. Hence $\rho_e^{\text{lift}}(T_e) = 1$ whenever $G_e > 0$, and $\mathrm{Int}_e(T_e)$ equals the specialist's own off-domain score change.*

The result requires patching every changed computation. Our language-model candidate set $\mathcal{M}$ omits the embeddings, final layer norm, and unembedding, so Section 5.2 reports the patching ceiling c_e, the recovery when every component in $\mathcal{M}$ is patched.

4.2 Linearization under joint patching

Dense fine-tuning changes every parameter, so Proposition 4.1 rarely applies exactly. We then want to know how much of the lift a small set recovers and how that recovery divides among its components. A first-order expansion addresses both. Treat the score as a function of the values placed at the patched sites, with everything else recomputed. Moving those values from the base activations to the specialist's changes the score by approximately the gradient times the change, summed over the sites, and the curvature of the score controls the error. Fix (x, y) and a set $S \subseteq \mathcal{M}$ of distinct patch sites, write $\Delta_m(x) := a_m^e(x) - a_m^0(x)$ and $\Delta_S = (\Delta_m)_{m\in S}$, and for replacement values b_S define $F_{S,x,y}(b_S) := \psi_e(f_{\theta_0}^{\mathrm{do}(A_S:=b_S)}; x, y)$. Every site in S is clamped and all other computations are recomputed, so $F_{S,x,y}(a_S^0) = \psi_e(f_0; x, y)$ and $F_{S,x,y}(a_S^e) = \psi_e(P_S; x, y)$.

Proposition 4.2 (Linearized recovery under joint patching)**.** *Suppose $F_{S,x,y}$ is twice continuously differentiable on a neighborhood of the segment $\{a_S^0 + t\Delta_S : 0 \le t \le 1\}$. Let $g_{m,S}(x, y) :=$*

$\nabla_{b_m} F_{S,x,y}(a_S^0(x))$ *and* $H_S(t;x,y) := \nabla^2_{b_S} F_{S,x,y}(a_S^0(x) + t\Delta_S(x))$. *Then*

$$\psi_e(P_S;x,y) - \psi_e(f_0;x,y) = \sum_{m\in S} \langle g_{m,S}(x,y), \Delta_m(x)\rangle + R_S(x,y),$$
$$|R_S(x,y)| \le \tfrac{1}{2} \sup_{t\in[0,1]} \|H_S(t;x,y)\|_{\mathrm{op}} \|\Delta_S(x)\|_2^2.$$

When the summands and the remainder bound are integrable, $\mathrm{Rec}_e(S) = \sum_{m\in S} \alpha_{e,m;S} + \mathbb{E}[R_S(X,Y)]$, *where* $\alpha_{e,m;S} := \mathbb{E}\langle g_{m,S}(X,Y), \Delta_m(X)\rangle$.

The first-order terms show what a single-component score measures. For $S = \{m\}$ the derivative is the usual attribution-patching derivative, which defines the singleton score $\alpha_{e,m} := \alpha_{e,m;\{m\}}$. The derivative for a larger set can differ, because a patch on a downstream site overwrites what earlier sites send through it. If no selected site is an ancestor of another, nothing is overwritten and the joint derivatives at the base agree with the singleton derivatives. Across layers they can differ. Consider a chain $u \to v \to z$ with score $\psi = z$, in which the patch moves u and v by the same amount. Clamping v cuts the only path from u to z, so the joint gradient is $(0, 1)$ and the set recovers the change once. Measured alone, each site has derivative 1, so the singleton sum counts the change twice. We therefore use singleton scores to rank candidates and measure the recovery of the resulting sets directly (Appendix C).

4.3 From mediator patching to localized fine-tuning

A patch inserts specialist activations at run time, while a localized adapter updates only parameters near S and has to produce the same computation through those parameters. A set that patches well therefore leaves a second question open, whether some local update can imitate the patched model. The next result separates the two requirements. Let $\mathcal{N}(S)$ denote the parameter neighborhood adjacent to components in S (for example the incoming/outgoing weights of selected heads or the LoRA modules attached to selected blocks), and let $\Theta(S) \subseteq \{\theta_0 + \Delta\theta : \mathrm{supp}(\Delta\theta) \subseteq \mathcal{N}(S)\}$ be the actual admissible adaptation class. Equality holds for unrestricted local updates, while a LoRA configuration also imposes its module choices and ranks.

Theorem 4.3 (Patching-to-adaptation transfer)**.** *Assume the task loss is L_ℓ-Lipschitz in the model outputs. Suppose there exists a localized adaptation $\tilde\theta \in \Theta(S)$ such that on $\mathcal{D}_e$, $\mathbb{E}[\|f_{\tilde\theta}(X) - P_S(X)\|] \le \varepsilon_{\mathrm{approx}}$. Then*

$$\mathcal{R}_e(\tilde\theta) \le \mathcal{R}_e(\theta_e) + L_\ell\Big(\delta_{\mathrm{patch}}(S) + \varepsilon_{\mathrm{approx}}\Big).$$

In particular, if S patches almost all of the dense specialist's behavior and the local parameter class can realize that patch, then a localized adaptation supported near S achieves risk close to the dense specialist.

The proof applies the triangle inequality through the patched model, and the Lipschitz constant converts output distance into risk. The patching gap $\delta_{\mathrm{patch}}(S)$ depends on the choice of S, and the approximation gap $\varepsilon_{\mathrm{approx}}$ also depends on the trainable parameter class.

Remark 4.4 (Scope of the transfer bound). The theorem is conditional on two properties, that $\Theta(S)$ can approximate the patched model (small $\varepsilon_{\mathrm{approx}}$) and that patching recovers the specialist in output space (small $\delta_{\mathrm{patch}}(S)$). It is stated in output space (logits, for cross-entropy), and we distinguish three quantities. *Score recovery* $\mathrm{Rec}_e(S)$ is measured in E2. *Output-space matching*, $\delta_{\mathrm{patch}}(S)$ and $\varepsilon_{\mathrm{approx}}$, consists of the theorem's hypotheses, which we do not measure here. The *task performance of a trained adapter* is measured in E3 and also reflects estimation and optimization error.

An empirical analogue (Remark A.1 in Appendix A) replaces $\tilde\theta$ by a trained adapter $\hat\theta_S$ under uniform-deviation and empirical-suboptimality conditions, adding an estimation and an optimization term. In Section 5.4 we report the normalized excess task NLL of a trained adapter as an *empirical adapter residual*. It is an empirical counterpart of the left-hand side of this four-term bound rather than an estimate of $\varepsilon_{\mathrm{approx}}$. Proposition B.1 in Appendix B adds a complementary check in parameter space, bounding the risk of keeping only the part of the dense update near S.

5 Experiments

The experiments evaluate three claims. E1 tests whether activation patching recovers planted mediators under confounded activation magnitudes. E2 tests whether the specialist-vs-base gap in language models concentrates in small patchable component sets, at two model scales. E3 compares localized LoRA supports at exactly matched parameter counts. Unless stated otherwise, numbers are mean $\pm$ standard error across training seeds (five in E1, three in E2 and E3). The Pythia-1.4B check is a single seed, for which we report item-level uncertainty.

5.1 E1: Synthetic mediator recovery

Setup. E1 uses small models whose true mediators are known by construction. A 4-layer, 4-head causal transformer reads a sequence of 32 tokens and predicts one of 4 classes, and the base learns to read the class from label tokens at three fixed positions. The dense specialist is a copy of the base in which only 3 planted heads T_e are fine-tuned, on data whose label tokens have moved to three other positions. Head patches act before each head's output projection. The same design is repeated in a 20-node linear SCM, in which a classifier reads $|T_e|$=5 planted nodes (Appendix D). *Regime A* is the planted construction, *Regime B* adds parameter noise to non-planted heads, and *Regime C* copies planted heads into 3 distractor heads with output projections scaled by 0.05, so their activations change strongly while their output effect is attenuated.

Results. We compare greedy, exhaustive, single-patch, singleton gradient-weighted ($\alpha_{e,m}$), magnitude ($\|\Delta_m\|$), and random selection (Appendix F). Table 6 in Appendix F reports Jaccard similarity at the oracle budget $k = |T_e|$, and Figure 3 shows recovery curves. The Regime A linear SCM satisfies the hypotheses of Proposition 4.1, and patching its planted set recovers the full lift ($\rho_e^{\text{lift}} = 1.00$). In the transformer, the fine-tuned output projections of planted heads sit downstream of the patch ($\rho_e^{\text{lift}} = 0.71 \pm 0.02$ for exhaustive search at $k = 3$), so its results test support recovery under this intervention. Magnitude identifies the planted heads exactly in Regimes A and B. The discriminating case is Regime C, whose distractors change their activations strongly while their outputs are attenuated. There magnitude selection loses support recovery (Jaccard 0.48 ± 0.15) while greedy patching remains substantially closer to the planted mediator set (0.80 ± 0.12). Recovery fractions in Regime C are 0.69 ± 0.04 for patching vs. 0.52 ± 0.07 for magnitude. The gap is larger in the SCM, where greedy patching achieves Jaccard 1.00 in every regime while magnitude falls to 0.39 under confounding.

