NOT ALL DRIFT IS A MECHANISM: TRACING PROMPT ADAPTATION IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Prompt-based adaptation offers several ways to specialize a frozen language model, ranging from natural-language prompt optimization to learned continuous prompts. These methods are usually compared through downstream performance, although similar outputs need not arise from similar internal computations. We study how discrete prompt optimization, prompt tuning, and prefix tuning alter the processing of dense and mixture-of-experts language models. Our analysis shows that the location of the adaptation predicts how task information is carried to the output: optimized text primarily changes response construction, input-level prompts write their influence into content representations, and layer-wise prefixes maintain direct control through attention. In MoE models, these differences further induce distinct expert-routing patterns. At the same time, common representation-level measurements respond strongly to generic context changes and do not reliably distinguish useful adaptations from matched placebos. We argue that understanding prompt adaptation requires separating representation drift from the part of that drift that is aligned with, and causally used by, the model's task computation. This perspective provides practical guidance for selecting, evaluating, and interpreting prompt adaptation methods.

1 INTRODUCTION

Large language models (LLMs) are increasingly adapted as frozen backbones, allowing one model to support many tasks without storing or serving a separate set of weights for each application. Context becomes the adaptation interface: a task can be specified through a natural-language instruction, learned continuous vectors at the input, or learned states inserted into the attention computation (Agrawal et al., 2026; Lester et al., 2021; Li & Liang, 2021).

The most direct approach searches for a better textual prompt. Early methods optimized individual tokens, while recent systems use language models to propose, critique, and refine complete instructions (Shin et al., 2020; Pryzant et al., 2023). A complementary line replaces text with trainable continuous states. Prompt tuning learns input embeddings, whereas prefix tuning exposes learned keys and values to every Transformer layer (Lester et al., 2021; Li & Liang, 2021). These methods optimize the context seen by the same frozen model, but provide different access to its computation.

In practice, these methods are usually compared by downstream score. This is insufficient for understanding what the adaptation changed. A gain may arise from a better task decision, more reliable adherence to an output protocol, or a different route through which an existing decision reaches the language-model head. In a mixture-of-experts (MoE) model, the same contextual change can additionally alter which expert parameters process each token (Olson et al., 2025). Similar outputs therefore need not imply similar internal computations.

Research on prompt optimization has made substantial progress in improving task performance, while mechanistic studies of contextual learning have identified task vectors and prompt-induced geometric changes (Agrawal et al., 2026; Todd et al., 2024; Kirsanov et al., 2025). These directions have largely developed separately. Performance-oriented studies rarely locate the computation that mediates the gain, while representation analyses often show that a prompt changes hidden states without testing whether the measured change is sufficient to alter the output (Hewitt & Liang, 2019; Zhang & Nanda, 2024).

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

We study prompt adaptation from this functional perspective. We compare discrete prompt optimization, prompt tuning, and prefix tuning under a common frozen-model protocol, follow task information through a dense model, and test observational signatures against matched placebos and causal interventions. We then examine sparse models, where a hidden-state change can become a discrete routing decision. Our results support a common account: the insertion site constrains how an adaptation can influence the network, while common internal metrics often capture broad representation drift rather than the smaller state change sufficient to mediate the adapted output.

Contributions. Therefore, we contribute the following:

- **A mechanistic characterization of prompt adaptation.** We provide a unified empirical study of discrete prompt optimization, prompt tuning, and prefix tuning in dense and MoE language models. The analysis connects each method's intervention site to the way it preserves task information, controls generation, and changes expert selection, yielding concrete guidance about what each adaptation mechanism is likely to provide in practice.
- **A functional account of adaptation-induced representation change.** We show that internal movement alone is not evidence of a useful mechanism. Format-aware evaluation, matched placebos, and causal interventions separate generic context drift from the task-aligned changes that affect model behavior, and expose several cases in which otherwise plausible interpretability measurements are misleading.

2 Related Work

Continuous prompts. Continuous prompting adapts a frozen model by optimizing trainable states rather than model parameters. Prefix tuning introduced task-specific keys and values that remain available throughout the attention stack (Li & Liang, 2021), while prompt tuning showed that embeddings inserted only at the input become increasingly competitive as model scale grows (Lester et al., 2021). Deep variants such as P-Tuning v2 extend these states across layers and tasks (Liu et al., 2022). These methods therefore differ not only in parameter count, but also in the depth at which they can influence computation. Theoretical analyses characterize both the expressive power of prompt-mediated attention and the restrictions imposed by a frozen backbone (Oymak et al., 2023; Petrov et al., 2023). Their learned states remain difficult to interpret, since nearest-token projections can be unfaithful and human-readable constraints may reduce task performance (Patel et al., 2025).

Discrete prompt evolution. A parallel line retains the natural-language interface and searches for better instructions. Prompt sensitivity first motivated gradient-guided trigger selection in AutoPrompt (Shin et al., 2020) and reward-based token generation in RLPrompt (Deng et al., 2022). With stronger instruction-following models, optimization moved from isolated tokens to complete prompts: APE samples and selects candidate instructions (Zhou et al., 2023), ProTeGi combines textual gradients with beam search (Pryzant et al., 2023), and OPRO conditions an optimizer on previous solutions and scores (Yang et al., 2024). Later methods jointly optimize instructions, demonstrations, or multistage programs (Opsahl-Ong et al., 2024). GEPA adds natural-language reflection and evolutionary selection (Agrawal et al., 2026), while SPRIG studies iterative system-prompt optimization and transfer (Zhang et al., 2026). This progression makes discrete adaptation accessible without gradients through the task model, but most evaluations still describe the resulting behavior rather than the computation that produced it.

Mechanistic analysis. Mechanistic studies of contextual learning first asked which computations prompts elicit. Induction heads identify circuits that copy contextual patterns (Olsson et al., 2022), simplified Transformers can implement optimization-like updates during the forward pass (von Oswald et al., 2023), and task or function vectors can transfer behavior under intervention (Hendel et al., 2023; Todd et al., 2024). This work motivated observational diagnostics of prompt-induced computation, including representation geometry (Kirsanov et al., 2025), task descriptions recovered through Patchscopes (Ghandeharioun et al., 2024; Ramati et al., 2025), and hidden-state shifts after system-prompt optimization (Zhang et al., 2026). Such measurements establish that context changes the network, but not that the measured change produces the output. Linear probes and the tuned lens localize decodable information (Hewitt & Liang, 2019; Belrose et al., 2023); activation steering and

patching test causal sufficiency (Turner et al., 2023; Zhang & Nanda, 2024); and sparse autoencoders provide a feature basis in which these interventions can be expressed (Cunningham et al., 2023; Gao et al., 2024). Our comparison brings these stages together for three adaptation mechanisms under the same task and backbone, using matched placebos to test observational measurements and causal transfer to identify state-level mediators.

MoE routing. This distinction becomes more explicit in mixture-of-experts models, where a contextual change can alter which parameters process a token. MoE Transformers route each token through a small expert subset, increasing total capacity at approximately fixed active computation (Fedus et al., 2022). Load-balancing objectives limit concentrated traffic and unused experts (Wang et al., 2024), while recent analyses show that routing reflects both token identity and contextual semantics (Olson et al., 2025). Existing adaptation work mainly modifies weights, routers, or routed adapter modules (Liu et al., 2025). The response of a frozen router to discrete, input-level, and layerwise prompt adaptation remains comparatively unexplored.

3 Dense Models: From Adaptation Interface to Functional State Change

GEPA, prompt tuning, and prefix tuning all optimize context while keeping the language model frozen, but they expose different parts of its computation to that optimization. GEPA rewrites natural-language tokens, prompt tuning learns embeddings inserted only at the input, and prefix tuning supplies learned keys and values to every attention block (Agrawal et al., 2026; Lester et al., 2021; Li & Liang, 2021). These insertion sites define distinct routes by which an adaptation can remain available during generation: through textual instructions, through content states that inherit an input-level intervention, or through a layerwise attention channel.

For input x, let $h^M_{\ell,t}(x) \in \mathbb{R}^d$ denote the residual state after block ℓ at token position t under condition M, with $M = 0$ denoting the seed instruction. We write τ for the final non-padding prompt position, use *block* and *layer* interchangeably, and define the seed-relative displacement

$$\delta^M_{\ell,t}(x) = h^M_{\ell,t}(x) - h^0_{\ell,t}(x). \tag{1}$$

Experiments use Gemma 2 2B-IT (Gemma Team et al., 2024) on five-label Civil Comments classification (Borkan et al., 2019). The primary metric is empty-aware samples-F_1. A strict parser enforces the prescribed label order, whereas a set parser evaluates only the predicted label set. Data partitions, adapter objectives, and complete protocols are given in Appendix B.2.

The insertion site specifies where an adaptation can enter the network, but it does not identify which downstream consequence produces the gain. Likewise, the displacement in Equation (1) establishes that a state moved, but not that the movement is task-relevant. The analysis therefore separates changes in the generated answer from changes in task information, then tests which residual changes survive matched controls and causal transfer.

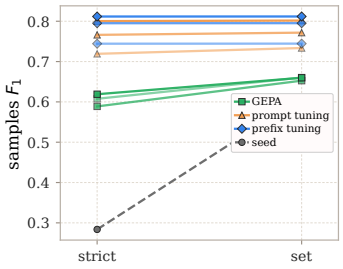
Research question 1

Which consequences of an adaptation's insertion site remain active in the output computation, and which measured state changes are generic effects of changing the context?

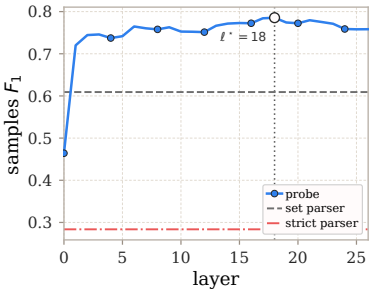
3.1 Adaptation changes both the decision and its realization

The strict evaluator combines label selection with answer construction. Under set parsing, most of GEPA's advantage over the seed is removed, whereas prompt and prefix tuning retain their gains (Figure 1a). GEPA's observed gain is therefore concentrated in the realization of the output contract in this task, while the gains of the continuous methods remain visible in the selected label set. The full score sweep is reported in Appendix A.

The seed computation already contains substantial label information. A linear probe trained on seed states reaches most of its final accuracy by the middle of the stack (Figure 1b). Adaptation is therefore not applied to a representation devoid of the task; it changes how available information is transformed and exposed to the output computation.



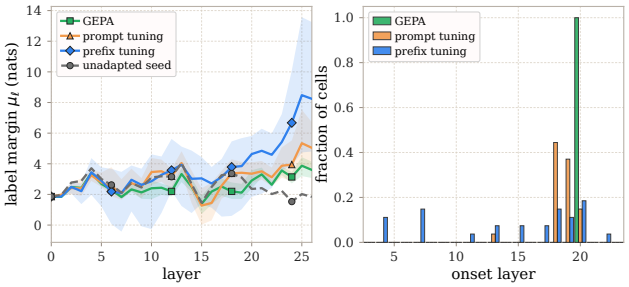
(a) Label selection versus answer construction.



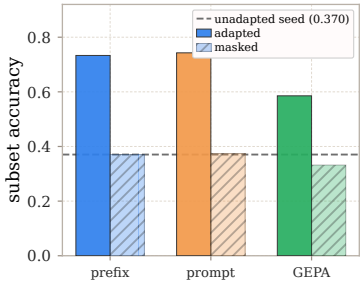
(b) Seed-state label information.

Figure 1: Behavioral gains and pre-existing task information. Ignoring label order removes most of GEPA's strict-score improvement, while a seed-state probe decodes the label set well before the final block.

A calibrated tuned lens tracks the layerwise gold-label margin under each adapted context relative to the seed (Belrose et al., 2023). Prefix tuning produces the earliest observed onsets in several configurations, consistent with keys and values that are available at every block. Prompt tuning and GEPA concentrate their onsets later in the stack (Figure 2a). These readouts describe availability and masking establishes that the adapted contexts remain functionally involved. Removing attention to the optimized instruction, prompt embeddings, or prefix states returns performance to the seed level (Figure 2b).



(a) Depth at which adapted label margins become readable.



(b) Adapted-context dependence.

Figure 2: Availability and use of the adapted context. The tuned lens tracks the emergence of adapted label margins, while masking removes the gain when access to the corresponding context is blocked.

Together, the behavioral and readout results give an interface-level account. GEPA's gain is concentrated in answer construction, prompt tuning introduces an input-level influence whose label margin becomes readable later in the stack, and prefix tuning maintains a layerwise route to the attention computation. Representative token-level attention profiles provide a complementary view in Appendix B.4. Residual movement, however, also follows from changing the context without optimizing it, so access alone does not make the resulting geometry task-specific.

3.2 Generic context produces substantial representation drift

A length-matched padding prompt isolates the effect of adding context positions without task optimization. It changes residual geometry, degrades probe transfer, and lowers task performance. Padding therefore breaks any monotonic link between distance from the seed and adaptation quality (Figure 7a in Appendix B.5).

The structure of the displacement leads to the same conclusion. Early in the stack, the dataset mean accounts for most of the shift energy under every condition, including padding. At later blocks, the centered remainder becomes high-rank for both adapted and padded contexts (Figure 7b). A large shared component and a structured per-example remainder therefore also arise when context is added without task optimization; neither signature identifies a useful adaptation by itself.

This placebo changes the interpretation of the observational measurements. Distance, probe transfer, mean energy, and effective rank establish that the contexts induce different representations, but none identifies the part of the change that is sufficient to reproduce adapted behavior. That distinction is resolved by intervening on the seed computation.

3.3 Per-example state differences transfer adapted outputs

The simplest intervention treats adaptation as a shared direction. At block ℓ and generation anchor τ, we average the displacement over N examples and add it once to the seed state:

$$v_\ell^M = \frac{1}{N}\sum_{i=1}^{N} \delta_{\ell,\tau}^M(x_i), \qquad \widetilde{h}_{\ell,\tau}^0(x;\lambda) = h_{\ell,\tau}^0(x) + \lambda v_\ell^M. \tag{2}$$

Here λ controls the magnitude of the injected direction.

