On Potentiality and Actuality Of Foundation Models

Anonymous authors
Paper under double-blind review

Abstract

Foundation models are built through large-scale pretraining, often followed by post-training, and are then adapted to specific tasks and behaviors. But, does the adaptation add truly new capabilities, or the base model already holds it? We decompose the update induced by different adaptations into a component shared across all inputs and a residual that varies with the input. We find that the input-dependent residual carries little of the adaptation gain: as evidence, a shift adapter that learns a constant vector per layer, identical for every input, reproduces the behavior of LoRA and full fine-tuning. We call this the *potentiality principle* of foundation models: the base model already contains the capacity needed for task-specific expertise. To adapt to a new task, the model just needs to be *actualized*, *i.e.*, pushed in a certain direction via a constant, input-independent shift. We show the implications of this principle on parameter-efficient adaptation, steering, and convergence across models, and how it explains the neural thickets phenomenon.

1 Introduction

The rapid progress of foundation models (Bommasani et al., 2021) has been driven by large-scale pre-training on general-domain data, often followed by post-training and finally adaptation to a specific task, behavior, or domain. To adapt these models effectively, different adaptation strategies have been explored. The earliest form of adaptation is full fine-tuning, which updates all model parameters on a given task (Howard & Ruder, 2018; Devlin et al., 2019). To reduce computational cost, parameter-efficient methods freeze the model and learn only a small update, such as the widely used low-rank update of LoRA (Hu et al., 2022; Li & Liang, 2021; Lester et al., 2021). In-context learning showed that it is possible to adapt by eliminating parameter updates altogether and instead conditioning the model on demonstrations in the prompt (Brown et al., 2020). Similarly, activation steering leaves the model parameters unchanged, but adapts model behavior at inference time by directly modifying hidden activations (Turner et al., 2023; Rimsky et al., 2024; Zou et al., 2023). So far, the field has largely treated these adaptation strategies as different tools suited for different settings.

Potentiality and Actuality Principle

Foundation models already contain the task-specific expertise in *potentiality*. Adaptation only *actualizes* this expertise via a constant, input-independent shift.

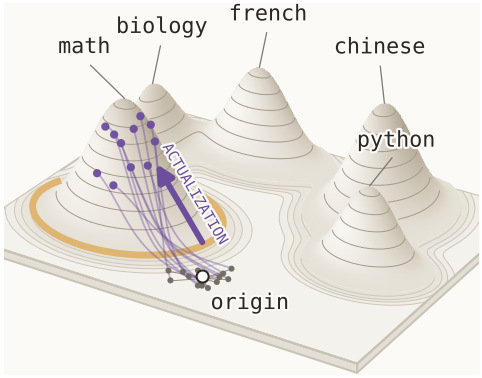

Figure 1: **The Potentiality and Actuality Principle.** A foundation model holds task-specific expertise (hills) in *potentiality*. For most tasks, a single shift, same for every input (arrow), is all it takes to move the inputs to one of these tasks. We call this process *actualization*.

But what is the underlying mechanism of adaptation, and what changes inside a foundation model when it gains task-specific expertise? While these questions remain largely unanswered, a growing body of evidence suggests that task adaptation may rely on small changes to a pre-trained model. In particular, fine-tuning succeeds when the update is restricted to a small parameter space (Aghajanyan et al., 2021; Hu et al., 2022), or even when it is restricted to biases alone (Ben Zaken et al., 2022; Sinii

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.

et al., 2025). Recent work on neural thickets even showed that random perturbations of pre-trained weights find dense sets of diverse task experts (Gan & Isola, 2026). Why such small interventions are sufficient to produce task-specific expertise remains poorly understood. Previous work has suggested that we are in the elicitation overhang: AI models have untapped capabilities, and we do not know how to elicit them from frontier models (Turner et al., 2023).

To understand the adaptation mechanism, we first decompose the update that LoRA learns into two components: *(i)* a component shared across all inputs, and *(ii)* an input-specific component, *i.e.*, a residual that varies with the input. Across many diverse tasks, we find that little input-specific adaptation happens: the update vectors point in nearly the same direction independent of the input. Quantitatively, on 26 benchmarks and 17 language models from 7 architecture families, the component shared across all the inputs preserves 85% of the adapted behavior. Motivated by this finding, we adapt the model via *shift adapter*, a simple adaptation mechanism that learns only the shared part of adaptation: a single vector per layer, constant for every input. The shift adapter matches LoRA's performance across 17 tasks and 17 language models, with roughly two orders of magnitude fewer parameters. The same holds for adaptation without the rank constraint and for full fine-tuning, where the savings are even larger.