5.2 E2: Localization on language models

Setup. E2 fine-tunes a general-purpose language model to answer questions in one subject area and asks which components carry the change. Pythia [2] is a family of open GPT-style models trained on a broad corpus of English text to predict the next token, with no task or subject built in. Given a prompt, it continues the text one token at a time. We use Pythia-410M, with about 410 million parameters, 24 layers, and 16 attention heads per layer, as the base generalist in E2 and E3. The three domains are question-answering benchmarks. GSM8K [3] contains grade-school math word problems, and the model should write a worked solution ending in the numerical answer. MMLU [4] contains multiple-choice questions in 57 subjects, from mathematics to law, and the model should give the correct option letter. PubMedQA [5] contains biomedical research questions with passages from the article's abstract as context, and the model should answer yes, no, or maybe. For each domain and seed we fine-tune all parameters on 1,024 training examples for three epochs to obtain a dense specialist (Appendix D). PubMedQA training and calibration examples come from its automatically labeled PQA-A subset and test examples from its expert-labeled PQA-L subset, whose label distributions differ, with about 93% of the PQA-A training examples labeled yes against 53–56% of the PQA-L test items. Each domain uses 500 calibration examples (patch scores and selection only), 500 validation examples, and 500 test examples (all reported numbers), drawn separately for each seed.

Scored target. The score ψ_e is the negative NLL of the target sequence, averaged over its tokens. It is teacher-forced, meaning that the model sees the prompt and the preceding reference tokens and is scored on the probability it gives the next one. The target is the reference answer followed by the EOS token, which gives 98 tokens on average for GSM8K and two tokens for MMLU and

Table 2: Area under the recovery curve (AURC) on three Pythia-410M language-model domains. Mean $\pm$ standard error over 3 seeds, with higher better. AURC is computed over $k \in [0,\ 0.3|\mathcal{M}|]$ with the budget axis normalized by $|\mathcal{M}|$, so a method that immediately reaches the patching ceiling has AURC ≈ 0.3. Negative AURC for magnitude on GSM8K means the patch worsens the in-domain score relative to the base model before enough components accumulate.

Method	GSM8K	MMLU	PubMedQA
Single-patch (ours)	$\mathbf{+0.250 \pm 0.003}$	$\mathbf{+0.293 \pm 0.002}$	$\mathbf{+0.299 \pm 0.004}$
Magnitude	-0.118 ± 0.012	$+0.225 \pm 0.018$	$+0.287 \pm 0.002$
Random	$+0.131 \pm 0.006$	$+0.193 \pm 0.007$	$+0.218 \pm 0.034$

PubMedQA. The EOS token is scored like any other target token. The lift G_e thus measures a change in answer-sequence likelihood, which includes answer formatting and stopping behavior.

Specialist quality. At 410 million parameters and 1,024 training examples, neither the base nor the specialist answers these questions reliably, so accuracy separates them only weakly. Fine-tuning does change the probability the model gives the reference answer, including its format and its stopping point, and NLL measures that change. We therefore localize on NLL. Under option-likelihood scoring (the highest-probability allowed label at the answer position), MMLU accuracy is 0.227 ± 0.007 for the base and 0.239 ± 0.028 for the specialist, against a chance level of 0.25. PubMedQA accuracy rises from 0.404 ± 0.009 to 0.541 ± 0.012, matching the majority-label baseline (0.540), with the specialist predicting yes on 99.7% of test items. On GSM8K, greedy decoding with up to 256 new tokens gives 0.020 ± 0.003 for the base and 0.035 ± 0.008 for the specialist. The NLL lift is 1.21, 3.82, and 8.38 nats on GSM8K, MMLU, and PubMedQA, of which the EOS position accounts for 6%, 89%, and 60–63% (Appendix H).

Localization methods and metric. Mediator candidates consist of the $n_{\text{layers}} \times n_{\text{heads}} = 384$ attention heads and the 24 MLP-block outputs, for a total $|\mathcal{M}| = 408$. Heads are patched additively at the per-head output projection, and MLP blocks are patched by full output replacement. The final LM grid compares single-component patching (single-patch), activation magnitude $\|\Delta_m\|$, and uniformly random selection over all 408 components. Single-patch scores each component by the gain in ψ_e when its specialist activation is patched into the base, averaged over the calibration set. Rankings are computed in batched groups with the other group members held at their base activations, so each MLP score is conditional on the other MLP blocks. Exact single-component scores give closely related rankings (Spearman 0.81–0.95) and comparable top-k recovery (Appendix I). Performance is summarized by the area under the recovery curve (AURC) over $k \in [0,\ 0.3\,|\mathcal{M}|]$, which captures how efficiently each method localizes specialty within a small budget.

The patching ceiling. Embeddings and the unembedding stay at base (Figure 4), so patching all of $\mathcal{M}$ need not reproduce the specialist exactly. The ceiling is close to one in every domain (1.004, 0.997, and 1.013 on GSM8K, MMLU, and PubMedQA), and we plot ρ_e^{patch} for LM recovery curves.

Results. Table 2 summarizes AURC across the three domains and three seeds, and single-patch ranks first on every (domain, seed) pair. Interference and component-type breakdowns are in Appendix G. Attention-only patching reaches 0.735 of the GSM8K lift, against $c_e \approx 1.0$ with MLPs. Table 3 gives the curves of Figure 2 in absolute terms, and GSM8K needs sixteen components for 0.84 of 1.21 nats. Under exact single-component scores, MLP blocks fill three to five of the top five places in every domain and seed (Appendix I and Figure 6).

Ranking stability. With the specialist fixed, rankings correlate at 0.978 across disjoint calibration halves (1.4B) and at 0.71–0.99 across prompt templates. Accuracy-based rankings are moderately related to NLL-based ones on PubMedQA and unrelated on MMLU, where accuracy is at chance (Appendix J).

5.3 Scale check: Pythia-1.4B

To test whether the concentration survives a larger backbone while holding the model family and specialist construction fixed, we repeated the PubMedQA protocol with Pythia-1.4B, the member of

Table 3: Absolute recovery of the dense NLL lift by the top-k single-patch components. Pythia-410M, mean $\pm$ s.e. over 3 seeds. Each entry is the NLL recovered by patching the top-k components (nats) and, in parentheses, its fraction of the dense lift G_e. Values above 100% occur because the chimera can score slightly better than the specialist (Section 3). Patching all 408 components recovers 100%, 100%, and 101% of the lift.

Domain	Dense lift G_e	$k = 3$	$k = 5$	$k = 16$
GSM8K	1.207 ± 0.004	0.263 ± 0.020 (22%)	0.411 ± 0.036 (34%)	0.838 ± 0.007 (69%)
MMLU	3.820 ± 0.023	2.821 ± 0.121 (74%)	3.545 ± 0.121 (93%)	3.756 ± 0.036 (98%)
PubMedQA	8.379 ± 0.157	7.700 ± 0.139 (92%)	8.600 ± 0.078 (103%)	8.554 ± 0.083 (102%)

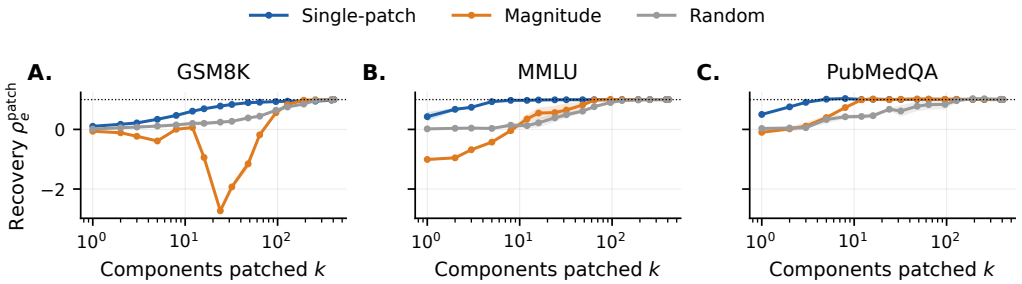

Figure 2: Recovery vs. number of patched components on three LM domains. Curves show patchable-gap recovery ρ_e^{patch}, so all methods reach 1 at $k = |\mathcal{M}|$ (log scale on the horizontal axis). Single-patch reaches half of the patchable gap at $k \approx 8$ (2% of $\mathcal{M}$) on GSM8K and with one or two components on MMLU and PubMedQA, and exceeds 0.9 at $k = 5$ on both. Bands are ± 1 s.e. over 3 seeds.

the same family with about 1.4 billion parameters (24 layers, 16 heads, so again $|\mathcal{M}| = 408$), using dense fine-tuning on the same 1,024 training examples for three epochs, 256 calibration examples, and the same 500 test examples (seed 0). On the test set, NLL falls from 6.767 to 1.458, a paired lift of 5.309 nats (item-level s.e. 0.065, 95% paired bootstrap interval $[5.201, 5.436]$). Table 4 shows that single-patch recovers 76% of the lift with three components and 94% with five, while magnitude selection moves the model away from the specialist and random selection recovers almost nothing. The five top-ranked components are all MLP blocks. With exact single-component scores the top five are again all MLP blocks and recover 88% (Appendix I). As at 410M, the EOS position carries 55% of the lift (Appendix H).