Across the evaluated layers and scales, the prefix mean performs like a norm-matched random direction; the GEPA mean produces the same qualitative comparison (Figure 3). The dominant average movement in the tested intervention therefore does not reproduce the adapted behavior. Adding the prefix mean at every decoded position recovers about 9% of the prefix gain (Appendix C.6).

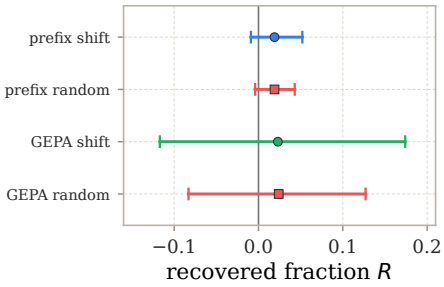


Figure 3: Dataset-mean steering and its random control.

Per-example state transfer gives a different result. For source condition S and target condition T, we add the matched-example difference $h_{\ell,\tau}^T(x) - h_{\ell,\tau}^S(x)$ to the source state and run the remaining source blocks unchanged (Zhang & Nanda, 2024). For source, target, and patched next-token distributions $p_S(x)$, $p_T(x)$, and $p_{S\rightsquigarrow T}(x)$, transfer is measured by

$$R_{\mathrm{KL}}(S\to T) = 1 - \frac{\sum_x \mathrm{KL}(p_T(x)\|p_{S\rightsquigarrow T}(x))}{\sum_x \mathrm{KL}(p_T(x)\|p_S(x))}. \tag{3}$$

We run this patch for GEPA paired with each continuous method, in both directions. At a late residual state, the exact per-example difference closes most of the source–target KL gap when aggregated across the two directions (Table 1). Sparse-autoencoder reconstruction and feature-restricted transfer provide a complementary decomposition in Appendix B.6 (Lieberum et al., 2024).

Table 1: Exact transfer at block 20.

Conditions	R_{KL}
GEPA–prompt	0.917 ± 0.049
GEPA–prefix	0.832 ± 0.056

The two interventions separate broad drift from a transferable state difference. The tested dataset mean does not reproduce the gain, whereas one late matched-example difference is sufficient to transfer most of the next-token distributional difference. The exact patch uses the target state directly; the remaining step is to produce a comparable correction from the seed computation alone.

3.4 Learned corrections recover adapted generation

For target method M, we learn an affine correction from the seed residual state at the final block L:

$$\widehat{\delta}_{L,t}^M = f_\phi^M(h_{L,t}^0) = W_M h_{L,t}^0 + b_M, \qquad \widetilde{h}_{L,t}^M = h_{L,t}^0 + \widehat{\delta}_{L,t}^M. \tag{4}$$

The map has about 5.3M parameters. Training pairs come from the adapted model's own greedy answers: we run the seed and adapted models on each answer and regress the seed state onto the displacement $\delta_{L,t}^M$, with state MSE alone or together with KL divergence from the adapted output distribution (Equations (17) and (19)). During free generation, the map reads the current seed-context state and the answer prefix produced with preceding corrections, then predicts a new correction at every token. The adapted context and target states are absent at inference. Heyman & Vandeputte (2026) learn activation-dependent corrections that imitate textual prompting with a single learned direction, and Cheng & Kriegeskorte (2026) study which state maps reproduce prompt-induced task structure. Our maps are full affine corrections, fitted to all three adaptation interfaces and evaluated by the score of complete generated answers.

Table 2: Complete-answer set samples-F_1.

Target	Native	MSE	MSE+KL
GEPA	$0.678 \pm .007$	$0.683 \pm .015$	$0.681 \pm .012$
Prompt	$0.777 \pm .013$	$0.746 \pm .001$	$0.759 \pm .006$
Prefix	$0.805 \pm .007$	$0.744 \pm .004$	$0.765 \pm .006$

The learned corrections recover the small GEPA gap and most of the continuous gains (Table 2), 81% and 68% of the prompt and prefix gains with MSE and 89% and 79% with MSE+KL. With MSE+KL, one correction computed at the start of the answer comes within 0.010 of the per-token predictor (Table 6), whereas a constant bias does not beat the seed for prompt and prefix tuning, so the gain depends on the input. The true shifts of GEPA and either continuous method have median cosines near 0.2, against 0.46 between prompt and prefix tuning (Appendix C.4), matching the split in Figure 1a. A map from the seed block-20 state to the adapted block-25 state can even replace blocks 21 to 25, reaching 0.688, 0.766 and 0.770 (Appendix C.3). Training details and the rank sweep are in Appendices C.1 and C.2.

Dense-model conclusion. The dense experiments form a single progression. Behavioral evaluation first separates label selection from answer construction; layerwise readouts then show how adapted information becomes available; and the padding placebo shows that the resulting representation drift is not specific to task improvement. Causal transfer isolates the finer distinction: the tested dataset mean is ineffective, while a matched-example late-state difference transfers most of the adapted output difference. Learned affine corrections recover much of the same behavior throughout free generation, even with one correction per example when trained with output supervision. This establishes behavioral transfer, but not that the maps reproduce the native adaptation mechanism.

Every dense residual state still passes through the same feed-forward parameters. In an MoE backbone, the same state change can also alter which experts process the token. Routing therefore exposes a discrete intermediate decision that reveals whether adaptation changes not only the state, but also the parameters that act on it.

4 MoE Models: When Drift Becomes a Discrete Decision

A Transformer block combines an attention layer with a feed-forward network. In a dense model, the feed-forward network applies the same parameters to every token, so an adaptation can change only the states it processes. A mixture-of-experts (MoE) model replaces this network with n_E experts $f_{\ell,e}$ and a linear router W_ℓ^r that sends each token to k of them, so different tokens, and the same token in a different context, can take different paths through the network. For the state $h_{\ell,t}$ with which token t enters the expert block of layer ℓ,

$$g_{\ell,t} = W_\ell^r h_{\ell,t}, \qquad \mathcal{E}_{\ell,t} = \text{top-}k\big(g_{\ell,t}\big), \qquad y_{\ell,t} = \sum_{e \in \mathcal{E}_{\ell,t}} \pi_{\ell,e}(h_{\ell,t})\, f_{\ell,e}(h_{\ell,t}), \tag{5}$$

where $\pi_{\ell,e}$ are the softmax weights of the selected experts. An adaptation that changes $h_{\ell,t}$ changes the inputs and weights of the selected experts and can also change the set $\mathcal{E}_{\ell,t}$ itself. Since the experts chosen in one layer shape the state that reaches the next router, such changes can compound with depth.

The router was trained on the states of the frozen base, not for the new task, and adaptation changes the distribution of its inputs. Rerouting is therefore expected, and it may be what the new task needs. Because the router also spreads the tokens over the experts, every change of routing is a change of load, and the load decides what the model costs to serve: under expert parallelism the busiest device sets the pace of a layer, and a rarely used expert holds memory while doing little work.

Let $u_{\ell,e}$ and $\tilde{u}_{\ell,e}$ be the shares of the layer-ℓ assignments that expert e receives in the frozen base and in an adapted model. Unless stated otherwise, both are counted over the tokens of the classified comment, which are identical in every arm, whereas the rest of the sequence differs between arms in length and content. How far adaptation moves this profile is measured by the total variation distance TV_ℓ between $\tilde{u}_{\ell,\cdot}$ and $u_{\ell,\cdot}$, the fraction of routing mass that must move to restore the base profile, from 0 for the same profile to 1 for disjoint experts (Appendix E.1). For example, in layer 1 of gpt-oss prefix tuning with 500 virtual tokens raises the share of two experts from 2 and 3 percent of the comment's assignments to 15 and 14 percent; with smaller moves elsewhere this gives $\mathrm{TV}_1 = 0.31$, against 0.02 for GEPA. How unevenly a profile spreads the load is measured by its Gini coefficient,

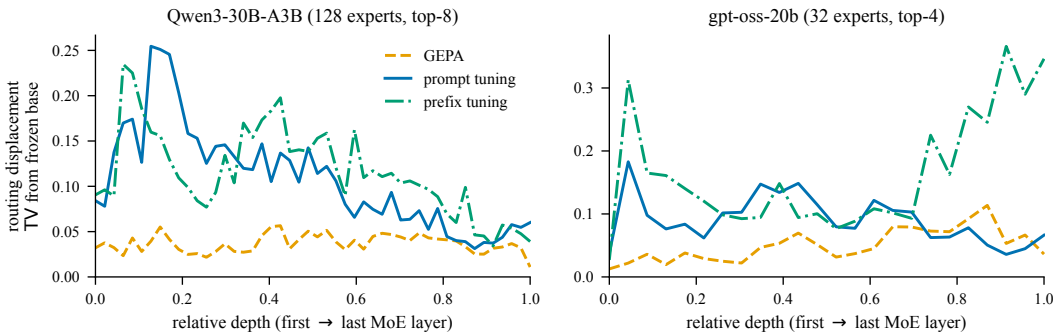


Figure 4: Routing displacement TV_ℓ from the frozen base, layer by layer, for the comment tokens that all arms share; continuous arms at 500 virtual tokens.

0 for a uniform profile and close to 1 when one expert takes every token, and by the busiest expert's share $\max_e u_{\ell,e}$.

In this section we study how GEPA, prompt tuning and prefix tuning change the routing of a frozen MoE model and the load it induces. Specifically, we ask the following question:

Research question 2

How does each adaptation method change which experts process a token and how evenly the load is spread, and does an expert's load tell whether the adapted model needs it?

The second half of the question matters because practice acts on the load. Auxiliary losses and router biases keep it balanced during pretraining and fine-tuning (Fedus et al., 2022; Wang et al., 2024), and expert pruning removes the experts a trained model rarely uses to make inference cheaper (Lu et al., 2024). Pruning thus assumes that a rarely used expert is dispensable, and masking the experts of an adapted model by their load tests that assumption directly. We measure how adaptation reroutes the tokens it shares with the frozen model (Section 4.1) and changes the load it serves (Section 4.2), prune by frequency (Section 4.3), and study the super experts, a few rarely used experts with extreme outputs (Section 4.4).

The task and the adaptation methods are those of Section 3, scored by empty-aware samples-F_1 on the predicted label set. The backbones are Qwen3-30B-A3B-Instruct-2507 (48 MoE layers, 128 experts, top-8) and gpt-oss-20b (24 MoE layers, 32 experts, top-4). Both route with a linear top-k gate and softmax weights over the selected experts, gpt-oss with an added router bias, and neither has a shared expert. The continuous methods use 100, 200 and 500 virtual tokens. Routing is logged on 500 held-out calibration rows, disjoint from the test set (Appendix E.1).

gpt-oss reasons before it answers, as do its base and GEPA arms; its prompt- and prefix-tuned arms, trained on answer-only targets, do not, as is known for reasoning models fine-tuned without reasoning traces (Twist et al., 2026; Lobo et al., 2025). Their gains on gpt-oss thus mix adaptation with a change of generation mode.

4.1 Only continuous adaptation substantially reroutes the shared tokens

GEPA moves only a few percent of the routing mass of the comment tokens, at every depth and on both backbones, whereas prompt and prefix tuning move several times more (Figure 4). Routing in MoE models is determined mostly by token identity and to a lesser degree by context (Xue et al., 2024; Olson et al., 2025). GEPA leaves the comment tokens unchanged and rewrites only the natural-language instruction around them, whereas virtual tokens and prefix states are learned vectors, not restricted to the vocabulary, that change the context more strongly. For the tokens it shares with the base, a text-level adaptation therefore keeps the expert assignments on which expert placement, caching or pruning of the base would be tuned; a continuous one does not. It also leaves the router's mismatch with the task in place, which a retrained router still removes (Appendix E.4).

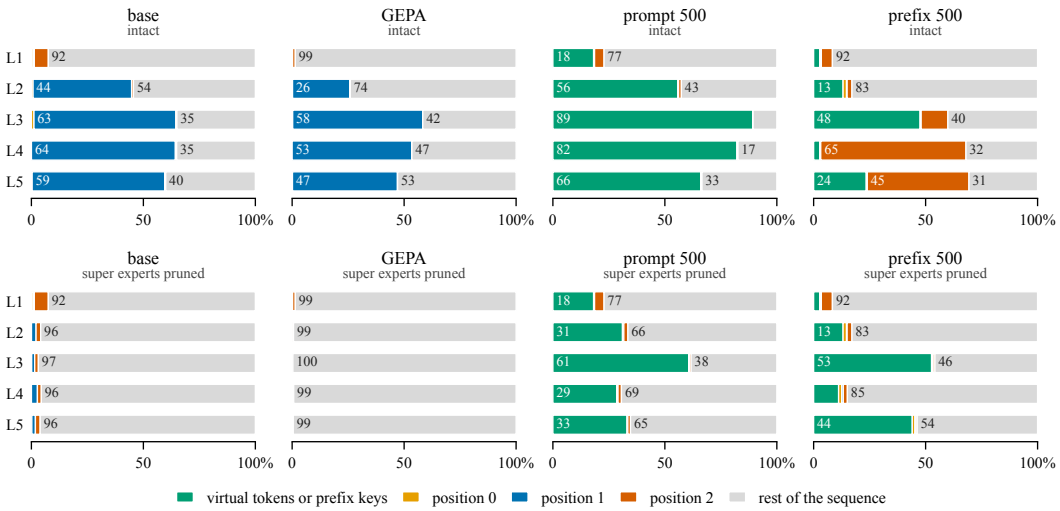


Figure 5: Share of attention by layer on Qwen, intact and with the super experts pruned.

The displacement does not compound with depth. On Qwen, both continuous methods move the early and middle layers most and the last quarter of the layers least, and prompt tuning does the same on gpt-oss: the late layers route the comment tokens nearly as the base does.

Prefix tuning on gpt-oss is the exception: its displacement rises again in the last layers, toward the experts the base uses for the text it generates (Appendix E.1), as if the prefix, present in every layer, prepared there the direct answer that replaces reasoning.

4.2 The load changes with the routed sequence, not with rerouting

On the comment tokens, adaptation reassigns the load without concentrating it: the Gini coefficient of every arm stays close to the base's (Table 19 in Appendix E.6), although the same tokens move up to a third of their routing mass in some layers. Adaptation changes which experts are busy, not how uneven the load is.