These findings suggest that the base model itself already has the capacity for diverse task-specific expertise and only a small push in the right direction is needed to put this capacity to work. We call this the *potentiality principle* of foundation models, and the push towards a direction to gain task-specific expertise the *actualization* of a foundation model. The terms reflect Aristotle's distinction between *potentiality* (*dunamis*), a capacity that a thing holds without exercising it, and *actuality* (*energeia*), that same capacity at work (Aristotle, ca. 350 B.C.E.). In this terminology, a pre-trained foundation model holds different task-specific expertise as potentiality, and that expertise can be actualized via a constant shift that moves the model into a state where it can be exercised (Figure 1).

The potentiality and actuality principle has implications on *fine-tuning*, *steering*, *neural thickets*, and *model convergence*. It provides an explanation for why fine-tuning can succeed in a tiny parameter space (Aghajanyan et al., 2021; Hu et al., 2022; Ben Zaken et al., 2022; Sinii et al., 2025), identifies the trained shift as a reference direction for steering (Turner et al., 2023; Rimsky et al., 2024; Zou et al., 2023), relates the dense task experts found by random perturbations to states that actualize existing capacities (Gan & Isola, 2026), and shows that the states of different models grow alike as the models gain capability (Huh et al., 2024; Gröger et al., 2026). In each case the mechanism is the same: a state that actualizes a capacity the model already has.

2 Shared and input-specific components of adaptation

Imagine that a foundation model already holds the task-specific expertise in *potentiality* and that adaptation only has to *actualize* it by selecting the desired expertise. In such a case, the selection would be the same for every input of the task, *i.e.*, the update learned during adaptation would not vary across inputs, or would vary only slightly.

To test this hypothesis, we split the change that adaptation does into two components: *(i)* a component shared by all the inputs, and *(ii)* an input-specific component. We measure how much of each component changes during adaptation, and evaluate the model using only one of the two components to determine which component the adaptation relies on. We explore this for different kinds of adaptation: LoRA (Hu et al., 2022), the same update trained without the rank constraint (the *unconstrained update*), full fine-tuning (Howard & Ruder, 2018; Devlin et al., 2019), and in-context learning (Brown et al., 2020).

Adaptation field. Every form of adaptation changes the hidden states that the model computes for an input, whether it updates the weights or places demonstrations in the prompt. We thus describe adaptation through this change. Let $h_\ell^{\text{base}}(x) \in \mathbb{R}^d$ and $h_\ell^{\text{adapted}}(x) \in \mathbb{R}^d$ be the hidden states after layer ℓ of the base and the adapted model for input x, where d is the hidden size. We define the *adaptation field* at layer ℓ as their difference and decompose it into a component shared by all inputs, its mean, and an input-specific component,

$$\mathcal{F}_\ell(x) = h_\ell^{\text{adapted}}(x) - h_\ell^{\text{base}}(x) = \underbrace{\bar{\mathcal{F}}_\ell}_{\text{shared component}} + \underbrace{\widetilde{\mathcal{F}}_\ell(x)}_{\text{input-specific component}}, \qquad \bar{\mathcal{F}}_\ell = \mathbb{E}_x[\mathcal{F}_\ell(x)]. \quad (1)$$

The shared component $\bar{\mathcal{F}}_\ell$ is one vector per layer, the same for every input. The input-specific component $\widetilde{\mathcal{F}}_\ell(x)$ has zero mean over inputs, so the two components add in energy (Proposition B.1),

$$\mathbb{E}_x\|\mathcal{F}_\ell(x)\|^2 = \|\bar{\mathcal{F}}_\ell\|^2 + \mathbb{E}_x\|\widetilde{\mathcal{F}}_\ell(x)\|^2. \tag{2}$$

We quantify the decomposition with two statistics (Figure 2, Appendix C.7): *(i)* the share of the field's energy that the shared component holds defined as $\|\bar{\mathcal{F}}_\ell\|^2/\mathbb{E}_x\|\mathcal{F}_\ell(x)\|^2$ (the first term of Equation (2) over its left side), which we call the *field share*, and *(ii)* the average cosine between each input's field $\mathcal{F}_\ell(x)$ and the shared component $\bar{\mathcal{F}}_\ell$. The cosine similarity is close to 1 if adaptation moves every input in the same direction and close to 0 if it moves each input differently.