5.4 E3: Localized adaptation at matched parameter budgets

E3 moves from patching to training. Instead of fine-tuning the whole model, we train a small adapter on a few blocks of Pythia-410M with the same domain data, and ask whether blocks chosen from the patching ranking work better than other blocks at the same parameter count.

Setup. LoRA [22] freezes the model's weights and trains a small low-rank update on selected modules. An adapter is set by its support, the k of $N{=}24$ transformer blocks it may update, and by the rank r of the update. Every adapter updates the attention and MLP projections of its blocks, and the five methods differ only in the support. *Patching-localized* takes the k blocks that contain the highest-ranked heads in the attention-head entries of the E2 single-patch ranking. *Random-localized* takes k blocks at random. *Dense-update-norm* takes the k blocks whose weights moved most under dense fine-tuning ($\|\theta_e - \theta_0\|$). *Domain-probe* is a training-free selector that ranks blocks by activation separation between in-domain and off-domain batches [inspired by 24]. *Full LoRA* adapts all N blocks at the scaled rank r' that gives the same budget. Parameter budgets match exactly, so in each of the nine $(k/N, r)$ cells all 45 adapters (five methods, three domains, three seeds) have identical trainable-parameter counts, from 393,216 at $(0.25, 4)$ to 4,718,592 at $(0.75, 16)$. We train on 256 examples for 5 epochs over $k/N{\in}\{0.25, 0.5, 0.75\}$ and $r{\in}\{4, 8, 16\}$, and compare supports by test NLL.

Results. Full LoRA has the lowest grid-averaged test NLL on GSM8K (Table 5), below patching-localized in each seed (1.227 vs. 1.310). On MMLU and PubMedQA, patching-localized has the

Table 4: Pythia-1.4B PubMedQA, one seed. NLL recovered (nats) and fraction of the 5.309-nat dense lift for the top-k components of each ranking, on 500 test items. Patching all 408 components recovers 5.322 nats (100.2%).

Ranking	$k = 3$	$k = 5$	$k = 16$
Single-patch	4.030 (75.9%)	4.975 (93.7%)	5.700 (107.4%)
Magnitude	−4.226 (−79.6%)	−3.774 (−71.1%)	−0.872 (−16.4%)
Random	0.188 (3.5%)	0.209 (3.9%)	2.343 (44.1%)

Table 5: Localized adapters at matched parameter budgets. *Grid NLL* is in-domain test NLL averaged over the nine $(k/N, r)$ cells within each training seed (mean ± s.e. across 3 seeds, lowest in bold). *Residual* is the empirical adapter residual at the method's lowest-NLL cell, in percent. Dense specialist NLL is 1.051, 0.714, and 1.350, and base NLL is 2.173, 4.527, and 9.726.

	GSM8K		MMLU		PubMedQA	
Method	Grid NLL	Residual	Grid NLL	Residual	Grid NLL	Residual
Patching-localized	1.3098 ± 0.0071	12.8	$\mathbf{0.7268 \pm 0.0079}$	0.06	$\mathbf{1.178 \pm 0.146}$	−3.4
Full LoRA	$\mathbf{1.2272 \pm 0.0019}$	11.7	0.7271 ± 0.0087	0.02	1.296 ± 0.191	−1.8
Random-localized	1.2776 ± 0.0135	12.8	0.7298 ± 0.0083	−0.04	1.192 ± 0.146	−3.7
Dense-update-norm	1.2777 ± 0.0044	11.8	0.7390 ± 0.0080	0.19	1.226 ± 0.148	−2.6
Domain-probe	1.2985 ± 0.0016	13.2	0.7325 ± 0.0081	0.05	1.222 ± 0.177	−3.1

lowest mean and is below random-localized in each seed, by margins smaller than the across-seed variation. Table 5 also reports each method at its own lowest-NLL cell as an *empirical adapter residual*, the fraction of the dense NLL gain that the adapter leaves unrecovered, $(\mathrm{NLL}_{\text{adapter}} - \mathrm{NLL}_{\text{dense}})/(\mathrm{NLL}_{\text{base}} - \mathrm{NLL}_{\text{dense}})$. On GSM8K every method leaves 11.7–13.2% of the gain, and every method's best cell is the largest budget in the grid, although patching all 408 components reproduces the specialist's GSM8K score ($c_e = 1.004$). On MMLU every residual is below 0.2%, and on PubMedQA every method's best cell reaches lower test NLL than the dense specialist. The similar GSM8K residuals are compatible with a limitation shared by block-level LoRA under this training budget, and separating its sources is future work. Block selection uses head scores while MLP blocks are among the highest-ranked components in E2, and LoRA adapts whole blocks while E2 ranks individual components, which leaves room for supports designed around the full ranking. Appendix K adds block-overlap, tiny-budget, and head-level adapter comparisons.

6 Discussion and Conclusion

Our central object is the specialist mediator, the internal components through which dense fine-tuning changes the model's in-domain score. The results separate three levels of evidence. *Score recovery*, whether patching a few components recovers the specialist's task score, is what E2 measures. *Output-space matching*, whether the patched model's outputs match the specialist's (δ_{patch}) and whether a local class can match the patched model ($\varepsilon_{\text{approx}}$), is the level at which Theorem 4.3 is stated. *Trained-adapter performance*, whether an adapter on the localized support reaches the specialist's task risk, is what E3 measures. Successful activation localization does not by itself yield a better localized adapter.

This separation gives a scoped order of checks for anyone who wants to use localization before sparse adaptation. A low full-set reference signals a mismatch worth investigating, possibly a candidate set that omits changed components, though a smaller set can still patch well. If a small set patches well but adapters on its support fit poorly, the open questions concern the adapter's capacity, module support, and training signal. If both succeed, the set is a candidate sparse-adaptation target.

Limitations and outlook. The language-model results cover one model family at two scales, with one seed and one domain at 1.4B. Natural next steps are other model families and PEFT-trained specialists at larger scale, answer-only scores that exclude EOS, and per-token $\mathrm{KL}(f_e \,\|\, P_S)$ and $\mathrm{KL}(P_S \,\|\, f_{\hat{\theta}_S})$ with adapters distilled to the patched model, which would test agreement with the patch more directly. Cheaper specialization also lowers the barrier for harmful specialists with subtler weight drift, and localization itself can serve as a detection diagnostic.

Acknowledgments and Disclosure of Funding

This work received no third-party funding. The author declares no competing interests.

References

[1] Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *International Conference on Learning Representations*, 2024.

[2] 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 *Proceedings of the 40th International Conference on Machine Learning*, pages 2397–2430, 2023.

[3] 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.

[4] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

[5] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, 2019.

[6] Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 9574–9586, 2021.

[7] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[8] Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[9] Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

[10] Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D. Goodman. Finding alignments between interpretable causal variables and distributed neural representations. In *Proceedings of the Third Conference on Causal Learning and Reasoning*, pages 160–187, 2024.

[11] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.

[12] Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart M Shieber, Tal Linzen, and Yonatan Belinkov. Causal analysis of syntactic agreement mechanisms in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1828–1843, 2021.

[13] Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7035–7052, 2023.

[14] Nicola De Cao, Leon Schmid, Dieuwke Hupkes, and Ivan Titov. Sparse interventions in language models with differentiable masking. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 16–27, 2022.

[15] Neel Nanda. Attribution patching: Activation patching at industrial scale. https://www.neelnanda.io/mechanistic-interpretability/attribution-patching, 2023. Blog post.

[16] Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, 2024.

[17] János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. AtP*: An efficient and scalable method for localizing LLM behaviour to components. *arXiv preprint arXiv:2403.00745*, 2024.

[18] Amir Asiaee. Certified interventional fidelity: Anytime-valid, adaptive evaluation of causal claims in mechanistic interpretability. In *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, pages 210–227, 2026.

[19] Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, 2024.

[20] Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[21] Neil Houlsby, Andrei Giurgiu, Stanisław Jastrzębski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2790–2799, 2019.

[22] 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.

[23] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32100–32121, 2024.

[24] Fangcong Yin, Xi Ye, and Greg Durrett. LoFiT: Localized fine-tuning on LLM representations. In *Advances in Neural Information Processing Systems*, 2024.

[25] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. ReFT: Representation finetuning for language models. In *Advances in Neural Information Processing Systems*, 2024.

[26] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations (ICLR)*, 2023.

[27] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. In *Advances in Neural Information Processing Systems*, 2023.

[28] Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Reduan Achtibat, Patrick Kahardipraja, Thomas Wiegand, Wojciech Samek, Alexander Binder, and Sebastian Lapuschkin. Attribution-guided pruning for insight and control: Circuit discovery and targeted correction in small-scale LLMs. *arXiv preprint arXiv:2506.13727*, 2025.

[29] Sander Beckers and Joseph Y. Halpern. Abstracting causal models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2678–2685, 2019.

[30] Amir Asiaee. Causal mechanism reduction: Mechanism replacement for neural network pruning and abstraction. *arXiv preprint arXiv:2602.24266*, 2026.

[31] Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[32] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision*, 2018.

[33] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[34] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[35] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11893–11902, 2020.

[36] Amr Hendy, Mohamed Abdelghaffar, Mohamed Afify, and Ahmed Y. Tawfik. Domain specific sub-network for multi-domain neural machine translation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 351–356, 2022.