An adapted model, however, serves its whole sequence, and there the load does change, through the tokens each method adds or removes rather than through rerouting (Appendix E.8). GEPA's long instruction multiplies the routed positions, so its serving cost lies in the length of its prompt rather than in its routing. The virtual tokens of prompt tuning on Qwen go largely to experts the base almost never uses and spread the load. On gpt-oss every adapted arm concentrates it, and the tuned arms do so because they no longer write the reasoning over which the base spreads its load.

A load profile measured on one arm therefore does not transfer to another. Under expert parallelism these shifts cost less than the base's own imbalance: timing the MoE layers on two to eight GPUs, the base is 3 to 41 percent slower than under a uniform load, and adaptation adds at most 6 percent on two and four GPUs and 14 percent on eight (Appendix E.5).

4.3 Frequency pruning removes experts the adapted model needs

A concentrated load leaves many experts nearly idle: on this task the least-used half of every layer carries only 2 to 13 percent of the routing mass, and three quarters 18 to 40 percent. Such experts are the natural candidates for removal. We prune them by masking, in every layer, the router logits of the least-used half or three quarters of the experts, ranked on the arm's own calibration map.

Masking half of the experts leaves most arms close to their original score, and masking three quarters breaks every arm, mostly through answers that are unreadable or name no label (Table 12 in Appendix E.2). The damage does not follow the removed traffic. On Qwen, prefix tuning at 500 tokens loses every readable answer when its mask removes 5.5 percent of the routing mass, whereas prompt tuning at 500 tokens keeps its score after losing 12.3 percent (Appendix E.2). On gpt-oss

the prefix arms keep readable answers even at -75%, whereas the prompt arms lose many of them already at -50%. A mask built from traffic therefore cannot be judged by the traffic it removes; what matters is which experts it removes.

4.4 Rarely used super experts decide whether an adapted model survives

Load is not the only way in which experts differ. Language models send much of their attention to a few initial tokens with little content, the attention sinks (Xiao et al., 2024), which are created by massive activations, a few hidden-state entries orders of magnitude larger than the rest (Sun et al., 2024; Gu et al., 2025). In MoE models these are written by a handful of rarely selected super experts whose removal collapses the model (Su et al., 2026): experts 68, 92 and 82 of layers 1 to 3 on Qwen and expert 5 of layers 6 and 17 on gpt-oss (Appendix E.3). Unlike a Qwen head, every head of gpt-oss has a learned sink logit that lets it attend to no token (OpenAI, 2025).

Qwen's router sends its super experts about the uniform share of tokens and gpt-oss's about three times that share, so frequency masks remove all of Qwen's but never both of gpt-oss's. Every Qwen cell whose mask removes all three loses every readable answer, while masks of the same size that spare them keep the score (Appendix E.2).

How much an adapted model still uses its super experts depends on the adaptation (Table 15 in Appendix E.3). GEPA leaves them as they are, since its instruction begins after the positions where they fire; prompt tuning moves their peak onto a virtual token, which then draws most of the attention (Figure 5); prefix tuning can suppress them by up to three orders of magnitude, depending on the budget and the learning rate. Pruning them costs an arm about as much as it still uses them: nothing where prefix tuning has suppressed them, as on gpt-oss, and every readable answer on the Qwen arms that keep them, except prompt tuning at 100 tokens, which loses half its gain. gpt-oss hardly depends on them, since its learned sink takes the attention they leave (Figure 13), whereas Qwen's attention sink disappears with them.

MoE conclusion. Adaptation reroutes an MoE model where it acts: GEPA keeps the expert assignments of the shared tokens, prompt and prefix tuning reassign many of them. Load statistics and pruning masks built from them are specific to each adapter, and load does not show which experts the model needs: a few rarely used super experts decide whether an arm survives pruning.

5 Conclusion

Prompt-based adaptation methods differ less in how much they change a frozen model than in where they act, with practical consequences for evaluation, interpretation and deployment. Under set parsing most of GEPA's gain disappears while the continuous methods keep theirs, so an optimized prompt should be scored both ways before its gain is credited to task knowledge. A length-matched padding prompt reproduces the distance, probe, mean-energy and rank signatures of adaptation without improving the task, so representation metrics need such a placebo as a reference. An affine map from the seed state recovers most of the continuous gains without the adapted context and can even replace the last blocks, which makes such maps a practical probe of what an adaptation computes, although they need not reproduce the native mechanism.

In MoE models, retraining the router alone recovers most of the adaptation gain, and the choice of adaptation is also a serving decision. Text-level optimization keeps the routing of shared tokens but lengthens every request, whereas continuous prompts reroute many of them, so expert placement, caches and pruning masks tuned on the base should be recomputed per adapter. Pruning by traffic alone is unsafe, since a few rarely used super experts decide whether a pruned model still answers and the adaptation decides whether it needs them.

AI use statement

In this work, we used generative AI tools to aid and polish the writing and for retrieval and discovery, such as finding related work. We have not used generative AI tools for research ideation or execution, including proposing hypotheses, designing experiments, implementing methods and interpreting results, for drafting sections of the paper, for generating synthetic datasets, or for translation, and

proving mathematical claims is not applicable to this work. Additionally, we used generative AI tools to create and modify scientific figures and to review software code for errors. We have reviewed all AI-assisted work. We read every cited paper and checked each reference against its original source. All plotted values come from our experimental outputs, and every figure was checked against them. Every code change suggested by these tools was inspected and tested by the authors before it was used. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

All experiments use public models and data. The dataset processing, the label thresholds, the disjoint adaptation, probe, intervention and test partitions, and the optimizer pools and seeds are described in Appendix. The adaptation objectives and insertion sites of GEPA, prompt tuning and prefix tuning are given. GEPA uses a proprietary reflector model, so its search is not exactly repeatable, and we therefore give an optimized instruction in Appendix D. We will release the code for adaptation, evaluation and all interventions, together with the trained adapters, the GEPA instructions and the learned correction maps. Anonymized code is available at `https://anonymous.4open.science/r/moe-routing-drift-5ECE`.

References

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of the 2019 World Wide Web Conference*, pp. 491–500, 2019. doi: 10.1145/3308560.3317593.

Fan L. Cheng and Nikolaus Kriegeskorte. Decomposing how prompting steers behavior, 2026. URL `https://arxiv.org/abs/2606.03093`.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3369–3391, 2022. doi: 10.18653/v1/2022.emnlp-main.222.

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.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15466–15490, 2024.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *International Conference on Learning Representations*, 2025.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. *arXiv preprint arXiv:2310.15916*, 2023.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of EMNLP-IJCNLP*, pp. 2733–2743, 2019. doi: 10.18653/v1/D19-1275.

Geert Heyman and Frederik Vandeputte. Steer like the llm: Activation steering that mimics prompting, 2026. URL https://arxiv.org/abs/2605.03907.

Artem Kirsanov, Chi-Ning Chou, Kyunghyun Cho, and SueYeon Chung. The geometry of prompting: Unveiling distinct mechanisms of task adaptation in language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1855–1888, 2025. doi: 10.18653/v1/2025.findings-naacl.100.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021. doi: 10.18653/v1/2021.emnlp-main.243.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pp. 4582–4597, 2021. doi: 10.18653/v1/2021.acl-long.353.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2, 2024. URL https://arxiv.org/abs/2408.05147.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, 2022. doi: 10.18653/v1/2022.acl-short.8.

Yilun Liu, Yunpu Ma, Yuetian Lu, Shuo Chen, Zifeng Ding, and Volker Tresp. Parameter-efficient routed fine-tuning: Mixture-of-experts demands mixture of adaptation modules. *arXiv preprint arXiv:2508.02587*, 2025.

Elita Lobo, Chirag Agarwal, and Himabindu Lakkaraju. On the impact of fine-tuning on chain-of-thought reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 11679–11698, 2025.

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*, 2024.

Matthew Lyle Olson, Neale Ratzlaff, Musashi Hinck, Man Luo, Sungduk Yu, Chendi Xue, and Vasudev Lal. Probing semantic routing in large mixture-of-expert models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 18263–18278, 2025.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025.

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. *arXiv preprint arXiv:2406.11695*, 2024.

Samet Oymak, Ankit Singh Rawat, Mahdi Soltanolkotabi, and Christos Thrampoulidis. On the role of attention in prompt-tuning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 26724–26768, 2023.

Oam Patel, Jason Wang, Nikhil Shivakumar Nayak, Suraj Srinivas, and Himabindu Lakkaraju. Towards interpretable soft prompts. *arXiv preprint arXiv:2504.02144*, 2025.

Aleksandar Petrov, Philip H. S. Torr, and Adel Bibi. When do prompting and prefix-tuning work? a theory of capabilities and limitations. *arXiv preprint arXiv:2310.19698*, 2023.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, 2023. doi: 10.18653/v1/2023.emnlp-main.494.

Enrique Queipo-de Llano, Álvaro Arroyo, Federico Barbero, Xiaowen Dong, Michael Bronstein, Yann LeCun, and Ravid Shwartz-Ziv. Attention sinks and compression valleys in LLMs are two sides of the same coin. *arXiv preprint arXiv:2510.06477*, 2025.

Dana Ramati, Daniela Gottesman, and Mor Geva. Eliciting textual descriptions from representations of continuous prompts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 16545–16562, 2025. doi: 10.18653/v1/2025.findings-acl.849.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 4222–4235, 2020. doi: 10.18653/v1/2020.emnlp-main.346.

Zunhai Su, Qingyuan Li, Hao Zhang, Weihao Ye, Qibo Xue, YuLei Qian, Yuchen Xie, Ngai Wong, and Kehong Yuan. Unveiling super experts in mixture-of-experts large language models. In *International Conference on Learning Representations*, 2026.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *Conference on Language Modeling*, 2024.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *International Conference on Learning Representations*, 2024.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Lukas Twist, Helen Yannakoudakis, and Jie M. Zhang. Reasoning-trace collapse: Evaluating the loss of explicit reasoning during fine-tuning, 2026.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35151–35174, 2023.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. OpenMoE: An early effort on open mixture-of-experts language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, volume 2024, pp. 1651–1678, 2024.

Lechen Zhang, Tolga Ergen, Lajanugen Logeswaran, Moontae Lee, and David Jurgens. SPRIG: Improving large language model performance by system prompt optimization. In *International Conference on Learning Representations*, 2026.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*, 2023.

A Numerical Results

Let Δ_{str} and Δ_{set} denote the gain over the seed under order-sensitive and order-invariant parsing. We report the fraction of the former gain that disappears under the latter evaluator:

$$\rho = \frac{\Delta_{\mathrm{str}} - \Delta_{\mathrm{set}}}{\Delta_{\mathrm{str}}}. \tag{6}$$

Both the seed and adapted scores change with the parser. Thus ρ is an operational comparison of evaluation rules, not a causal decomposition of learning into formatting and label prediction. For example, at $n = 1000$, prompt tuning scores 0.8003 and 0.8017, but its gains over the seed are 0.5166 and 0.1925 under the two rules.

For a residual intervention λv_ℓ^M, the recovered fraction of the target task-score gap is

$$R^M_{\mathrm{task}}(\ell, \lambda) = \frac{F_{\ell, \lambda v_\ell^M}(\mathcal{C}_0) - F(\mathcal{C}_0)}{F(\mathcal{C}_M) - F(\mathcal{C}_0)}, \tag{7}$$

where F is the task metric and $\mathcal{C}_0$ and $\mathcal{C}_M$ are the seed and adapted conditions.

Table 3: Held-out test performance, 3000 rows, empty-aware samples F_1 under both parsers, used in Figure 1a. The unadapted seed scores 0.2837 strict and 0.6092 set. The parser-dependent gain reduction ρ of Equation (6) is 0.85 for the discrete branch. GEPA entries average over three optimizer seeds. Continuous entries average over $m \in \{100, 200, 500\}$ and three seeds.

	discrete (GEPA)		prompt tuning		prefix tuning	
n	strict	set	strict	set	strict	set
200	0.6077	0.6604	0.7192	0.7336	0.7443	0.7445
500	0.5888	0.6526	0.7662	0.7717	0.7951	0.7952
1000	0.6190	0.6598	0.8003	0.8017	0.8117	0.8118

Table 4: Recovered fraction R_{task} of Equation (7), aggregated over 12 cells per branch, against a norm-matched random direction injected at the same site, used in Figure 3.

branch	R [95% CI]	norm-matched random
prefix tuning ($m = 500$)	$+0.019$ $[-0.009, +0.052]$	$+0.019$ $[-0.004, +0.043]$
discrete (GEPA)	$+0.023$ $[-0.117, +0.174]$	$+0.024$ $[-0.083, +0.127]$

B Additional Dense Diagnostics

This section reports the diagnostics of Sections 3.1 and 3.2 and the adapters, partitions and interventions they use.

B.1 Adaptation Objectives and Insertion Sites

Each method optimizes an auxiliary object z with the backbone θ frozen. For dataset $\mathcal{D}$ and method-specific feasible set $\mathcal{Z}_M$,

$$z_M^\star \in \arg\min_{z \in \mathcal{Z}_M} \mathcal{J}_M(z; \mathcal{D}), \tag{8}$$

where $\mathcal{J}_M$ is the autoregressive sequence loss for continuous methods and the negative black-box evaluation score for GEPA. GEPA searches textual instructions through reflection and evolutionary selection (Agrawal et al., 2026). Prompt tuning learns input embeddings $P \in \mathbb{R}^{m \times d}$ (Lester et al., 2021), and prefix tuning learns keys and values $\{K_\ell^P, V_\ell^P\}_{\ell=0}^L$ for every attention block (Li & Liang, 2021). The insertion sites are

$$H_0^M(x) = \begin{cases} E([z_{\mathrm{GEPA}}; \mathcal{C}(x)]), & M = \mathrm{GEPA}, \\ [P; E(\mathcal{C}(x))], & M = \mathrm{prompt}, \\ E(\mathcal{C}(x)), & M = \mathrm{prefix}, \end{cases} \tag{9}$$

$$\mathrm{Attn}_\ell^{\mathrm{prefix}} = \mathrm{softmax}\left(\frac{Q_\ell [K_\ell^P; K_\ell]^\top}{\sqrt{d_k}} + \mathcal{M}_\ell \right) [V_\ell^P; V_\ell].$$

Here $\mathcal{C}(x)$ is the common scaffold, Q_ℓ, K_ℓ and V_ℓ are the content projections, and $\mathcal{M}_\ell$ extends the causal mask with unmasked prefix columns. Text and prompt embeddings enter at the input, while prefixes supply learned keys and values at every layer. These parameterizations provide different ways to condition the frozen model, as studied in theoretical analyses of continuous prompting (Oymak et al., 2023; Petrov et al., 2023).