[37] Jianze Liang, Chengqi Zhao, Mingxuan Wang, Xipeng Qiu, and Lei Li. Finding sparse structures for domain specific neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[38] Nan Zhang, Yanchi Liu, Xujiang Zhao, Wei Cheng, Runxue Bao, Rui Zhang, Prasenjit Mitra, and Haifeng Chen. Pruning as a domain-specific LLM extractor. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1417–1428, 2024.

[39] Lei Lu, Zhepeng Wang, Runxue Bao, Mengbing Wang, Fangyi Li, Yawen Wu, Weiwen Jiang, Jie Xu, Yanzhi Wang, and Shangqian Gao. All-in-one tuning and structural pruning for domain-specific LLMs. *arXiv preprint arXiv:2412.14426*, 2024.

[40] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, 1989.

[41] Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5, 1992.

[42] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations*, 2016.

[43] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient ConvNets. In *International Conference on Learning Representations*, 2017.

[44] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[45] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[46] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through L_0 regularization. In *International Conference on Learning Representations*, 2018.

[47] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[48] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[49] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[50] Utku Evci, Nicolas Le Roux, Pablo Castro, and Léon Bottou. Mean replacement pruning. OpenReview submission to ICLR 2019, 2018.

[51] Woojeong Kim, Suhyun Kim, Mincheol Park, and Geonseok Jeon. Neuron merging: Compensating for pruned neurons. In *Advances in Neural Information Processing Systems*, volume 33, pages 585–595, 2020.

[52] Uranik Berisha, Jens Mehnert, and Alexandru Paul Condurache. Variance-based pruning for accelerating and compressing trained networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4973–4982, 2025.

[53] Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[54] Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[55] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024.

[56] Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[57] Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, 2025.

[58] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003, 2016.

[59] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[60] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1930–1939, 2018.

[61] 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.

[62] 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.

[63] Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6172, 2024.

[64] Enshu Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B. Blaschko, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*, 2024.

[65] Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti. Modular deep learning. *Transactions on Machine Learning Research*, 2023.

[66] Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *International Conference on Learning Representations*, 2021.

[67] Gabriel Béna and Dan F. M. Goodman. Dynamics of specialization in neural modules under resource constraints. *Nature Communications*, 16:187, 2025.

[68] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, 2021.

[69] Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.

[70] Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.

[71] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.

[72] 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*, 2023.

[73] 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 *Proceedings of the 41st International Conference on Machine Learning*, pages 57755–57775, 2024.

[74] MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[75] Leonardo Iurada, Marco Ciccone, and Tatiana Tommasi. Efficient model editing with task-localized sparse fine-tuning. In *International Conference on Learning Representations*, 2025.

[76] Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 10121–10143, 2025.

[77] Ronald Skorobogat, Karsten Roth, and Mariana-Iuliana Georgescu. Subspace-boosted model merging. *arXiv preprint arXiv:2506.16506*, 2025.

[78] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in LLMs, MLLMs, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58(8):1–41, 2026.

[79] Paul K. Rubenstein, Sebastian Weichwald, Stephan Bongers, Joris M. Mooij, Dominik Janzing, Moritz Grosse-Wentrup, and Bernhard Schölkopf. Causal consistency of structural equation models. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

[80] Sander Beckers, Frederick Eberhardt, and Joseph Y. Halpern. Approximate causal abstractions. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*, pages 606–615, 2020.

[81] Riccardo Massidda, Atticus Geiger, Thomas Icard, and Davide Bacciu. Causal abstraction with soft interventions. In *Proceedings of the Second Conference on Causal Learning and Reasoning*, 2023.

[82] Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah D. Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64, 2025.

[83] Satchel Grant. Model alignment search. *arXiv preprint arXiv:2501.06164*, 2025.

[84] Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing, a method for rigorously testing interpretability hypotheses. AI Alignment Forum, 2022. Online essay.

[85] Claudia Shi, Nicolas Beltran-Velez, Achille Nazaret, Carolina Zheng, Adrià Garriga-Alonso, Andrew Jesson, Maggie Makar, and David M. Blei. Hypothesis testing the circuit hypothesis in LLMs. *Advances in Neural Information Processing Systems*, 37:94539–94567, 2024.

A Full proofs and the four-term empirical bound

This appendix proves the results of Section 4. We first give the empirical version of the transfer bound mentioned in Section 4, which replaces the comparator of Theorem 4.3 by a trained adapter and so adds an estimation term and an optimization term.

Remark A.1 (Empirical learning decomposition). Let $\widehat{\mathcal{R}}_{e,n}$ be empirical task risk, let $\hat{\theta}_S \in \Theta(S)$ be a trained adapter, and let $\tilde{\theta} \in \Theta(S)$ be a comparator satisfying Theorem 4.3. Suppose, on an event of interest,

$$\sup_{\theta\in\Theta(S)} |\mathcal{R}_e(\theta) - \widehat{\mathcal{R}}_{e,n}(\theta)| \le \eta_S, \qquad \widehat{\mathcal{R}}_{e,n}(\hat{\theta}_S) \le \inf_{\theta\in\Theta(S)} \widehat{\mathcal{R}}_{e,n}(\theta) + \varepsilon_{\mathrm{opt}}.$$

On that event,

$$\mathcal{R}_e(\hat{\theta}_S) - \mathcal{R}_e(\theta_e) \le L_\ell\big(\delta_{\mathrm{patch}}(S) + \varepsilon_{\mathrm{approx}}\big) + 2\eta_S + \varepsilon_{\mathrm{opt}}.$$

Indeed, $\mathcal{R}_e(\hat{\theta}_S) \le \widehat{\mathcal{R}}_{e,n}(\hat{\theta}_S) + \eta_S \le \widehat{\mathcal{R}}_{e,n}(\tilde{\theta}) + \varepsilon_{\mathrm{opt}} + \eta_S \le \mathcal{R}_e(\tilde{\theta}) + \varepsilon_{\mathrm{opt}} + 2\eta_S$, and Theorem 4.3 bounds the last risk. This is the four-term decomposition with estimation term $\varepsilon_{\mathrm{est}} = 2\eta_S$. If the class is selected using the same data, the deviation event must hold simultaneously over the possible selected classes.

Proof of Proposition 4.1 (exact mediator sufficiency)

Let $\mathcal{V} = \{m_1, \dots, m_M\} \supseteq \mathcal{M}$ be the set of all nodes of the finite computation graph, ordered so that m_i is an ancestor of m_j only if $i < j$ (topological order). For any node m_i, write $\mathrm{pa}(m_i)$ for its parent set and let g_i^0 and g_i^e denote the structural equations of m_i in the base model and specialist respectively. By hypothesis, $g_i^0 = g_i^e$ for all $m_i \notin T_e$.

We prove by induction on i that the activation of the patched model P_{T_e} equals that of the specialist at every node. *Base case.* Consider m_1. If $m_1 \in T_e$, the patching operator replaces $a^0_{m_1}(x)$ with $a^e_{m_1}(x)$, giving equality. If $m_1 \notin T_e$, then $g_1^0 = g_1^e$ and m_1 has no parents, so $a^0_{m_1}(x) = g_1^0(x) = g_1^e(x) = a^e_{m_1}(x)$. *Inductive step.* Suppose for all $j < i$ that the patched model's activation at m_j equals $a^e_{m_j}(x)$. If $m_i \in T_e$, patching forces equality. If $m_i \notin T_e$, $g_i^0 = g_i^e$ and parent activations agree by induction, so $a^{P_{T_e}}_{m_i}(x) = g_i^0(a^e_{\mathrm{pa}(m_i)}(x)) = g_i^e(a^e_{\mathrm{pa}(m_i)}(x)) = a^e_{m_i}(x)$. The model output is itself a node of $\mathcal{V}$, so $P_{T_e}(x) = f_e(x)$ for every x, and the ρ_e^{lift} and Int_e claims follow by substitution. □

Proof of Proposition 4.2 (linearized recovery under joint patching)

Fix (x, y) and S, and put $h(t) = F_{S,x,y}(a_S^0 + t\Delta_S)$. Clamping at the base values reproduces the base forward pass, so $h(0) = \psi_e(f_0; x, y)$, and clamping at the specialist values gives $h(1) = \psi_e(P_S; x, y)$. The chain rule gives $h'(0) = \sum_{m\in S}\langle g_{m,S}, \Delta_m\rangle$ and $h''(t) = \Delta_S^\top H_S(t; x, y)\Delta_S$. Taylor's formula with integral remainder gives

$$h(1) - h(0) = h'(0) + \int_0^1 (1-t)\,\Delta_S^\top H_S(t; x, y)\Delta_S\, dt.$$

The operator-norm inequality and $\int_0^1 (1-t)\,dt = \frac{1}{2}$ give the stated bound, and integrability permits taking expectations. For a singleton there are no other clamps. If selected sites have no ancestor relations, clamping another selected site at its base value removes no directed path from m to the output and changes no base value, so the derivative at the base is the singleton derivative. □

Proof of Theorem 4.3 (patching-to-adaptation transfer)