For the gradient-trained arms and target sequence $y = (y_1, \ldots, y_T)$, the loss in Equation (8) is the autoregressive negative log-likelihood

$$\mathcal{L}_{\mathrm{NLL}}(x, y; z) = -\sum_{t=1}^{T} \log p_\theta(y_t \mid y_{<t}, x; z). \tag{10}$$

Prompt tuning optimizes $P \in \mathbb{R}^{m \times d}$ and concatenates it with the embedded textual scaffold only once. Prefix tuning optimizes a separate block of m keys and values for every attention layer. These are concatenated with the content keys and values as in Equation (9), but do not enter the residual stream as token positions. GEPA instead treats the task score as a black-box objective and may rewrite only the natural-language instruction. The label list, input wrapper, and output contract remain fixed in every arm.

B.2 Diagnostic Setup

The backbone is `google/gemma-2-2b-it`, with 26 transformer blocks and hidden size 2304. Civil Comments scores are thresholded at 0.5 for five non-exclusive labels: `toxicity`, `obscene`, `threat`, `insult`, and `identity_attack`. Empty-aware samples-F_1 assigns one when the prediction and target sets are both empty. About one third of each split has an empty target set.

The adaptation, probe, intervention and 3000-example test partitions are disjoint. The performance sweep uses optimizer pools $n \in \{200, 500, 1000\}$ and three seeds. GEPA has a budget of at most 10^4 metric calls and uses `gpt-5.6-luna-pro` as its reflector. Prompt and prefix tuning use the same optimizer pools and $m \in \{20, 100, 200, 500\}$ virtual tokens. Table 3 averages over $m \in \{100, 200, 500\}$. The final test set is not used to train probes or select adapted prompts.

The length-matched padded prompt $\mathcal{C}_{\mathrm{pad}}$ adds context without adaptation. We compare an internal measurement under the adapted and padded contexts before attributing it to adaptation. A change shared with padding can arise from adding context positions alone.

In Table 3, the continuous methods improve monotonically with optimizer pool size, while GEPA does not. Averaged over the three pool sizes, the prefix lead over GEPA is 0.179 under order-

sensitive parsing and 0.126 under set parsing. Ignoring output order reduces this measured gap by about 30%, an evaluator-dependent comparison rather than a causal decomposition of the gain. Appendix D gives an example of the optimized GEPA instruction from this sweep.

Mean displacement and the padding control. For each method, the one-shot intervention uses the mean displacement v_ℓ^M at the generation anchor τ defined in Equation (2). Subsequent positions receive no direct intervention in this setting. We measure the fraction of displacement energy captured by the mean,

$$\eta_\ell^M = \frac{\|v_\ell^M\|_2^2}{N^{-1}\sum_i \|\delta_{\ell,\tau}^M(x_i)\|_2^2}. \tag{11}$$

At layer 6, this fraction is 0.961 for GEPA, 0.981 for prefix tuning, 0.967 for prompt tuning, and 0.948 for padding. At layer 18 the respective values are 0.636, 0.676, 0.569, and 0.484. A large shared displacement therefore also occurs under a context change that does not improve the task. The one-shot steering sweep covers four layers and three seeds per branch. The reported relative-norm scale settings are $\{0.05, 0.1, 0.2\}$, measured against the state norm at the injection site. These settings specify the perturbation size, while λ in Equation (2) denotes the multiplier of the mean displacement. Norm-matched random directions are injected at the same sites. Table 4 reports the aggregated recovery. The prefix interval is narrow around a small effect, while the GEPA interval is wide because its target gap is small. The reported recoveries for both branches are similar to those of their random controls.

B.3 Tuned Lens and Linear Probes

We measure label decodability with a linear probe (Hewitt & Liang, 2019) and the output distribution accessible through a calibrated tuned lens (Belrose et al., 2023). Their readouts are

$$p_\ell^{\mathrm{TL}}(x) = \mathrm{softmax}(W_U\,\mathrm{LN}(A_\ell h_{\ell,\tau}(x) + b_\ell)), \qquad \widehat{y}_{c,\ell}(x) = \sigma\big(w_{c,\ell}^\top h_{\ell,\tau}(x) + \beta_{c,\ell}\big). \tag{12}$$

At each layer, the affine lens translator minimizes the expected KL from the model's final output distribution to p_ℓ^{TL}. The regularized logistic probes use 8000 training and 2000 validation examples, extracting the residual state at the last prompt position. These auxiliary readouts measure available information, without identifying the computation used by the model's own output head.

For the tuned lens, we measure the margin between present and absent labels,

$$\mu_\ell(x) = \frac{1}{|G|}\sum_{c\in G}\log p_\ell^{\mathrm{TL}}(c) - \frac{1}{|\bar{G}|}\sum_{c\in\bar{G}}\log p_\ell^{\mathrm{TL}}(c), \tag{13}$$

where G is the gold label set. We average only over examples with $0 < |G| < 5$, for which both terms are defined. GEPA remains above the seed from layer 20 in every evaluated configuration. Prompt tuning crosses at about layer 18, while prefix tuning crosses between layers 4 and 20 (Figure 2a).

Under the seed prompt, probe validation samples-F_1 rises from 0.464 at the first layer to about 0.75 by layer 10 and peaks at 0.7855 in layer 18 (Figure 1b). A seed-trained probe scores between 0.77 and 0.83 on seed states but between 0.16 and 0.73 on adapted states. Retraining within each adapted stream restores scores between 0.74 and 0.80. Length-matched padding similarly degrades probe transfer. Failed transfer therefore does not, by itself, identify an adaptation-specific mechanism.

B.4 Attention to the Adapted Context

For head-averaged attention $\alpha_\ell(t, j)$ from the read position t, we sum over virtual-token positions to measure attention to continuous contexts. For a textual instruction, we plot cumulative mass $M_\ell(k) = \sum_{j\le k}\alpha_\ell(t, j)$ along its positions. The prompt diagnostic uses $m = 20$, $n = 20000$, and seed 43. Its virtual-token mass peaks at 0.252 in layer 7 and falls to 0.021 at the final layer. The prefix diagnostic uses $m = 500$, $n = 20000$, and seed 44. Its virtual-token mass rises from 0.523 to 0.940 and remains between 0.90 and 0.95 during generation. Its attention mass on the text instruction falls to 0.033. The models differ in seed and capacity, so these values do not isolate an effect of adaptation method.

The optimized GEPA instruction contains about 1850 tokens. At layers 12, 19 and 25, cumulative attention grows mainly over the opening task statement and closing format rule, with little growth

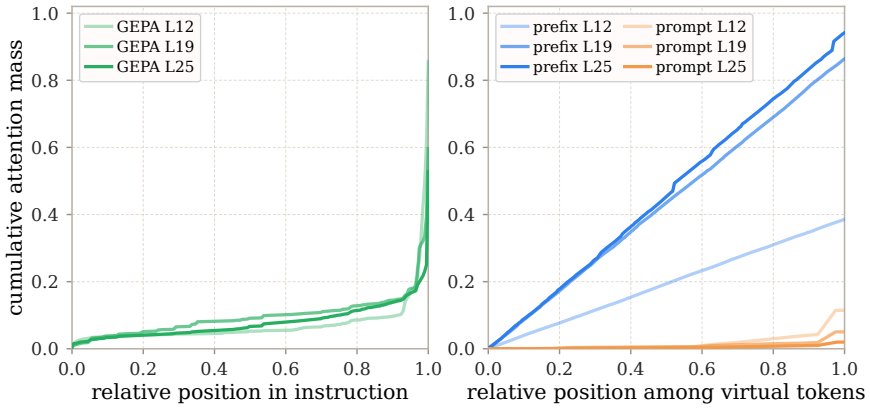
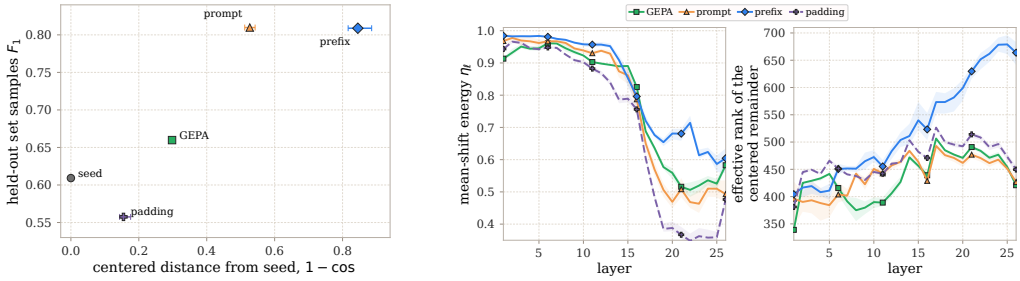


Figure 6: Representative access profiles for the adapted contexts. Curves show cumulative attention within the GEPA instruction (left) and learned continuous contexts (right) at selected blocks.



(a) Movement does not order task quality.

(b) Mean energy and rank under adaptation and padding.

Figure 7: A matched context placebo separates representation drift from task utility. Padding reproduces prominent geometric signatures without improving the task.

over the intervening label criteria (Figure 6). These profiles are descriptive. Masking access to either continuous context returns subset accuracy to the zero-shot level, providing the intervention test reported in the main text. For GEPA, masking the instruction lowers subset accuracy slightly below the seed level (Figure 2b).

B.5 Mean Shifts and Cosine Geometry

We analyze the diagnostic states at the final prompt position on $N = 1000$ fixed examples. These measurements concern the observed seed-relative displacements, rather than the learned predictor matrices in Appendix C.4. After subtracting the dataset mean in Equation (11), we measure the effective rank of the remaining variation. With $d = 2304$ and $N = 1000$, the centered matrix has rank at most 999.

At layer 18, the centered remainder has effective rank between 476 and 573 across adaptation methods, compared with 500 for padding (Figure 7b). Both the large mean component and the broad remaining variation therefore occur under a context change that does not improve the task.

The pairwise mean can hide how shift directions vary across inputs. We retain this variation with

$$\kappa_{\ell}^{A,B}(x) = \frac{\langle \delta_{\ell,\tau}^{A}(x), \delta_{\ell,\tau}^{B}(x) \rangle}{\|\delta_{\ell,\tau}^{A}(x)\|_2 \, \|\delta_{\ell,\tau}^{B}(x)\|_2}. \tag{14}$$

At block 25, mean per-example cosine is 0.011 for GEPA and prefix, 0.238 for prefix and prompt, and 0.327 for GEPA and prompt. The last pair has the broadest distribution (Figure 8). These means use the diagnostic adapters of Appendix B.2. Appendix C.4 reports the corresponding medians for the adapters of the predictor experiment.

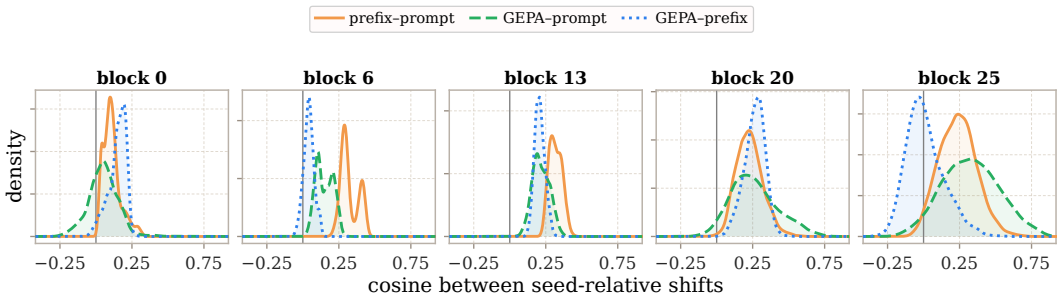


Figure 8: Per-example cosine between seed-relative shifts from two adaptation methods. Each panel pools 1000 fixed examples. Early-layer shifts have small norms, so their directions are less stable.

Table 5: Mean per-example cosine similarity with the seed-prompt residual at layer 18 over three seeds. Centered states have their dataset mean removed.

condition	raw	centered
GEPA	0.902	0.702
prompt tuning ($m = 500$)	0.830	0.473
prefix tuning ($m = 500$)	0.497	0.155
$\mathcal{C}_{\text{pad}}$	0.961	0.845

Subtracting each condition's dataset mean reduces its cosine similarity with the seed (Table 5). GEPA remains more aligned with the seed than either continuous method. Prefix tuning has the lowest centered cosine, 0.155, while padding remains much closer at 0.845.

Figure 7a compares this distance with task quality. Padding stays close to the seed but performs worse. GEPA improves while moving less than either continuous method. The geometric distance alone therefore does not identify a useful intervention.

B.6 SAE Fidelity and Cross-Method State Patching

We apply Gemma Scope SAEs (Lieberum et al., 2024) to residual states at real token positions. Learned input embeddings and prefix keys and values do not share these residual coordinates. On 500 test examples, replacing one selected state by its SAE reconstruction changes samples-F_1 by at most 0.014. Output agreement, averaged over ten evaluated layers, ranges from 0.873 to 0.975 across the evaluated conditions. This checks the reconstruction's behavioral fidelity before using it to represent a displacement.