Write $\ell_e(f(x), y)$ for the per-example loss, so $\mathcal{R}_e(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_e}[\ell_e(f_\theta(x), y)]$. By L_ℓ-Lipschitzness of ℓ_e in its first argument, $|\ell_e(f_{\tilde{\theta}}(x), y) - \ell_e(f_{\theta_e}(x), y)| \le L_\ell \|f_{\tilde{\theta}}(x) - f_{\theta_e}(x)\|$. By the triangle inequality, $\|f_{\tilde{\theta}}(x) - f_{\theta_e}(x)\| \le \|f_{\tilde{\theta}}(x) - P_S(x)\| + \|P_S(x) - f_{\theta_e}(x)\|$. Taking

expectations,

$$\begin{aligned}\mathcal{R}_e(\tilde{\theta}) - \mathcal{R}_e(\theta_e) &\le L_\ell \, \mathbb{E}\big[\|f_{\tilde{\theta}}(X) - f_{\theta_e}(X)\|\big] \\ &\le L_\ell \Big(\mathbb{E}\big[\|f_{\tilde{\theta}}(X) - P_S(X)\|\big] + \mathbb{E}\big[\|P_S(X) - f_{\theta_e}(X)\|\big]\Big) \\ &\le L_\ell \big(\varepsilon_{\text{approx}} + \delta_{\text{patch}}(S)\big),\end{aligned}$$

where the last step uses the hypothesis on $\tilde{\theta}$. □

For token-level cross-entropy with model outputs taken as logits, the Lipschitz condition holds with $L_\ell = \sqrt{2}$ in the Euclidean norm, since the gradient of the loss with respect to the logits is the difference between the predicted distribution and a one-hot vector.

B A parameter-space bound

Theorem 4.3 reasons through the patched model. A simpler question asks what happens if we keep the part of the dense update near S and discard the rest. When the risk is smooth around the dense specialist, the cost depends on the size of the discarded part, so it is small when dense fine-tuning moved little outside $\mathcal{N}(S)$.

Proposition B.1 (Restricted-update bound)**.** *Assume the population risk $\mathcal{R}_e$ is β_e-smooth in parameters. Let $\Delta\theta^{\text{full}} := \theta_e - \theta_0$ be the dense specialist update, decomposed as $\Delta\theta^{\text{full}} = \Delta\theta_S + \Delta\theta_{\bar{S}}$, where $\Delta\theta_S$ is supported on $\mathcal{N}(S)$ and $\Delta\theta_{\bar{S}}$ on its complement. Then*

$$\mathcal{R}_e(\theta_0 + \Delta\theta_S) - \mathcal{R}_e(\theta_e) \le \langle \nabla\mathcal{R}_e(\theta_e), -\Delta\theta_{\bar{S}} \rangle + \frac{\beta_e}{2}\|\Delta\theta_{\bar{S}}\|_2^2.$$

If $\|\nabla\mathcal{R}_e(\theta_e)\|_2 \le \gamma_e$, the right-hand side is at most $\gamma_e\|\Delta\theta_{\bar{S}}\|_2 + \frac{\beta_e}{2}\|\Delta\theta_{\bar{S}}\|_2^2$.

The truncated update keeps the dense update's values, so it need not belong to a low-rank adapter class.

Proof of Proposition B.1 (restricted-update bound)

Since $\theta_0 + \Delta\theta_S = \theta_e - \Delta\theta_{\bar{S}}$, β_e-smoothness of $\mathcal{R}_e$ gives the quadratic upper bound $\mathcal{R}_e(\theta_e - \Delta\theta_{\bar{S}}) \le \mathcal{R}_e(\theta_e) + \langle \nabla\mathcal{R}_e(\theta_e), -\Delta\theta_{\bar{S}} \rangle + \frac{\beta_e}{2}\|\Delta\theta_{\bar{S}}\|_2^2$. Cauchy–Schwarz gives the quantified form. □

C Remainder bound for the linearized decomposition

The remainder in Proposition 4.2 measures how far the first-order sum is from the recovery of the set, and blocks of the Hessian show where that error comes from. With block norms $h_{mn} := \sup_{t\in[0,1]} \|[H_S(t)]_{mn}\|_{\text{op}}$,

$$|R_S| \le \frac{1}{2} \sum_{m,n\in S} h_{mn}\|\Delta_m\|_2\|\Delta_n\|_2.$$

The off-diagonal blocks measure how much two patched components interact, and the diagonal blocks measure the curvature of each component's own effect. The linear approximation is useful when both are small at the size of the activation change.

Remark C.1 (Selection heuristics and attribution patching). The singleton coefficients $\alpha_{e,m}$ are the familiar attribution-patching scores [15–17]. A top-k guarantee for recovery would require uniform control of the departure of Rec_e from a set-independent additive score. The coefficients are structurally analogous to curvature-aware importance scores for structured pruning, which expand a replacement risk around a hard intervention (replacing a unit by a constant) rather than around an activation swap [30]. We apply this standard expansion to the specialist-vs-generalist recovery problem and connect it to localized adaptation via Theorem 4.3.

D Detailed experimental setup

This appendix gives the details needed to reproduce the experiments of Section 5.

Synthetic transformer (E1). The architecture is a 4-layer causal transformer with 4 heads per layer, $d_{\text{model}} = 128$, $d_{\text{head}} = 32$, $d_{\text{ff}} = 512$, vocabulary size 64, sequence length 32, 4 classes. Each sequence starts with a classification token, and its other tokens are random except at six signal positions. In the base data, positions 1–3 hold the label's token with probability 0.8 (otherwise a wrong class token) and positions 4–6 are random. In the specialist data, positions 1–3 hold the label's token with probability 0.2 and positions 4–6 with probability 0.9. The base trains on all parameters for 250 steps, and the specialist is a copy in which the query, key, value, and output projections of 3 planted heads train for 220 steps. Each head's cached and patched activation is taken before its output projection. In Regime C, copying and attenuating happen after specialist training, and the distractors keep a small effect on the output.

Linear SCM (E1). A deterministic lower-triangular linear model over $d = 20$ nodes reads six Gaussian input features, two of which carry the label signal. Base input weights are drawn from $\mathcal{N}(0, 0.08^2)$ and node-to-node weights from $\mathcal{N}(0, 0.03^2)$, and the classifier reads 5 planted mediator nodes. The specialist adds to the planted nodes' input weights and to their weight on the preceding node. Regime B adds noise to the structural equations of non-planted nodes. Regime C makes 3 distractor nodes depend strongly on planted nodes, with zero classifier weight.

Output representation for the transfer bound. For token-level cross-entropy, Theorem 4.3 takes outputs to be vocabulary logits with the Euclidean norm. In the teacher-forced application, draw an example and then a scored target position uniformly within it. Here X is the prompt and reference prefix and Y the next target token. The expected token loss then equals the expected per-example mean target-token NLL, and all three models are evaluated on the same context. We use logits because probability-space NLL has no global Lipschitz constant. The reported GSM8K values follow slightly different token weighting, per batch in E2 and over all test tokens in E3. The two-token targets of MMLU and PubMedQA are unaffected.

Language model fine-tuning (E2–E3). The base model is Pythia-410M [2] (24 layers, 16 attention heads, d_{model}=1024). Dense specialists are obtained by full fine-tuning for 3 epochs on 1,024 training examples with AdamW (learning rate 2×10^{-5}, weight decay 0.01), a cosine schedule with linear warm-up over the first 10% of the 384 updates, per-device batch size 2, and 4 gradient-accumulation steps (effective batch size 8). Training examples are the first 1,024 of a seed-specific shuffle of the GSM8K training split (after its calibration and validation examples), the MMLU auxiliary training split, and the PubMedQA PQA-A (artificial) subset (after its calibration examples). Calibration examples (500) come from the GSM8K training split, the MMLU validation split, and PQA-A. Validation examples (500) come from the GSM8K training split, the MMLU validation split, and PQA-L. Test examples (500) come from the GSM8K test split, the MMLU test split, and PQA-L. Inputs are truncated from the left to 256 tokens so that the answer cue is kept. Patching, recovery curves, and E3 evaluation run in 16-bit floating point (float16).

Prompt templates. GSM8K: "Solve the grade-school math problem.\nQuestion: {question}\nAnswer:", target the reference solution, which ends in "#### {answer}". PubMedQA: "Read the biomedical abstract and answer yes, no, or maybe.\nQuestion: {question}\nContext: {first two context passages}\nAnswer:", target the label. MMLU: "Answer the multiple-choice question by giving the correct option letter.\nSubject: {subject}\nQuestion: {question}\nChoices:\nA. . . . \nAnswer:", target the option letter. Each target is preceded by a space and followed by EOS.

Accuracy evaluation. MMLU and PubMedQA accuracy uses option-likelihood scoring, in which the prediction is the allowed label (A–D, or yes/no/maybe) with the highest next-token probability after the answer cue, with prompts left-truncated to their final 256 tokens. GSM8K accuracy uses greedy decoding for up to 256 new tokens, and the prediction is the number after "####" or, if absent, the last number in the completion. The specialist ends 97% of its completions with "####", the base none.