We also measure how well the SAE reconstructs the displacement itself. For method M relative to the seed, the dense displacement, decoded displacement and recovered energy are

$$\begin{aligned} \delta^M_{\ell,\tau}(x) &= h^M_{\ell,\tau}(x) - h^0_{\ell,\tau}(x), \\ \widehat{\delta}^M_{\ell,\tau}(x) &= \mathrm{Dec}_\ell\big(\mathrm{Enc}_\ell(h^M_{\ell,\tau}(x))\big) - \mathrm{Dec}_\ell\big(\mathrm{Enc}_\ell(h^0_{\ell,\tau}(x))\big)\,, \\ q^M_\ell &= 1 - \frac{\sum_x \|\delta^M_{\ell,\tau}(x) - \widehat{\delta}^M_{\ell,\tau}(x)\|_2^2}{\sum_x \|\delta^M_{\ell,\tau}(x)\|_2^2}. \end{aligned} \tag{15}$$

In the matched $n = 1000$, $m = 20$ comparison, q_{25} is 0.421 for prompt tuning, 0.582 for prefix tuning and 0.616 for GEPA. Recovered energy can be negative in early layers, where reconstruction error exceeds the energy of the displacement. We therefore restrict this analysis in SAE coordinates to late layers. Recovered energy measures state reconstruction, while the following intervention measures transfer of the output distribution.

For a source condition S and a target condition T, the exact patch adds $h^T_{\ell,\tau}(x) - h^S_{\ell,\tau}(x)$ to the source state, as defined in the main text. Its SAE counterpart is

$$\widehat{\delta}^{S\to T}_{\ell,\tau}(x) = \mathrm{Dec}_\ell\big(\mathrm{Enc}_\ell(h^T_{\ell,\tau}(x))\big) - \mathrm{Dec}_\ell\big(\mathrm{Enc}_\ell(h^S_{\ell,\tau}(x))\big)\,. \tag{16}$$

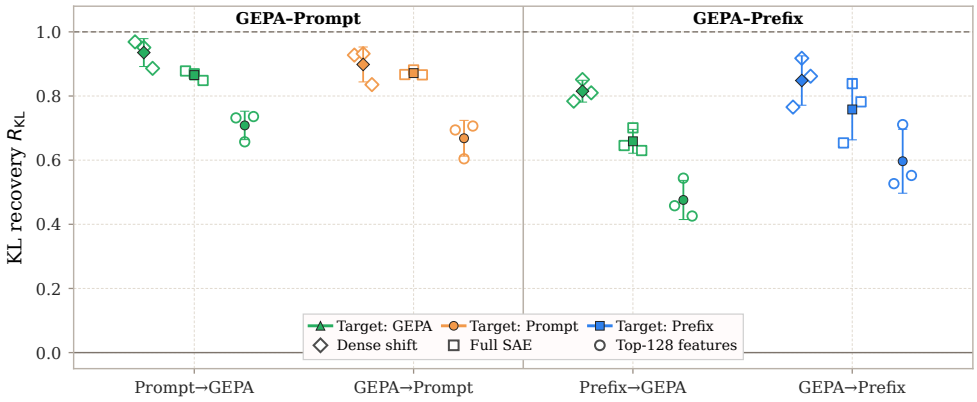


Figure 9: Exact-state, SAE-reconstructed, and Top-128 feature transfer. The Top-128 subset is selected by displacement magnitude on disjoint calibration examples.

The patch is applied at the final non-padding prompt position, and the remaining source blocks run unchanged (Zhang & Nanda, 2024). It therefore concerns the first output distribution, before either model has generated an answer. We report the fraction of the next-token KL gap closed by each patch using R_{KL} from Equation (3). The bidirectional GEPA/prompt comparison recovers 0.917 ± 0.049 with exact states and 0.868 ± 0.012 through the SAE. The corresponding GEPA/prefix values are 0.832 ± 0.056 and 0.708 ± 0.084. These comparisons use continuous adapters trained with $n = 20000, m = 20$ and GEPA prompts trained with $n = 1000$. They describe state transfer between the selected models, rather than a controlled estimate of adaptation-method effects.

Selecting the 128 most changed SAE features on 32 disjoint calibration examples recovers 0.688 of the GEPA/prompt gap and 0.536 of the GEPA/prefix gap. This feature subset is selected by change magnitude. Its transfer score does not identify the semantics of the selected features (Figure 9).

C Shift Predictor Details

C.1 MSE and MSE+KL on 3000 Test Examples

The new evaluation compares state MSE with MSE plus an output KL term at the final block, in two regimes. Repredict computes a new shift at each generated position. Fixed computes one shift at the generation anchor and reuses it. Fixed-regime details and a low-rank ablation are in Appendix C.2. Appendix C.5 reports an earlier experiment with additional SAE losses. The manual seed-model samples-F_1 on these 3000 examples is 0.617. Block 25 is the final transformer block.

Design and objective. We evaluate 36 runs spanning two regimes, three adaptation targets, two objectives, and seeds $42, 43, 44$. Each predictor is an affine map with 5,310,720 parameters. Training pairs come from the adapted model's own greedy answers, on which both the seed and the adapted model are run. All runs use the same 3000 test examples. Reported dispersion is sample standard deviation across the three seeds.

The configured predictor splits contain 4096 training and 256 validation examples. These selections and the test share no IDs or exact texts with one another or with the adaptation optimizer pools. The state-only objective is

$$\mathcal{L}^{M}_{\mathrm{state}} = \mathbb{E}_{(x,t)}\left[\frac{1}{d}\left\| f^{M}_{\phi}(h^{0}_{L,t}) - \delta^{M}_{L,t}(x)\right\|^{2}_{2}\right]. \tag{17}$$

Each active objective term is normalized by its value for the training mean-shift baseline. With

$$\mathcal{L}^{M}_{\mathrm{KL}} = \mathbb{E}_{(x,t)}\,\mathrm{KL}\big(p(h^{M}_{L,t}) \,\big\|\, p\big(h^{0}_{L,t} + f^{M}_{\phi}(h^{0}_{L,t})\big)\big)\,, \tag{18}$$

Table 6: Predicted final-state corrections at block 25 during free generation on 3000 test examples. Entries are mean and sample SD over adaptation seeds $42, 43, 44$. Native denotes the adapted model. This predictor suite reports a separately evaluated seed baseline of 0.617; the performance sweep in Table 3 reports 0.6092, so we report raw scores rather than combine the two baselines. Target adapters use $n = 1000$, with $m = 20$ virtual tokens for the continuous methods. The score is empty-aware set samples-F_1, with invalid outputs assigned zero.

		Repredict		Fixed	
Target	Native	MSE	MSE+KL	MSE	MSE+KL
GEPA	$0.678 \pm .007$	$0.683 \pm .015$	$0.681 \pm .012$	$0.638 \pm .008$	$0.683 \pm .011$
Prompt	$0.777 \pm .013$	$0.746 \pm .001$	$0.759 \pm .006$	$0.637 \pm .015$	$0.756 \pm .003$
Prefix	$0.805 \pm .007$	$0.744 \pm .004$	$0.765 \pm .006$	$0.506 \pm .121$	$0.755 \pm .007$

where $p(h)$ applies the frozen final normalization and output head, including the model's logit soft-capping, followed by a softmax, the combined objective is

$$\mathcal{L}_{\text{MSE+KL}} = \frac{\mathcal{L}^M_{\text{state}}}{s_{\text{state}}} + \frac{\mathcal{L}^M_{\text{KL}}}{s_{\text{KL}}}. \tag{19}$$

Neither objective uses SAE features.

The MSE-only runs select the last checkpoint at step 800. The reported values therefore characterize the training budget used here, without establishing convergence.

Metric. We report empty-aware set samples-F_1, assigning zero to invalid outputs. The parser accepts comma-separated allowed labels, ignores their order and removes duplicates. It accepts `NONE` for an empty prediction. A valid empty prediction against an empty gold set receives one. Like the set parser of Table 3, this metric ignores label order. Unlike that parser, it scores an unparseable answer zero instead of treating it as an empty prediction. The continuous adapters here use $m = 20$ virtual tokens, whereas the continuous entries of Table 3 average over $m \in \{100, 200, 500\}$.

Constant-shift control. For an affine predictor $Wh+b$, initialization at $W = 0$ and the train mean shift does not establish that the learned gain comes from Wh, since b can also change. We therefore train a bias-only predictor at block 25 with the MSE+KL objective and evaluate it in the repredict regime on the same 3000 test examples. It reaches $0.633 \pm .005$, $0.604 \pm .006$ and $0.543 \pm .057$ for GEPA, prompt and prefix, against $0.681 \pm .012$, $0.759 \pm .006$ and $0.765 \pm .006$ for the affine map. Its invalid-answer rates are 1.6%, 6.0% and 18.4%. For prompt and prefix tuning the constant shift stays below the seed score of 0.617, so their recovered gains come from the input-dependent term. The low-rank sweep in Appendix C.2 tests the dimension of the input-dependent part.

C.2 Fixed Regime and Predictor Rank

Fixed and repredict predictors are trained separately and have different learned matrices. At inference, a fixed predictor computes the shift once from the seed state at the generation anchor and adds it at every generated position. Scores therefore compare the resulting trained systems, rather than two inference modes of one fitted predictor. With MSE, the fixed regime leaves 1.9%, 6.8% and 30.9% of GEPA, prompt and prefix answers invalid, and the prefix score varies strongly across seeds. With MSE+KL these rates are 0.6%, 0.1% and 0.1% (Table 6).

The low-rank predictor is $f^M_\phi(h) = U_M V_M^\top h + b_M$ with $U_M, V_M \in \mathbb{R}^{d\times r}$, initialized like the full predictor with the train mean shift as bias. Table 7 reports one training seed per cell at the final block, trained with MSE and evaluated in both regimes. The full-rank column repeats the seed-42 run of the same configuration. These values describe a single seed and carry no uncertainty estimate. Rank one restricts the input-dependent term to a single direction, giving 0.634, 0.596 and 0.677 for GEPA, prompt and prefix. This resembles the single-direction design of prompt steering replacement models (Heyman & Vandeputte, 2026), but our affine correction includes a vector bias and does not use their ReLU gate. It is a rank ablation of our predictor, not a reproduction of PSR.

Table 7: Task score by predictor rank, seed 42, MSE objective, 3000 test examples. The metric matches Table 6, and the manual baseline is 0.617. All predictors operate at block 25.

Regime	Target	$r = 1$	$r = 2$	$r = 4$	$r = 8$	full
Repredict	GEPA	0.634	0.640	0.645	0.660	0.683
	Prompt	0.596	0.655	0.664	0.699	0.746
	Prefix	0.677	0.683	0.688	0.700	0.741
Fixed	GEPA	0.635	0.635	0.636	0.640	0.641
	Prompt	0.580	0.577	0.595	0.608	0.619
	Prefix	0.353	0.346	0.371	0.391	0.389

Under repredict, the score increases with rank for every target, while rank 8 remains between 0.023 and 0.047 below the full map. Under the fixed regime the spread across ranks is at most 0.045, and the fixed prefix predictor stays far below the seed baseline at every rank.

C.3 Early Exit

The early-exit model runs only blocks 0 to 20 of the seed-context model. After block 20 it forms

$$\widehat{h}^{M}_{25,t} = h^{0}_{20,t} + f^{M}_{\phi}(h^{0}_{20,t}) \tag{20}$$

and applies the frozen final normalization and output head. Blocks 21 to 25 are not executed. The map f^{M}_{ϕ} is a full-rank affine map with 5,310,720 parameters, trained with the MSE+KL objective of Equation (19) with the adapted block-25 state as target. It is applied at the prompt and at every generated position, as in the repredict regime. Table 8 compares it with the block-25 corrections of Table 6 on the same 3000 test examples. Fitting the same map to the seed model's own block-25 state gives 0.618, against 0.619 for the full seed model in the same evaluation (seed 42). For this task the last five blocks can thus be replaced by an affine map even without adaptation.

Table 8: Early exit from block 20 against the block-25 correction, 3000 test examples, mean and sample SD over seeds $42, 43, 44$. The metric matches Table 6.

Target	Native	Block-25 correction	Early exit
GEPA	$0.678 \pm .007$	$0.681 \pm .012$	$0.688 \pm .011$
Prompt	$0.777 \pm .013$	$0.759 \pm .006$	$0.766 \pm .012$
Prefix	$0.805 \pm .007$	$0.765 \pm .006$	$0.770 \pm .012$

Latency. We time generation on the first 100 test examples, batch size 16, one warm-up batch and two timed repeats, excluding model loading. Every condition decodes exactly 16 tokens, so the comparison does not depend on answer length. Table 9 reports the mean time per example. Early exit takes about 43 ms for all three targets, against 52.6 ms for the full seed model, a $1.22\times$ speedup close to the share of skipped blocks. The speedup over native prompt and prefix tuning is $1.28\times$ and $1.26\times$. Native GEPA is $7.1\times$ slower than early exit, mostly because its instruction has 1684 prompt tokens per example against 158 for the seed. This timing run is separate from the quality evaluation and uses a single seed.

Table 9: Time per example with 16 generated tokens, 100 test examples, batch size 16, mean over two repeats.

Condition	Native (ms)	Early exit (ms)
Seed	52.6	–
GEPA	303.7	43.0
Prompt	55.3	43.0
Prefix	54.3	43.1

C.4 Geometry of the Learned Corrections

We compare the full-rank repredict predictors across methods. For two predictors we compute the cosine between their flattened matrices W_M. Cross-method entries pair predictors with the same seed and average over seeds $42, 43, 44$. Within-method entries average over the three seed pairs of one method.

Table 10: Cosine between flattened block-25 predictor matrices W_M (repredict, full rank).

Pair	MSE	MSE+KL
GEPA, prompt	0.32	0.29
GEPA, prefix	0.27	0.24
Prompt, prefix	0.49	0.57
GEPA seeds	0.72	0.67
Prompt seeds	0.67	0.76
Prefix seeds	0.61	0.68

The sample-level analysis uses block 25, MSE and seed 42, the configuration for which trajectory caches are available. The repredict predictions have mean cosines of 0.935, 0.942 and 0.909 with the true shifts for GEPA, prompt and prefix tuning. On the first generated token, the median cosine between true shifts of two methods is 0.18, 0.20 and 0.46 for the pairs in Table 10, and the predicted shifts reproduce these values within 0.05. Later tokens are generated from different histories under different methods, and the continuous methods produce only three to four tokens, so the comparison is restricted to the first steps.

The mean direction carries 58.6%, 58.9% and 51.5% of the shift energy for GEPA, prompt and prefix tuning, the fraction η_ℓ defined in Equation (11). The mean directions of different methods have cosines of 0.09, 0.13 and 0.38. In the repredict predictors, the input-dependent term $W_M h$ has mean norms of 180, 264 and 315, against bias norms of 157, 212 and 365. These norm comparisons do not establish that the bias or input-dependent term is sufficient on its own; the matched controls in the main-text limitation remain necessary.

C.5 Earlier Composite-Loss Experiment

Before the MSE ablation, we trained predictors with a composite loss that includes SAE terms and evaluated them on 512 held-out examples.

Data and predictors. For each adapted model and seed, teacher-forced trajectories of the native answer pair the seed-model state $h^0_{25,t}$ with the adapted state $h^M_{25,t}$ at the same position. Training uses 4096 trajectories and validation 256. Evaluation uses 512 held-out examples and three adaptation seeds. Generation is greedy with at most 64 new tokens. The seed-model samples-F_1 is 0.624.

The affine predictor is $f_\phi(h) = Wh + b$, initialized with $W = 0$ and b equal to the training mean shift. The second predictor is a GELU MLP with hidden width 4608. Their parameter counts are approximately 5.3M and 21M, respectively. Neither is a small linear probe.

Application modes. Let τ be the generation anchor. One-shot predicts the shift from $h^0_{25,\tau}$ and applies it only at the anchor. Fixed predicts from the same state and reuses that vector at each decoded position. Repredict computes a new shift from the current state at each step. Both fixed and repredict are input-dependent. Their scores compare corrections reused across positions with corrections recomputed at each position. They do not test whether conditioning on the example is necessary.

Objective. The earlier loss is

$$\mathcal{L} = \frac{\mathcal{L}_{\mathrm{DENSE}}}{s_{\mathrm{DENSE}}} + \frac{\mathcal{L}_{\mathrm{ENC}}}{s_{\mathrm{ENC}}} + \frac{\mathcal{L}_{\mathrm{DEC}}}{s_{\mathrm{DEC}}} + \frac{\mathcal{L}_{\mathrm{KL}}}{s_{\mathrm{KL}}}, \tag{21}$$

where $\mathcal{L}_{\mathrm{DENSE}}$ is state MSE, and $\mathcal{L}_{\mathrm{ENC}}$ and $\mathcal{L}_{\mathrm{DEC}}$ are the corresponding errors in the encoder and decoder space of a Gemma Scope layer-25 SAE with 16k latents. The KL term compares adapted

Table 11: Strict samples-F_1 of the seed model patched at block 25 with a predicted shift, mean $\pm$ SD over three adaptation seeds on the same 512 test examples. Native is the adapted model itself. The seed model scores 0.624.

Method	Predictor	Native	One-shot	Fixed	Repredict
GEPA	Linear	$0.684 \pm .004$	$0.641 \pm .009$	$0.694 \pm .007$	$0.690 \pm .008$
	MLP		$0.630 \pm .009$	$0.695 \pm .018$	$0.692 \pm .009$
Prompt	Linear	$0.838 \pm .004$	$0.663 \pm .004$	$0.729 \pm .012$	$0.761 \pm .010$
	MLP		$0.648 \pm .005$	$0.747 \pm .005$	$0.744 \pm .009$
Prefix	Linear	$0.851 \pm .006$	$0.657 \pm .007$	$0.755 \pm .005$	$0.765 \pm .007$
	MLP		$0.637 \pm .009$	$0.757 \pm .004$	$0.754 \pm .011$

and patched output distributions through the frozen final norm, unembedding and logit soft-capping, with the adapted distribution as its first argument. Each s_k is the value of that term for the mean-shift baseline on the training split. All trajectory positions enter the loss. In one-shot training only the anchor receives a gradient, so objective values are not comparable with the recurrent regimes. Checkpoints are selected by validation objective.

In this earlier experiment, recurrent scores are higher than one-shot scores (Table 11). The largest difference between affine and MLP predictors in a recurrent cell is 0.018. The largest fixed-versus-repredict difference is 0.032. These observed differences do not establish equivalence. The MSE results in Table 6 show that transfer is possible without SAE supervision.

C.6 Recurrent Mean Shift

For prefix tuning, we add the dataset-mean shift at every decoded position on 500 intervention-validation examples, choosing its layer and scale on the same examples. With the selected configuration, samples-F_1 rises from 0.606 to 0.620, 0.625 and 0.627 for seeds 42, 43 and 44, while the adapted prefix models score 0.818, 0.800 and 0.811. This recovers 6.7%, 10.0% and 10.0% of the prefix gain.

At block 25, adding the raw train mean with scale one lowers task score below the seed for all six tested adapted models.

D GEPA Prompt Examples

Inspection of the frozen instructions in the GEPA sweep found that examples from GEPA's validation data were sometimes copied directly into the prompt. This observation concerns the optimizer's validation examples, not evidence of overlap with the held-out test set. Below is the full $n = 200$ instruction reproduced in the workshop draft, with identity terms in the quoted comments masked by asterisks. The quoted instruction is preserved verbatim.

GEPA with $n = 200$.

> Classify the supplied text for abusive or harmful language. Determine every label supported by the text itself, using context and the actual target. Do not infer intent or add labels merely because a statement is political, controversial, negative, or mentions a protected group.
>
> Apply labels independently:
>
> - toxicity: Use for hostile, contemptuous, demeaning, strongly antagonistic, or abusive rhetoric. This is the broad label for toxic language and often co-occurs with insult. Sarcasm, ridicule, dismissive mockery, and hostile political rhetoric can be toxic even without profanity, a threat, or a protected-identity target.
> - obscene: Use only for profanity, vulgarity, sexually explicit language, or clearly obscene wording. Rudeness, political aggression, or a non-obscene mention of sex or identity is not enough.
> - threat: Use only when the text expresses, warns of, or explicitly advocates violence, harm, or other serious retaliation against a person or group. Do not infer a threat

from anger, condemnation, predictions of punishment, saying someone deserves to suffer, proposing legal consequences, or hostile language without an actual intention or warning of harm.

- insult: Use for a direct or clearly implied derogatory characterization of a person, group, audience, institution, or their conduct, including name-calling, belittling comparisons, ridicule, or commands to leave. Words such as "stupid," "ridiculous," "childish," "incompetent," "over his head," "do nothing," or similar contemptuous descriptions can qualify when directed at a person, group, argument, or institution. A hostile political characterization can therefore receive both toxicity and insult. Mere disagreement, factual criticism, or a negative policy assessment without belittling language is not automatically an insult.
- identity_attack: Use only when hostility, degradation, exclusion, dehumanization, or denial of equal acceptance/treatment is directed at a protected identity or explicitly because of that identity (for example race, ethnicity, nationality, religion, sex, gender identity, or sexual orientation). Mentioning an identity, discussing discrimination, or criticizing beliefs, conduct, institutions, or policies is not by itself an identity attack. Calling someone "******," "****-*******," or similar is generally an accusation about conduct or beliefs, not an attack on the person's protected identity. Political affiliation and ideological disagreement are not protected-identity attacks.

Decision guidance:

1. Judge the wording and target, not whether the underlying political or factual claim is correct.
2. Broad or indirect hostile rhetoric can still be toxic. Political mockery aimed at a politician, party, legislature, or audience is not exempt.
3. Use insult when the wording belittles or ridicules a discernible target. An insult may target an argument or institution as well as an individual.
4. Toxicity and insult commonly apply together when contemptuous language is used. Toxicity may apply without insult when the hostility is general or lacks a clear derogatory characterization.
5. Do not label threat unless there is an actual threat, explicit call for harm, or expressed intention/warning of serious retaliation.
6. Do not label identity_attack merely because protected-category terms appear, or because the text accuses someone of prejudice.
7. Do not overread mild sarcasm, institutional criticism, or a hypothetical policy proposal as abuse when it contains no clearly degrading or hostile characterization of a target.

Calibration:

- "Stupid publicity stunt by another do nothing legislature." → toxicity and insult: "stupid" and "do nothing" are contemptuous characterizations of the legislature.
- "The idea ... is ridiculous. Anyone who uses this fallacious argument makes themselves look stupid." → toxicity and insult: the argument and its users are belittled.
- "I would have thought that a new Brian Mulroney wing at a federal penitentiary (for wayward politicians) ... would be more appropriate. I'm certain that it would fill up fast." → no labels: sarcastic political commentary/hypothetical institutional criticism, without sufficiently direct abusive language under this task's calibration.
- "There is no scientific evidence ... That is to be expected since there is none." → no labels: criticism or assertion without abusive language.
- "Trump is in way over his head ... his childish bluster." → toxicity and insult: direct political mockery and belittling characterization.
- "As long as you're a ****** ... He deserves to suffer and no jury would deny him that." → toxicity only unless other wording independently supplies a label: "******" is an accusation, and wishing or predicting suffering/legal punishment is not by itself a threat.
- "BDS ... ****-******* and ****** ... crawl back in your cave." → toxicity and insult; the accusations alone are not identity_attack, while "crawl back in your cave" is an insulting command.

Return only the labels selected by the fixed output contract; if none apply, return NONE. Never provide an explanation.

E MIXTURE OF EXPERTS

E.1 SETUP IN FULL

Data and metric. Civil Comments (Borkan et al., 2019), formulated as a multi-label choice among toxicity, obscene, threat, insult and identity attack. The test set is 2000 rows, scored by set-level F_1 averaged over rows. A third of the rows carry no label, and a model that answers nothing scores 0.330 on them. This floor is quoted throughout, and any cell sitting on it is a cell whose answers must be read before its score is used.

Routing maps. Logged on 500 validation rows, disjoint from the test set, so that the mask built from a map is never selected on the data it is evaluated on. Counts are kept per layer, per expert and per stage of the sequence: chat template and instruction, comment, virtual tokens, reasoning and answer. Figure 4 and the displacement numbers of Section 4.1 use the comment stage, the only one every arm shares.

Load statistics. For layer ℓ, $u_{\ell,e}$ ($\tilde{u}_{\ell,e}$ for an adapted arm) is the share of the layer's top-k assignments that expert e receives, summed over the 500 calibration rows and over prefill and decoding. The Gini coefficient is taken over all n_E shares of the layer, experts with no assignment included. The busiest expert's share is $\max_e u_{\ell,e}$, and the routing displacement of an adapted arm in layer ℓ is

$$\mathrm{TV}_\ell = \tfrac{1}{2} \sum_{e=1}^{n_E} \left| \tilde{u}_{\ell,e} - u_{\ell,e} \right| \in [0, 1]. \tag{22}$$

A dead expert is one that receives no assignment in the layer over all 500 rows. Table 19 reports the median over layers for the Gini and the busiest share and the mean over layers for the number of dead experts.

Depth profile of the displacement. Averaged over the first and the last quarter of the layers, TV_ℓ falls from 0.11–0.17 to 0.04–0.08 for both continuous methods on Qwen and from 0.09–0.12 to 0.06 for prompt tuning on gpt-oss, at all three budgets, while GEPA stays at or below 0.06 on both backbones. The gpt-oss prefix rises from 0.10–0.15 to 0.23–0.28. The odd layers of gpt-oss use dense attention and the even ones a 128-token band; the prefix displaces the dense layers by 0.18–0.19 on average against 0.12–0.13 for the banded ones, whereas prompt tuning (0.09–0.10 against 0.08–0.11) and GEPA (0.05 for both) do not separate them. In the last quarter, the prefix displacement $\tilde{u}_\ell - u_\ell$ has cosine 0.45–0.61 with the base's shift from comment to reasoning tokens, $u_\ell^{\text{reasoning}} - u_\ell$, and 0.35–0.44 with its shift to answer tokens, both taken from the stage-resolved maps. For prompt tuning these cosines lie between -0.16 and 0.16, and for GEPA they are negative.

Decoding. Greedy on both backbones, no temperature and no nucleus sampling, so a difference between two arms cannot be sampling noise. Token budgets are 512 for Qwen and 1024 for gpt-oss at reasoning effort low. The budgets are not the native defaults and were chosen from measured truncation rates rather than from the model cards, for the reason given in Section E.8.

Arms. GEPA optimises the instruction text. Prompt tuning learns virtual tokens at the embedding. Prefix tuning learns per-layer key and value prefixes. The virtual token budgets are 100, 200 and 500 for both continuous methods, and all checkpoints share seed 42.

E.2 COMPLETE PRUNING RESULTS

Table 13 lists every level of the sweep. The rows marked "SE kept" repeat the -50% mask of the two Qwen arms that collapse there, with the super experts exempted and the same number of other experts removed instead; the removed routing mass is unchanged.

E.3 SUPER EXPERTS

Identification. We use the criterion of Su et al. (2026): over the first three quarters of the layers, an expert qualifies when the largest absolute output of its down-projection exceeds the 99.5th percentile over all experts and a tenth of the overall maximum, on 32 WikiText-2 segments of 2048 tokens. The

Table 12: Set-level F_1 on 2000 test rows with the least-used half or three quarters of every layer's experts masked, the share of unreadable answers, and F_1 with only the super experts pruned, next to the super experts the arm still uses. Answering nothing scores 0.330.

		F_1			unreadable (%)		super experts	
		original	-50%	-75%	-50%	-75%	kept	pruned
Qwen3-30B-A3B	base	0.466	0.575	0.330	0.0	100.0	all three	0.328
	GEPA	0.730	0.728	0.328	3.3	100.0	all three	0.328
	prompt, 100	0.829	0.814	0.440	0.0	0.2	L2 E92, L3 E82	0.409
	prompt, 200	0.813	0.330	0.330	100.0	100.0	all three	0.328
	prompt, 500	0.832	0.812	0.450	0.0	0.0	all three	0.328
	prefix, 100	0.830	0.815	0.383	0.0	0.3	all three	0.328
	prefix, 200	0.807	0.611	0.330	0.0	100.0	L2 E92, L3 E82	0.328
	prefix, 500	0.807	0.330	0.330	100.0	100.0	L3 E82	0.328
gpt-oss-20b	base	0.628	0.565	0.388	19.6	34.5	both	0.653
	GEPA	0.755	0.753	0.328	0.9	100.0	both	0.754
	prompt, 100	0.784	0.502	0.330	42.4	98.2	both	0.726
	prompt, 200	0.787	0.584	0.343	26.2	84.3	both	0.784
	prompt, 500	0.799	0.607	0.334	14.3	96.8	both	0.768
	prefix, 100	0.819	0.736	0.528	0.0	0.9	none	0.818
	prefix, 200	0.824	0.793	0.486	0.0	3.0	both	0.814
	prefix, 500	0.820	0.797	0.405	0.0	1.1	none	0.823

statistic has to be taken inside the fused expert computation of the runtime used here, where a hook on per-expert modules records nothing. On Qwen3-30B-A3B the criterion returns the published set, expert 92 of layer 2, 68 of layer 1 and 82 of layer 3 (732, 456 and 428), and on the Instruct-2507 checkpoint the same three (840, 468 and 466). On gpt-oss-20b it returns expert 5 of layer 17 and expert 5 of layer 6 ($54{,}272$ and $43{,}008$). All reach their maximum on the first position of a raw-text segment; under the chat template the Qwen experts reach it on position 1, the role token, in all 232 inputs checked, whatever the content.