Localized LoRA (E3). For a set of selected blocks, LoRA modules are attached to the fused query-key-value projection, the attention output projection, and both MLP projections of each block, so each adapted block contributes $16{,}384\,r$ trainable parameters. Training uses AdamW with learning rate 10^{-4} and weight decay 0.01, a cosine schedule with 10% warm-up, 5 epochs, batch size 8, and 4 gradient-accumulation steps on the first 256 of the specialist's 1,024 training examples. The

tiny-budget runs in Appendix K use all 1,024. Full LoRA adapts all 24 blocks at the rank $r' = r\,k/N$ that matches the localized budget exactly. The domain-probe selector uses the mean activation separation between in-domain and off-domain calibration batches, a coarser, training-free variant inspired by LoFiT [24].

Pythia-1.4B scale check. Pythia-1.4B (d_{model}=2048, 24 layers, 16 heads) is fine-tuned densely in bfloat16 with gradient checkpointing on the same 1,024 PubMedQA examples for three epochs (learning rate 2×10^{-5}), with seed-0 splits. Single-patch scores use 256 calibration examples, and recovery uses the 500 test examples. Training took 5 minutes and single-patch scoring 15 minutes on one 24 GB GPU.

E Extended related work: causal abstraction, pruning, and modularity

This appendix surveys work further from our setting, including task- and domain-specific subnetworks, structured pruning, learned sharing and modularity, parameter-efficient adaptation, and causal abstraction. Each of these either constructs a compact or modular model or evaluates a single model, whereas our object is the difference between a base model and its dense specialist.

Task-specific subnetworks and parameter isolation. A direct route to specialization is to store multiple subnetworks inside one model. PackNet, Piggyback, and SupSup allocate task-specific masks or parameters via iterative pruning or learned binary masks, motivated by continual learning and parameter isolation [31–33]. The lottery ticket hypothesis and hidden-subnetwork results show that strong task-specific subnetworks exist even in randomly initialized or untrained weights [34, 35]. Domain-specific subnetwork extraction has been pursued directly in NMT—DoSS uses pruning-derived masks to assign a sub-network to each domain [36], and Prune-Tune learns tiny domain-specific sparse structures during fine-tuning [37]—and at LLM scale by D-Pruner [38] and ATP, which couples tuning with structural pruning in a single stage [39]. These methods demonstrate that specialization can be realized by masks or pruning. They build the specialized subnetwork, whereas we ask which components of an already trained dense specialist carry its lift over the base.

Structured pruning with replacement. Classical pruning methods rank parameters by second-order sensitivity [40, 41]. Modern compression targets *structured* units (filters, channels, attention heads, blocks) because wall-clock gains require dense kernels after pruning rather than irregular sparsity [42–46]. Related criteria include Taylor saliency, head importance, and activation statistics [47–49]. Mean-replacement pruning, neuron merging, and variance-based pruning are directly relevant because they replace deleted units by folded constants or nearby units rather than zeroing them [50–52]. Causal mechanism reduction treats these as special cases of a second-order replacement-risk objective [30]. For LLMs, one-shot Hessian-aware and magnitude-aware methods (OBC, SparseGPT, Wanda) extend post-training pruning to scale [53–55]. LLM-Pruner discovers inter-dependency groups [56] and ShortGPT removes whole transformer layers [57]. Our single-patch score borrows the interventional scoring idea but changes the target, measuring how much of the specialist's lift a component carries rather than how much the model's own loss depends on it.

Learned sharing, modularity, and experts. Multi-task and modular systems learn how to share representations or experts during training. Hard- and soft-sharing architectures such as cross-stitch networks [58], sluice networks for latent multi-task architecture learning [59], and multi-gate mixtures of experts [60] learn task-specific mixing weights or gates over a shared backbone. Mixture-of-experts architectures learn sparse routing over explicitly parameterized experts [61–64], and the modular deep learning paradigm organizes computation into autonomous parameter-efficient modules with conditional routing [65]. Whether trained networks themselves develop functionally modular substructure has been studied empirically. Differentiable weight masks reveal that standard architectures often fail to develop clean task-modules [66], and resource constraints and information-flow timing shape specialization dynamics [67]. We ask the complementary post-hoc question of which components mediate the difference between an already trained base and dense specialist, without training a new modular architecture.

Parameter-efficient adaptation and model merging. Adapters, low-rank updates, and localized interventions specialize on top of a frozen backbone [21–23, 68, 69]. Diff pruning enforces sparsity

on a task-specific parameter diff via a differentiable L_0 penalty [70], and LoFiT first identifies a sparse set of attention heads and then fine-tunes only small offsets there [24]. Model merging combines specialists in parameter space via task vectors, sign-aware merging, sparse masking, and subspace-aware merging [71–77]. See Yang et al. [78] for a survey. These approaches operate primarily in parameter-delta or weight space, whereas we work with activations of individual components and score them by intervention effects on the specialist's lift.

Causal abstraction and mechanistic interventions. Causal abstraction studies when a high-level structural causal model is a faithful simplification of a lower-level one under interventions, with exact and approximate variants and soft-intervention extensions [6, 29, 79–82]. For neural networks, the central interventional primitive is the *interchange intervention*, underlying interchange intervention accuracy, distributed alignment search, and causal similarity methods [10, 83], with closely related localization tools such as causal tracing, path patching, and causal scrubbing [7, 9, 84]. Recent work shows that localization results are sensitive to patching metrics and corruption choices [8, 19, 85]. We use this intervention semantics to localize a model difference rather than to certify a human-readable circuit.

Evaluation and uncertainty. Interventional claims are easiest to overstate when the search process is adaptive, since circuit metrics can move substantially with the ablation method or discrepancy function [19, 85]. Certified interventional fidelity addresses this with confidence intervals and anytime-valid confidence sequences for interventional estimands [18]. This motivates treating interventional evaluation as part of the procedure rather than an afterthought, which is why we report the patching ceiling, absolute recovery, and ranking stability across calibration samples, prompt templates, and scoring metrics alongside normalized recovery curves.

F E1 methods, support recovery, and recovery curves

Localization methods. This appendix gives the methods and full results behind Section 5.1. We compare (a) greedy forward selection (add the component with the largest marginal recovery gain); (b) exhaustive patching search over all size-k subsets, reported up to $k = |T_e|$ as an oracle baseline; (c) top-k by individual single-patch recovery; (d) gradient-weighted ranking via the singleton score $\alpha_{e,m}$ (transformer only); (e) magnitude ranking by $\|\Delta_m\|$; and (f) uniform random selection.

Table 6: Support recovery (Jaccard) at $k = |T_e|$. Jaccard similarity between the selected set and the planted set T_e. Transformer model, mean $\pm$ s.e. over 5 seeds. Bold marks the best per regime. In Regimes A–B, magnitude perfectly identifies the planted heads. In Regime C, magnitude drops to 0.48 while greedy patching maintains 0.80.

Method	Regime A	Regime B	Regime C
Greedy patching	0.90 ± 0.10	0.90 ± 0.10	$\mathbf{0.80 \pm 0.12}$
Exhaustive patching	0.90 ± 0.10	0.90 ± 0.10	$\mathbf{0.80 \pm 0.12}$
Gradient-weighted	0.70 ± 0.12	0.80 ± 0.12	0.74 ± 0.17
Single-patch	0.60 ± 0.10	0.80 ± 0.12	0.64 ± 0.16
Magnitude	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.48 ± 0.15
Random	0.08 ± 0.05	0.08 ± 0.05	0.08 ± 0.05

G E2 extras: ceiling, interference frontier, selection heatmap

This appendix collects results behind Section 5.2. Figure 4 shows why patching every component in $\mathcal{M}$ differs from running the specialist. Table 7 breaks the patching ceiling down by component type. Figure 5 compares in-domain recovery with off-domain interference, and Figure 6 shows where the highest-ranked components sit.

H The answer-sequence target and the EOS position

The scored target in Section 5.2 ends in the EOS token, so part of the lift can come from the model learning to stop after its answer. Table 8 splits the dense NLL lift into the contribution of the answer