Traffic and the frequency masks. Table 14 lists the super experts each mask removes, recomputed from the calibration maps with the sweep's rule. Qwen's sit in the middle and the tail of their layers and are removed in every arm, expert 68 of layer 1 already at -50%. On gpt-oss, expert 5 of layer 17 is among the three busiest experts of its layer in every arm and is never removed, and expert 5 of layer 6 is removed in four of the sixteen cells.

Magnitudes and pruning. The magnitude of a super expert is the largest absolute output of its down-projection over 200 Civil test rows. It counts as kept when that value stays within a factor of ten of the frozen base's. The super experts are pruned alone by zeroing their down-projection, and the task is scored with at most 64 generated tokens on Qwen and 1024 on gpt-oss. As in the frequency sweep, an unreadable answer counts as an empty label set, except for the pruned gpt-oss base, whose 1.5% unreadable answers count as wrong. The Qwen prefix trained at learning rate 10^{-4} is not an evaluated checkpoint; it is included as the one Qwen arm whose three super experts are all suppressed, with magnitudes below 1.

Where the attention goes. Attention shares are taken from the eager attention weights, per head and query, and averaged over heads and over queries from position 3 on, since the first positions can attend only to themselves, over whole sequences of 200 test rows. Pruning zeroes the down-projections of the super experts, as in Table 12. gpt-oss returns its weights without the learned sink column, so the sink's share is the weights' deficit from one; its panels show the full-attention layers 7 to 17. On Qwen the super experts build the sink from the second layer on, once expert 68 has written the peak: pruning them leaves the first layer unchanged and sends nearly all of the sink's attention to the rest of the sequence (Figure 5). Prompt tuning puts the peak on a virtual token and concentrates the attention on about twenty of its 500 virtual tokens. An untrained prefix already moves peak and sink to position 2, and the prefix trained at 10^{-4} has neither, intact or pruned (Figure 10).

Table 13: Frequency pruning, all levels. Mass is the routing mass removed, averaged over layers. Empty (answers with no label, unreadable ones included), unreadable, no end marker and looping (a phrase repeated until the budget runs out) are shares of the 2000 test answers.

model	arm	level (%)	mass (%)	F_1	exact	empty	unreadable	no end	looping
Qwen	base	0	0.0	0.466	0.211	26.5	0.0	0.0	0.0
	base	−50	5.8	0.575	0.167	21.1	0.0	0.1	0.0
	base	−75	28.3	0.330	0.330	100.0	100.0	61.9	21.7
	GEPA	−50	2.0	0.728	0.581	24.9	3.3	0.0	0.0
	GEPA	−75	17.7	0.328	0.328	100.0	100.0	100.0	100.0
	prompt, 100	−50	11.4	0.814	0.696	31.8	0.0	0.0	0.0
	prompt, 100	−75	35.7	0.440	0.026	3.0	0.2	29.1	28.7
	prompt, 200	−50	10.9	0.330	0.330	100.0	100.0	90.5	59.3
	prompt, 200	−50, SE kept	11.0	0.803	0.683	27.2	0.0	0.0	0.0
	prompt, 200	−75	32.8	0.330	0.330	100.0	100.0	93.8	6.3
	prompt, 500	−50	12.3	0.812	0.692	31.1	0.0	0.0	0.0
	prompt, 500	−75	32.3	0.450	0.389	83.9	0.0	0.0	0.0
	prefix, 100	−50	5.9	0.815	0.714	36.7	0.0	0.0	0.0
	prefix, 100	−75	27.7	0.383	0.368	93.1	0.3	0.0	0.0
	prefix, 200	−50	6.0	0.611	0.557	67.2	0.0	0.0	0.0
	prefix, 200	−75	28.0	0.330	0.330	100.0	100.0	100.0	18.0
	prefix, 500	−50	5.5	0.330	0.330	100.0	100.0	96.8	85.3
	prefix, 500	−50, SE kept	5.5	0.768	0.654	43.0	0.0	0.0	0.0
	prefix, 500	−75	27.7	0.330	0.330	100.0	100.0	99.8	15.0
gpt-oss	base	−50	13.3	0.565	0.422	59.4	19.6	1.3	4.7
	base	−75	39.8	0.388	0.309	76.3	34.5	24.1	31.3
	GEPA	−50	8.5	0.753	0.627	36.0	0.9	0.7	0.7
	GEPA	−75	29.2	0.328	0.328	100.0	100.0	100.0	98.5
	prompt, 100	−50	12.2	0.502	0.338	62.9	42.4	41.5	63.7
	prompt, 100	−75	33.6	0.330	0.325	98.3	98.2	49.6	55.3
	prompt, 200	−50	10.4	0.584	0.449	58.5	26.2	26.2	35.3
	prompt, 200	−75	29.7	0.343	0.326	95.7	84.3	84.1	96.7
	prompt, 500	−50	9.1	0.607	0.401	49.0	14.3	14.2	32.0
	prompt, 500	−75	25.5	0.334	0.323	97.2	96.8	96.7	96.0
	prefix, 100	−50	9.5	0.736	0.655	50.9	0.0	0.0	0.0
	prefix, 100	−75	32.6	0.528	0.450	75.7	0.9	0.0	0.0
	prefix, 200	−50	10.3	0.793	0.677	36.6	0.0	0.0	0.0
	prefix, 200	−75	34.6	0.486	0.332	69.0	3.0	0.0	1.0
	prefix, 500	−50	9.8	0.797	0.673	39.0	0.0	0.0	0.0
	prefix, 500	−75	31.7	0.405	0.381	91.0	1.1	0.0	0.0

Table 14: Super experts removed by the frequency masks of Table 12.

	Qwen3-30B-A3B		gpt-oss-20b	
	−50%	−75%	−50%	−75%
base	L1 E68, L2 E92	all three	—	—
GEPA	L1 E68	all three	—	L6 E5
prompt, 100	L1 E68	L1 E68, L3 E82	—	L6 E5
prompt, 200	L1 E68, L2 E92	L1 E68, L2 E92	—	—
prompt, 500	L1 E68	L1 E68, L2 E92	—	—
prefix, 100	L1 E68	L1 E68	—	—
prefix, 200	L1 E68, L2 E92	all three	L6 E5	L6 E5
prefix, 500	all three	all three	—	L6 E5

Removing only the channel of expert 68's output that carries the peak, on the first three positions, already takes two thirds of the sink's share from layer 3 on, whereas a random perturbation of the same norm moves much less of the attention, on the test rows and on WikiText-2 (Figures 11 and 12).

On gpt-oss the peak at position 0 is written in layer 6, persists to layer 16 and collapses at layer 17, the layers of its two super experts. Essentially only its full-attention layers look at position 0, 12.5%

Table 15: Largest output of each super expert over 200 test rows.

	Qwen3-30B-A3B			gpt-oss-20b	
	L1 E68	L2 E92	L3 E82	L6 E5	L17 E5
base	422	800	458	43 000	52 700
GEPA	422	800	458	43 000	52 700
prompt, 100	11.1	250	572	43 000	50 700
prompt, 200	470	764	472	43 000	51 500
prompt, 500	1000	952	572	43 000	46 900
prefix, 100	592	776	464	290	1 520
prefix, 200	0.65	188	486	43 000	47 900
prefix, 500	0.55	16.1	520	189	1 990

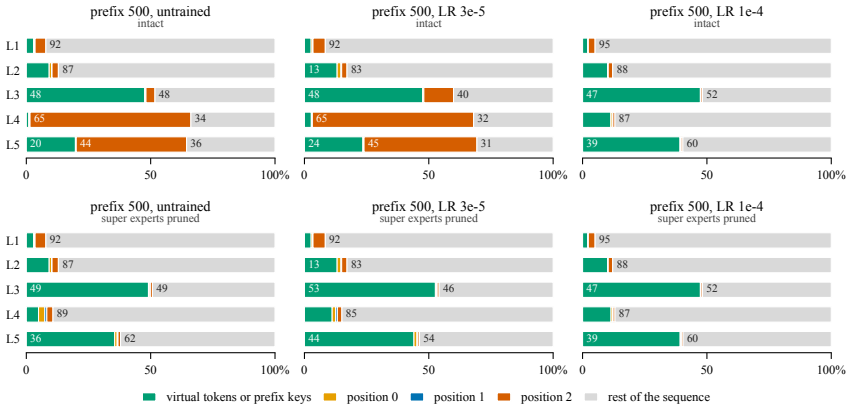


Figure 10: As Figure 5, for a Qwen prefix of 500 tokens, untrained and trained at two learning rates.

of their attention on average between layers 7 and 17, while the learned sink takes 13 to 70 percent in every layer. Pruning the super experts removes the peak, and the attention on position 0 falls to 1.1%; of the 11.4 points it loses, 5.0 go to the learned sink and the rest to the sequence (Figure 13). Masking them in the router gives the same profile. This is consistent with the learned sink supplying the attention that has nowhere else to go (Sun et al., 2024; Gu et al., 2025), which on Qwen can only be placed on the peak's token. gpt-oss nonetheless still builds the peak, which its sink logit, having no value vector, cannot replace (Queipo-de Llano et al., 2025); the pruning results say only that this task and this text do not need it.

Across the gpt-oss arms, prefix keys take over part of the learned sink's share in the first layers, whether the super experts survive or not (Figure 14).

Two prefixes, one score. The two Qwen prefixes of 500 tokens that differ only in the learning rate reach the same F_1, but one keeps a super expert and the other none. The one trained at 10^{-4} has neither the peak nor the sink (Figure 10) and loses almost nothing when its super experts are pruned (0.807 intact, 0.803 pruned), but it damages the model most outside the task, through the prefix of its first six layers (Table 16). One method at one score can thus come with opposite dependence on the super experts; with a single seed, this does not show that the learning rate decides it.

Prefix damage outside the task. Removing the prefix from the first six layers alone, while it stays in the other 42, brings the WikiText-2 perplexity of every Qwen prefix of Figure 10 back close to the base's, and the prefix trained at 10^{-4} keeps most of its task gain (Table 16). The damage outside the task is carried by the early layers of the prefix, the task gain mostly by the later ones. The prefix trained at $3 \cdot 10^{-5}$ loses its strict score because 91 and 65 of its 200 answers then name `toxic` instead of `toxicity`; accepting that alias, for every arm alike, gives 0.754 with values zeroed and 0.793 with keys masked, against 0.832 intact.

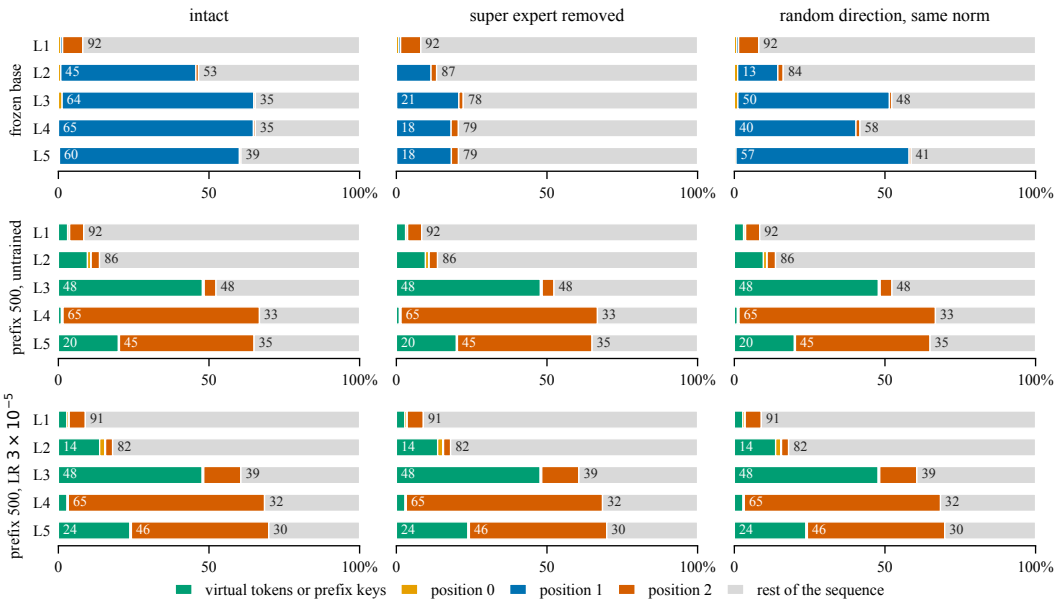


Figure 11: Attention shares on Qwen, layers 1 to 5, 200 Civil test rows: intact, with the super expert removed, and with a random perturbation of the same norm.