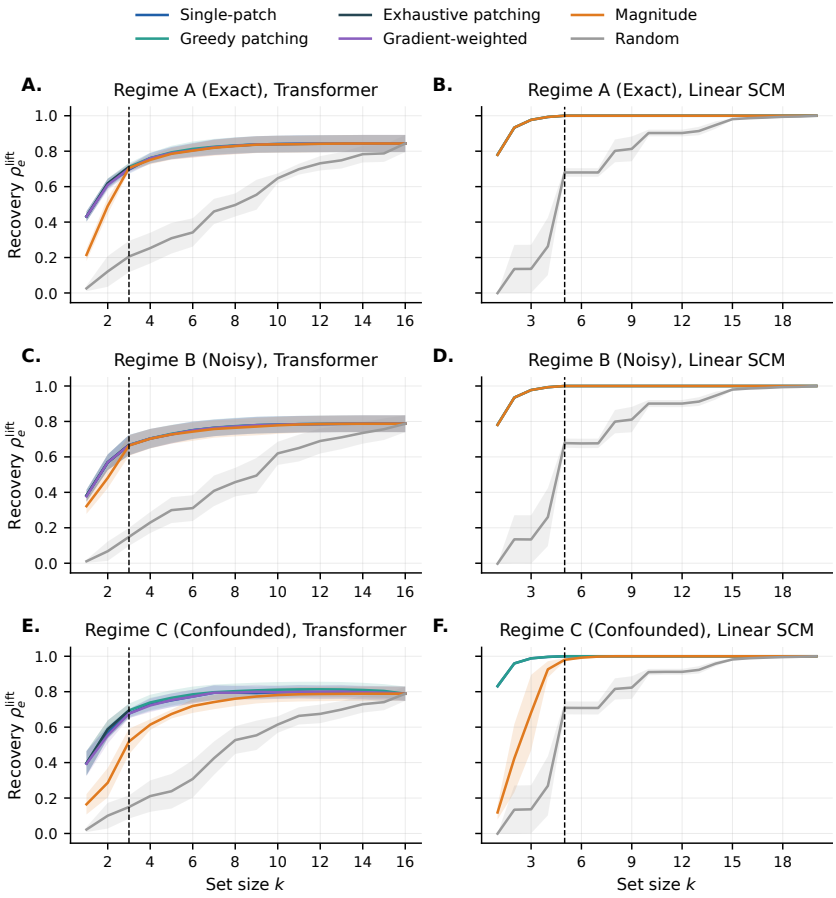

Figure 3: Synthetic mediator recovery across three regimes. Dense-lift recovery $\rho_e^{\text{lift}}(S)$ vs. mediator set size k. The left column is the transformer and the right column the linear SCM. Rows show Regime A (exact), B (noisy), and C (confounded). Gradient-weighted curves are shown for the transformer only. Bands are ± 1 s.e. over 5 seeds. The dashed line marks $k = |T_e|$.



A. Specialist forward pass. **B.** All internal components patched.

Figure 4: Why full internal patching can still be a chimera. If $\mathcal{M}$ omits embeddings, final layer norm, or the unembedding, then patching every component in $\mathcal{M}$ puts specialist internals between base endpoints. This motivates the patching ceiling c_e and the patchable-gap normalization ρ_e^{patch}.

tokens and of the final EOS token. The evaluation reuses the E2 test split, precision (float16 at 410M, bfloat16 autocast at 1.4B), batching, and aggregation (per-batch token mean, averaged over batches), and reproduces the reported dense lift of every run to within 2×10^{-4} nats. Because the lift is an average over target tokens, the two parts add up to the lift. The last two columns compare the specialist's NLL on the answer token alone with the entropy of the test-set label frequencies, which is the NLL of a constant predictor that outputs those frequencies.

On MMLU and PubMedQA the specialists assign almost all probability to EOS after the answer (specialist EOS NLL below 0.004 nats), whereas the base model does not. This is the largest single source of the lift. On the answer token, the specialists' NLL stays at or above the label-frequency entropy, and the PubMedQA specialist predicts yes with high confidence. The localized NLL lift on these two domains therefore reflects answer formatting, stopping, and the training label prior.

Table 7: Patching ceiling and component-type contribution. Dense-lift recovery $\mathrm{Rec}_e(S)/G_e$ for each component scope, mean $\pm$ s.e. over 3 seeds. The Attention+MLP row is the ceiling c_e. Attention-only captures most of the dense lift on MMLU and PubMedQA but only $\approx$ 73% on GSM8K, where MLPs contribute substantially. Values above 1 mean the patched model scores slightly better than the specialist, and negative values mean the patch lowers the score below the base.

Component scope	GSM8K	MMLU	PubMedQA
Attention only (384 heads)	0.735 ± 0.006	0.937 ± 0.036	1.034 ± 0.005
MLPs only (24 blocks)	-1.607 ± 0.110	0.745 ± 0.123	1.035 ± 0.015
Attention + MLP (full $\mathcal{M}$)	1.004 ± 0.001	0.997 ± 0.000	1.013 ± 0.001

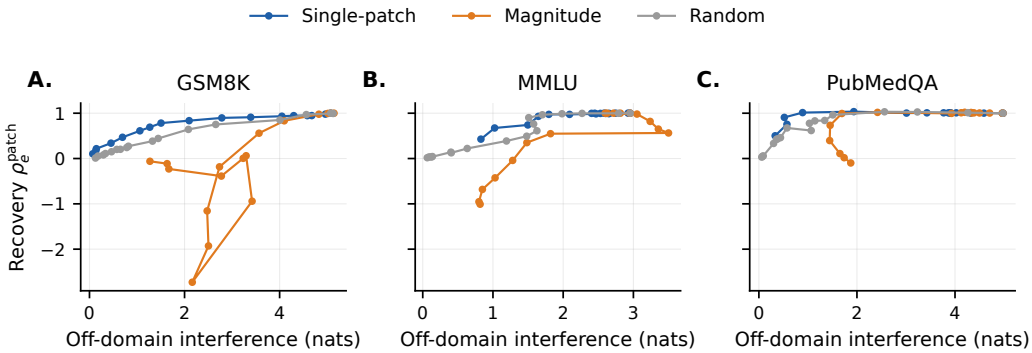

Figure 5: Recovery–interference trade-off. Each point is one budget k of one ranking, averaged over 3 seeds, with points connected in order of increasing k. Single-patch traces a higher-recovery, lower-interference frontier than magnitude or random. Interference is the absolute change in NLL on the two non-target domains, averaged.

I Exact single-component scores

The reported single-patch rankings (Section 5.2) score components in groups. One expanded forward pass patches each component of a group in its own copy of the batch while the other group members keep their base activations. When a group spans layers, a component's score excludes effects that pass through the later members of its group. This appendix compares those rankings with exact scores. In the notation of Proposition 4.2, it is a coordinate-wise finite difference of $F_{G,x,y}$ for the group G. In Pythia-410M the groups had 32 components (two layers of heads, or all 24 MLP blocks), and in Pythia-1.4B four. To obtain exact single-component scores, we regroup by layer. A layer's 16 heads and its MLP block read the same layer input under Pythia's parallel residual, so patching one of them does not change the others, and each score is the effect of patching that component alone (no site is an ancestor of another, as in Section 4). We recompute these scores on the same calibration items and evaluate set-patching recovery of their top-k components on the same test items. Recomputing the reported rankings' recovery with this code reproduces the reported recovery values to within 0.006 nats.

The two rankings agree closely in correlation but differ in their top few entries, mostly in how they order MLP blocks. The conclusion that a few components recover most of the answer-sequence NLL lift holds for both. With exact scores, five components recover 97% on MMLU, 98% on PubMedQA, and 88% at 1.4B, and GSM8K remains the domain that needs many components. Table 2 and Figure 2 use the reported ranking.

J Ranking stability

A localization is more useful when it does not depend on incidental choices. This appendix varies the calibration sample, the prompt template, the scoring metric, and the training seed.

Calibration sample. In the Pythia-1.4B run, we compare rankings from the first and second halves of the 256 shuffled calibration items. With model, specialist, and prompt fixed, they have full-rank

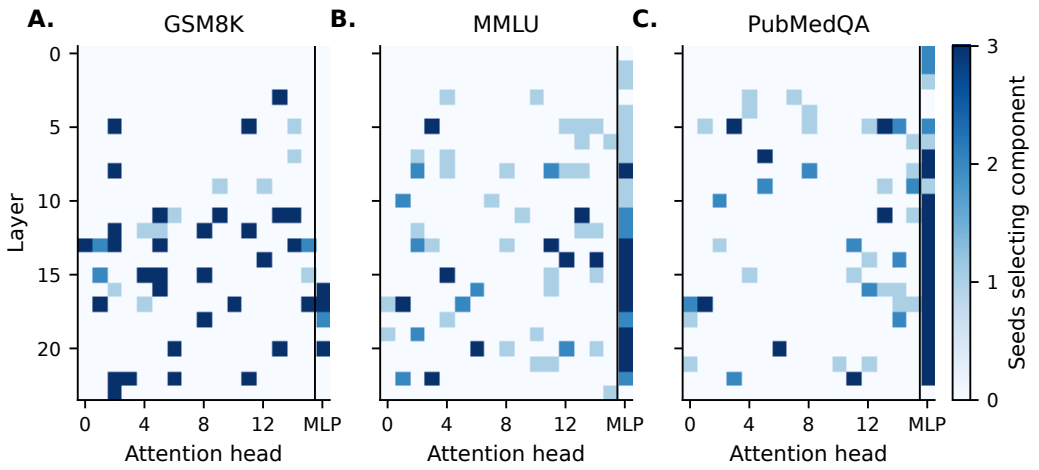

Figure 6: Where the top-ranked components sit. For each domain, the number of seeds (out of 3) in which a component is among the 40 highest single-patch scores (10% of $\mathcal{M}$). Columns 0–15 are attention heads, and the last column is the MLP block of each layer. MLP blocks are frequent among the top components on MMLU and PubMedQA.

Table 8: Composition of the dense NLL lift. EOS share is the fraction of the lift at the EOS position, answer-token NLL is the specialist's NLL on the answer token(s) alone, and label entropy is the entropy of the test-set label frequencies. Pythia-410M rows are mean (range) over 3 seeds, and the Pythia-1.4B row is one seed.

Model	Domain	Dense lift (nats)	EOS share	Answer-token NLL	Label entropy
410M	GSM8K	1.207	6.4% (6.4–6.5)	1.026 (per token)	–
410M	MMLU	3.820	89.2% (89.0–89.2)	1.43 (1.39–1.48)	1.38
410M	PubMedQA	8.379	61.6% (59.5–62.9)	2.70 (2.08–3.01)	0.93
1.4B	PubMedQA	5.309	55.4%	2.91	0.94

Spearman correlation 0.978, and top-k Jaccard is 1.00, 0.67, 1.00, 1.00, and 0.88 at $k = 3$, 5, 8, 16, and 32.

Prompt template. For the seed-0 Pythia-410M specialist of each domain, we recompute exact single-component scores (Appendix I) on the same 500 calibration items under the original template and under an alternate template that rewords the instruction, renames and reorders the fields, and keeps the "Answer:" cue and the target tokens. The alternate templates are: GSM8K, "Work out the answer to this grade-school math word problem.\nProblem: {question}\nAnswer:"; PubMedQA, "Abstract: {context}\nQuestion: {question}\nUsing the abstract, reply to the question with yes, no, or maybe.\nAnswer:"; MMLU, "The following is a multiple-choice question about {subject}.\n{question}\nOptions:\nA) . . . \nReply with the letter of the correct option.\nAnswer:". Table 10 reports the agreement. The specialist's calibration NLL lift barely changes on GSM8K (1.34 nats under both templates) but falls on MMLU (3.89 to 3.48) and PubMedQA (9.52 to 6.87), consistent with fine-tuning on a single template.

Accuracy versus NLL scores. In the same forward passes, we score each component by the change in option-likelihood accuracy on the calibration items (the fraction whose highest-probability allowed label at the answer position is the gold label). On MMLU the accuracy-based and NLL-based rankings are unrelated (Spearman -0.07, -0.17, and 0.00 for seeds 0, 1, and 2, with top-5 Jaccard at most 0.11), and 29–46% of components leave calibration accuracy unchanged. On PubMedQA they are moderately related (Spearman 0.66–0.73 across seeds and 0.55 at 1.4B, with top-5 Jaccard 0.11–0.43). The PubMedQA calibration items come from PQA-A, which is 92–93% yes, and the specialists' calibration accuracy (0.91–0.95) mostly reflects predicting yes, so an accuracy gain there measures movement toward yes. The NLL and mean target log-probability scores are equivalent, and accuracy is a separate endpoint.

Table 9: Exact single-component scores versus the reported single-patch ranking. Columns give the Spearman correlation between the two rankings, the number of MLP blocks among the top five of each, and the fraction of the dense NLL lift recovered on the test set by the top-k components of the exact (reported) ranking. Pythia-410M entries are means over 3 seeds (Spearman range in parentheses), and Pythia-1.4B is one seed.

Model	Domain	Spearman	MLPs in top 5	$k = 3$	$k = 5$	$k = 16$
410M	GSM8K	0.82 (0.81–0.83)	4.7 (1.3)	30% (22%)	42% (34%)	82% (69%)
410M	MMLU	0.87 (0.85–0.89)	3.3 (3.3)	88% (74%)	97% (93%)	99% (98%)
410M	PubMedQA	0.92 (0.90–0.95)	5.0 (5.0)	87% (92%)	98% (103%)	103% (102%)
1.4B	PubMedQA	0.96	5 (5)	79% (76%)	88% (94%)	107% (107%)

Table 10: Agreement between exact single-component rankings under two prompt templates. Pythia-410M, seed 0, 500 calibration items.

Domain	Spearman	Top-3 Jaccard	Top-5 Jaccard	Top-8 Jaccard	Top-16 Jaccard
GSM8K	0.99	1.00	0.67	0.78	1.00
MMLU	0.82	1.00	0.67	1.00	0.78
PubMedQA	0.71	0.50	0.43	0.60	0.68

Across training seeds. Rankings from specialists trained with different seeds, on different calibration draws, have mean pairwise full-rank Spearman correlation 0.96 on GSM8K, 0.62 on PubMedQA, and 0.44 on MMLU, and mean top-8 Jaccard 0.78, 0.78, and 0.21. These comparisons change the specialist and the calibration sample together.

K E3 extras: selection overlap, tiny-budget comparison, head-level adapters

This appendix adds three comparisons to Section 5.4. Table 11 reports how much each method's adapted blocks, read from the saved adapters, overlap with the patching-localized blocks at the same (domain, seed, k/N). The overlaps are close to the expected overlap of two independent uniformly random block sets of the same size.

Table 11: Median Jaccard overlap of selected blocks with patching-localized, pooled over domains and seeds (n=9 per cell). The last row is the expected overlap of two independent random sets of k out of 24 blocks.

Method	k/N=0.25	0.50	0.75
Random-localized	0.20	0.33	0.57
Dense-update-norm	0.09	0.26	0.64
Domain-probe	0.09	0.33	0.64
Full LoRA	0.25	0.50	0.75
Random sets (expected)	0.15	0.34	0.60

Block choice could matter more when the budget is very small, so the tiny-budget comparison (PubMedQA and MMLU, $k \in \{1, 2, 3\}$ blocks, $r \in \{4, 8\}$, 3 seeds, 10 random draws per cell) compares patching-localized against random-localized (median over draws) and dense-update-norm. The patching-minus-random NLL gap (positive favors random) is $+0.002$ $[-0.002, 0.006]$ on MMLU and $+0.053$ $[-0.008, 0.114]$ on PubMedQA, with both 95% CIs including zero.

Head-level versus block-level adapters. E2 ranks individual heads while LoRA adapts whole blocks, so we also train an adapter restricted to individual heads. At the largest E3 budget on PubMedQA (k/N=0.75, r=16, 18 blocks), we compare the block-level adapter (attention and MLP projections of the 18 selected blocks, 4,718,592 parameters) with a head-level adapter that applies rank-60 LoRA only to the query, key, value, and output projections of the highest-ranked head in each selected block (4,700,160 parameters). Test NLL is 1.010 ± 0.107 for the head-level adapter and 1.287 ± 0.286 for the block-level adapter (mean $\pm$ s.e. over 3 seeds). Paired by seed, the differences are -0.61, 0.00, and -0.21 nats. Both adapters predict yes on every test item, so their option-likelihood accuracy equals the majority-label rate. Both were trained on the 256 E3 training examples.

L Reproducibility and asset notes

The experiments use the public assets Pythia-410M and Pythia-1.4B (Apache License 2.0), GSM8K (MIT License), MMLU (MIT License), and PubMedQA (MIT License), loaded through the Hugging Face `transformers` and `datasets` libraries. Code, derived result tables, and the scripts that regenerate every table and figure are released at https://github.com/AsiaeeLab/specialist-mediators. All language-model experiments ran on single NVIDIA RTX 4500 Ada GPUs (24 GB). Single-patch scoring over 408 components and 500 calibration examples takes 13–27 minutes per domain and seed at 410M with float16 and a peak of 14 GB. Training one E3 adapter takes under one minute, and the full E3 grid of 405 adapters and its NLL re-evaluation take a few GPU-hours.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the formal object, the conditional theory, and the empirical results developed in Sections 3–5, including the localized quantity, answer-sequence NLL. Limitations are discussed in Sections 5.2–6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sections 5.2 and 5.4 state that accuracy separates the models only weakly, the EOS share of the NLL lift, the PubMedQA label shift, and the selection-support mismatch in E3. Section 6 states the model and seed coverage, and Remark 4.4 states the scope of the transfer theorem.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The theoretical statements in Section 4 state their assumptions explicitly, and complete proofs are in Appendices A and B. Appendix C gives the block-norm form of the linearization remainder bound.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 5 and Appendix D specify data sources and splits, prompt templates, scored targets, optimizers, budgets, metrics, seeds, and baselines, and Appendices H–I describe each additional analysis. The released code includes the commands that regenerate every table and figure from saved checkpoints.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper uses public datasets and public base models. Code, derived result tables, and reproduction instructions are released at the repository given in Appendix L.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 and Appendix D specify the data splits, optimizers, learning rates, epochs, budgets, metrics, and baselines needed to understand the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Tables and figures report mean $\pm$ standard error across training seeds (five in E1, three in E2 and E3), computed from per-seed values; the one-seed Pythia-1.4B check reports an item-level standard error and a paired bootstrap interval, and the tiny-budget comparison reports 95% intervals (Appendix K). Differences between adapters are reported against the across-seed variation.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix L gives the GPU type and memory, the wall time of single-patch scoring and adapter training, and the scale of the full E3 grid; Appendix D gives the Pythia-1.4B training and scoring times.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, this research conforms to the NeurIPS Code of Ethics. We use publicly available models and datasets and do not involve human subjects or sensitive private data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 6 discusses the positive diagnostic use of specialist localization and the risk that cheaper specialization can lower the barrier for harmful specialists. The discussion also notes localization itself as a possible detection diagnostic.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not introduce or release a new high-risk pretrained model, scraped dataset, or generative system. The intended release is method code and derived experimental artifacts.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The models and datasets are cited where first used, and Appendix L lists their licenses: Apache 2.0 for Pythia and MIT for GSM8K, MMLU, and PubMedQA.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The released code and derived result tables carry a README with reproduction commands, a map from paper tables and figures to scripts, and an MIT license (Appendix L).

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Pretrained language models are the experimental subjects of the paper, but LLM use is not a non-standard component of the method development beyond studying and fine-tuning those models.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.