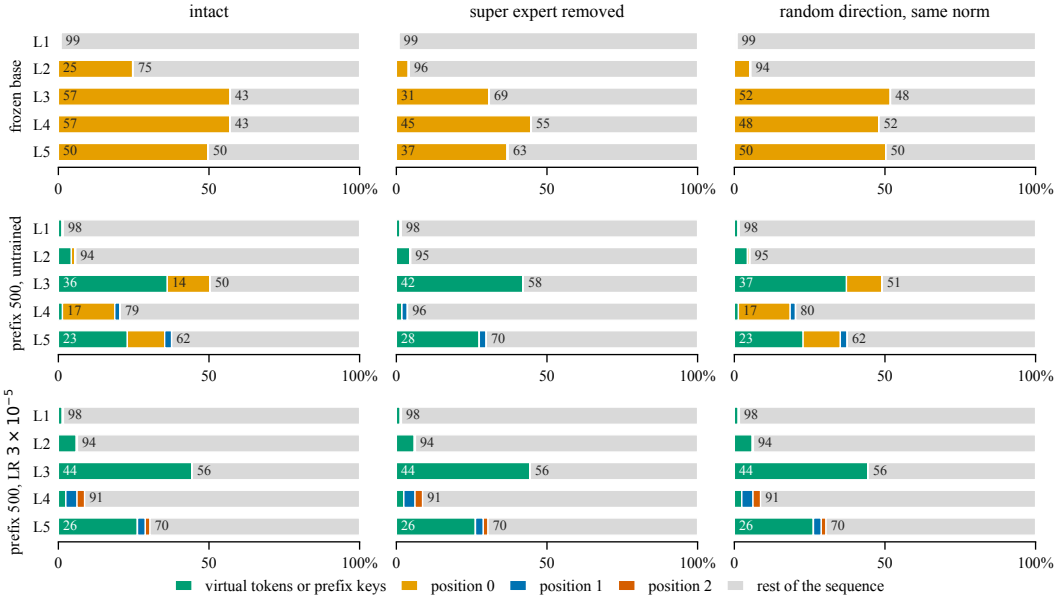


Figure 12: As Figure 11, on WikiText-2.

E.4 Retraining the router

Only the router of Qwen3-30B-A3B is trained: the 48 gate matrices, 128×2048 each and 12.6M parameters in total, with every expert, every adapter and all other weights frozen. Each arm is trained on its own 1000 training rows and targets, with cross-entropy on the answer tokens, AdamW at learning rates 10^{-5}, $3 \cdot 10^{-5}$ and 10^{-4} (linear warm-up over a tenth of the steps, then linear decay), batches of eight and four epochs; the optimiser keeps an fp32 copy of the gates. Epoch 0 is the frozen router. The learning rate and epoch are chosen on the 200 validation rows on which the arms' own checkpoints were selected, and the chosen router is scored on the 2000 test rows (Table 17).

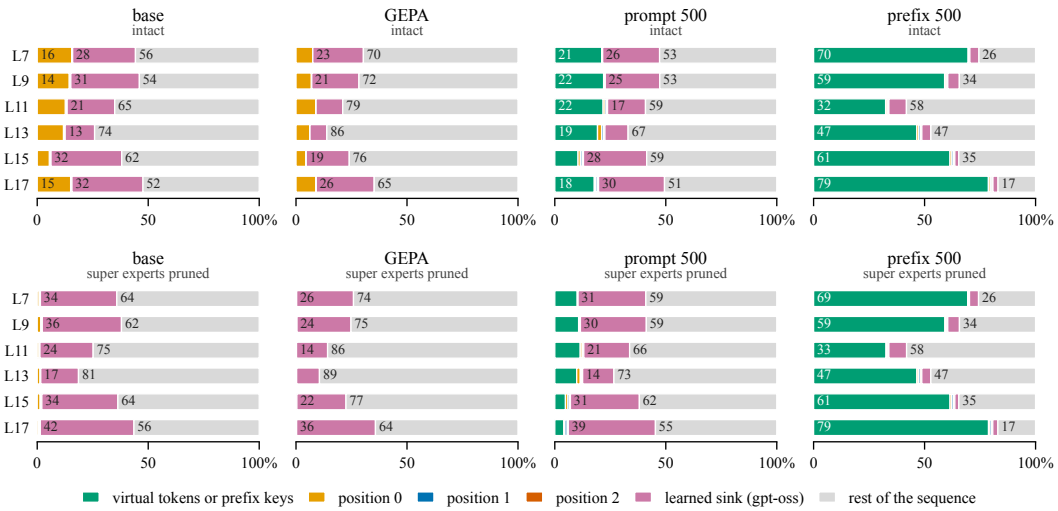


Figure 13: As Figure 5, on gpt-oss, full-attention layers 7 to 17.

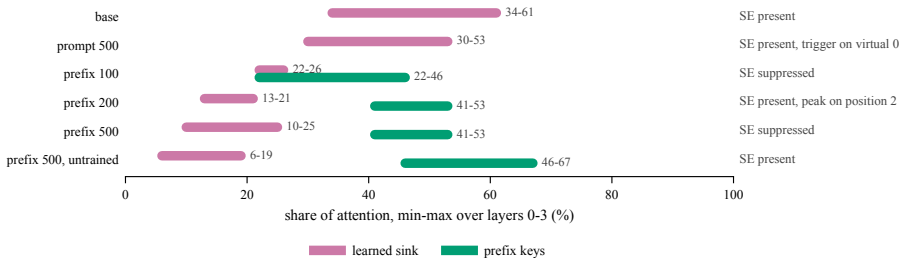


Figure 14: gpt-oss: range over layers 0 to 3 of the attention share taken by the learned sink and by prefix keys, per arm, with the state of the super experts.

On the frozen base, the retrained router recovers most of what adaptation reaches, while moving less of the comment tokens' routing mass than prompt or prefix tuning does (Figure 4). On the continuous arms it adds nothing: at every learning rate the test difference stays within ± 0.013, and a validation gain, such as $+0.021$ for prefix tuning at 500 tokens, does not carry to the test rows. GEPA, which routes the comment tokens almost as the base does, gains 0.022 to 0.073 depending on the learning rate, and at the chosen one reaches the 0.80 of the retrained base: the router recovers what GEPA gains and GEPA adds nothing on top of it. Retraining the router and adapting the context are thus substitutes, and combining them never beats the better of the two by more than 0.003. This repeats, on the present checkpoints, what the workshop draft found on another backbone pair with a different recipe.

E.5 Expert-parallel timing

Each of N GPUs holds E/N consecutive experts of every MoE layer and a batch of tokens. A layer is one real expert-parallel forward: an all-to-all that sends every token to the devices of its selected experts, a grouped matrix product over the local experts (SwiGLU, the real expert shapes, random weights), and an all-to-all back. Its time is the slowest device's, and a step is the sum over layers. Each token draws its k distinct experts per layer from the arm's measured expert shares over every routed position (the routing maps), so the per-device load follows the arm's own load. The timing covers the MoE layers only, without attention or a serving framework, and is not an end-to-end latency. At 4096 tokens per device, where compute dominates, the base's own imbalance costs more the fewer experts each device holds, and the adapted arms stay within 6 percent of the base on two and four devices and within 14 percent on eight (Table 18); at 512 tokens per device fixed costs dominate and every adapted arm lies within 6 percent of the base at every device count.

Table 16: Qwen prefixes of 500 tokens with the prefix of layers 0 to 5 removed, by zeroing its values or masking its keys, at every position. Perplexity on the 16 WikiText-2 segments of 2048 tokens, strict F_1 on 200 validation rows. The base scores 6.97 and 0.458.

	WikiText-2 perplexity			F_1		
prefix	intact	values zeroed	keys masked	intact	values zeroed	keys masked
untrained	48.0	7.22	6.87	0.459	0.475	0.446
LR $3 \cdot 10^{-5}$	34.0	7.48	7.16	0.832	0.413	0.532
LR 10^{-4}	275	8.95	8.63	0.798	0.762	0.773

Table 17: Retraining only the router on top of each Qwen arm: set-level F_1 on 2000 test rows with the frozen and the retrained router, the paired difference, and the routing displacement of Equation (22) that the retraining itself causes on the comment tokens of the validation rows.

arm	frozen	retrained	difference (t)	TV, retrained vs frozen
base	0.461	0.800	$+0.339$ (39.4)	0.080
GEPA	0.730	0.803	$+0.073$ (8.8)	0.076
prompt, 100	0.829	0.824	-0.005 (-1.5)	0.030
prompt, 500	0.832	0.822	-0.009 (-3.2)	0.011
prefix, 100	0.830	0.832	$+0.002$ (0.3)	0.057
prefix, 500	0.807	0.809	$+0.002$ (0.3)	0.134

E.6 Complete load statistics

The per-layer Gini values, the per-expert distribution of every layer, and the statistics of the experts that survive the masks of Section 4.3 are released as tables: the expert-by-layer distribution for all sixteen checkpoints, the same distribution resolved by prompt stage, per-layer Gini with the busiest-expert and busiest-tenth shares, and the before-and-after Gini of the surviving experts. Experts with no traffic are omitted from the distribution tables and counted separately, since a row of zeros is a fact about the router and not a fact about the tokens.

E.7 An experiment carried over from the workshop draft

The following result was established on the earlier backbone pair and is kept because it bears on whether the model needs the routes adaptation produces. It is reported here rather than in the main text because its backbone pairing is not the one used above.

Transplanting routing decisions. Substituting the adapted arm's expert choices into the frozen base moves F_1 by at most 0.0057, and transplanting adapted routes onto the base recovers at most 0.0099 of the adaptation gain. A matched randomisation, which perturbs exactly as many routing decisions as the substitution changed, costs prompt tuning 0.0930 and prefix tuning 0.0841 and leaves GEPA unchanged. The identities are therefore not what the gain is made of, while the structure is, and only for the continuous arms.

E.8 Measurement caveats

The level-zero numbers come from a different harness. Scores for the unpruned checkpoints were produced by the run that selected them, with its own decoding and a 64-token budget, while every pruned level here is decoded by ours. We closed most of the gap rather than assuming it away. The two scorers implement the same set-level F_1 and agree to 0.0009 on identical saved answers; the two parsers reproduce each other's predictions on all 1993 shared rows; the test sets overlap on all but seven rows. What remained unverified was generation itself, so level zero was re-measured with our harness on eight of the sixteen cells, and the difference is at most 0.0012 F_1 on the PEFT arms. It is not small on the two text arms, where the reasoning effort of gpt-oss turned out to matter by 0.075 F_1, and those cells were recomputed. The Qwen base, re-measured unmasked in the pruning harness, scores 0.466.

Table 18: Time of the MoE layers per step under expert parallelism relative to the base, 4096 tokens per H100 GPU, 20 repetitions; the last row is the base's absolute time.

	gpt-oss-20b			Qwen3-30B-A3B		
GPUs	2	4	8	2	4	8
uniform load	0.956	0.853	0.707	0.975	0.938	0.821
GEPA	1.010	1.040	1.109	1.008	1.046	1.062
prompt, 100	1.020	1.038	1.071	1.004	1.000	0.979
prompt, 200	1.038	1.039	1.096	1.014	1.014	0.993
prompt, 500	1.042	1.061	1.135	1.019	1.026	1.007
prefix, 100	1.006	1.020	1.052	1.006	1.012	1.013
prefix, 200	1.004	1.004	1.044	1.000	0.997	1.009
prefix, 500	1.006	1.029	1.062	1.003	1.004	1.011
base (ms)	82.2	95.8	117.9	143.8	162.0	188.4

Table 19: Expert load on 500 calibration rows, median over layers, every budget.

		positions per row	Gini	Gini, comment tokens	busiest expert (%)	dead experts
Qwen3-30B-A3B	base	153	0.635	0.757	5.2	7.8
	GEPA	2154	0.730	0.759	8.2	7.6
	prompt, 100	253	0.583	0.753	5.0	3.0
	prompt, 200	353	0.610	0.747	6.1	4.5
	prompt, 500	653	0.620	0.753	6.6	3.0
	prefix, 100	153	0.640	0.760	5.8	8.4
	prefix, 200	153	0.647	0.761	5.6	8.3
	prefix, 500	153	0.648	0.763	5.5	9.4
gpt-oss-20b	base	626	0.513	0.687	11.2	0.5
	GEPA	1218	0.595	0.706	16.3	0.8
	prompt, 100	313	0.563	0.693	15.8	1.0
	prompt, 200	413	0.607	0.689	17.9	0.8
	prompt, 500	713	0.661	0.711	19.3	1.2
	prefix, 100	213	0.584	0.682	15.6	2.2
	prefix, 200	213	0.572	0.685	14.8	2.0
	prefix, 500	213	0.591	0.661	17.0	1.6

Token budgets were set from measured truncation, not from defaults. At the native 512-token budget, pruned gpt-oss arms return to writing reasoning they have lost the ability to finish and spend the entire budget on it, which reads as truncation rather than as damage. Raising the budget to 4096 moved the frozen base from a 10.7% truncation rate to 0.75% and its score from 0.676 to 0.703. For Qwen the opposite holds: a healthy answer is under forty tokens, so 512 is a twelvefold margin, and rows that still fill it are degenerate rather than truncated. Reporting a wider budget would hide them.

Which positions are counted. Per calibration row, the comment spans 69 tokens on Qwen and 68 on gpt-oss under every arm. All routed positions together number 153 on Qwen for the base and the prefix arms, 253 to 653 for prompt tuning, which adds its virtual tokens, and 2154 for GEPA, whose instruction is about 2000 tokens long. On gpt-oss they number 626 for the base, 213 for the prefix arms and 313 to 713 for prompt tuning, and 1218 for GEPA; the base and GEPA reason and write long answers, the tuned arms answer in about eight tokens. A load profile over all positions therefore compares different token populations. Its displacement from the base, up to 0.56 per layer, is mostly a change in what is routed: routing every stage the base also has exactly as the base does, and keeping only the arm's own sequence composition, reproduces 93 to 98 percent of the median displacement of the prompt-tuned Qwen arms, 76 to 97 percent of the prompt-tuned gpt-oss arms, and 60 to 66 percent of the gpt-oss prefix arms, which lose the base's reasoning. The Qwen prefix arms route the same positions as the base, and their displacement is rerouting throughout. The virtual tokens of prompt tuning on Qwen put 14 to 16 percent of their assignments on experts that receive less than a tenth of the uniform share in the base, and reach about five of the 7.8 experts per layer

that the base leaves dead, which spreads the load; on gpt-oss the adapted PEFT arms stop generating the reasoning that spreads the base's load. The same construction moves the median Gini in the direction the arm's own Gini moves, for every arm whose composition differs from the base's.

Answers, not only scores. Every claim about a cell at the floor in Section 4.3 was checked against the raw transcripts. This is how the three distinct failure textures, repeating a label, repeating a phrase, and echoing the instruction, were separated from each other, and it is why the degeneration columns are in Table 13 rather than in a footnote.