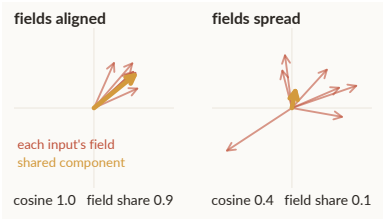

Figure 2: **The field statistics.** Fields that point the same way (left) have a high cosine to their shared component and a high field share. Fields that spread (right) have neither.

Low-Rank Adaptation (LoRA). In LoRA adaptation, a low-rank update $\Delta W = BA$ is added to each weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, which maps a d_{in}-dimensional input to a d_{out}-dimensional output, with $B \in \mathbb{R}^{d_{\text{out}} \times r}$, $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and rank $r \ll \min(d_{\text{in}}, d_{\text{out}})$. We adapt every linear projection in every layer, the common choice (Dettmers et al., 2023; Biderman et al., 2024), and consider one adapted matrix W at a time, since the derivation is the same for each. Let $z_t(x) \in \mathbb{R}^{d_{\text{in}}}$ be the hidden state that W multiplies at token position t for input x. The base model computes $Wz_t(x)$ and the adapted model $(W + \Delta W)z_t(x)$, so what LoRA adds at this matrix is the vector $a_t(x) = \Delta W z_t(x) \in \mathbb{R}^{d_{\text{out}}}$. Unless stated otherwise, we take z_t at the last token of the input, where the model has read the entire input and is about to answer, and drop the index t. Because ΔW is linear, the added vector can be split into the shared and input-specific component:

$$a(x) = \Delta W z(x) = \underbrace{\Delta W \mu}_{\text{shared component}} + \underbrace{\Delta W\big(z(x) - \mu\big)}_{\text{input-specific component}} = \bar{a} + \widetilde{a}(x), \tag{3}$$

where $\mu = \mathbb{E}_x[z(x)] \in \mathbb{R}^{d_{\text{in}}}$ is the average hidden state estimated on held-out inputs the adapter has never been trained on.

Experimental setup. We apply the decomposition to trained LoRAs of rank 8, on 17 models from 7 architecture families: 13 instruction-tuned, 3 base models, and 1 trained on instructions but not preference-tuned (Appendix C.2). Each model is adapted on 26 benchmarks from two categories: *(i)* 17 reasoning and knowledge tasks with labeled answers which test what the model knows and how it reasons, and *(ii)* 9 behaviors including 4 traits which ask for the answer in a given language, persona, or format, 3 transformations which ask the model to apply a specific rule to the input, and 2 maps, a letter cipher and a lookup table, which assign each input an answer by an arbitrary rule. Nothing in pretraining fixes that rule, so the maps are the one category the model can only learn from the adaptation data. We refer to the first category as *tasks*, and the second as *behaviors*. The tasks span commonsense reasoning, text classification, subject-specific question answering, and recent complex reasoning benchmarks (e.g., BoolQ, HellaSwag, MMLU-Pro, and SuperGPQA), following the protocol of Hu et al. (2023); Liu et al. (2024b) (details Appendix C.3). Tasks are evaluated by accuracy on their evaluation splits, while behaviors are evaluated on 1,000 held-out generations by adherence to the target behavior. For each intervention, we report the fraction of the adaptation gain over the base model that is retained. Unless stated otherwise, results are averaged over 3 seeds.

Observation. Across 17 tasks and 17 models, each input's field points along the shared component at a cosine of 0.91 on average, and the shared component holds 83% of the field's energy (Figure 3a, all models in Appendix D.2). Across the 9 behaviors and 17 models, the cosine is 0.84 and the shared component holds 71% of the field's energy (Appendix D.5). The same decomposition applies to the adapter's own output $a(x) = \Delta W z(x)$, before it propagates through the model. There the shared component $\bar{a}$ holds 93% of the energy on these behaviors, averaged over every adapted matrix (Appendix D.3). The fields of different inputs thus point in nearly the same direction: LoRA moves every input mostly the same way, and the part of the change that depends on the input holds the smaller share of the energy on both tasks and behaviors.

Interventions. We next explored which component the LoRA gains come from using two interventions: *(i)* keeping only the shared component, and *(ii)* removing the shared component. In the first

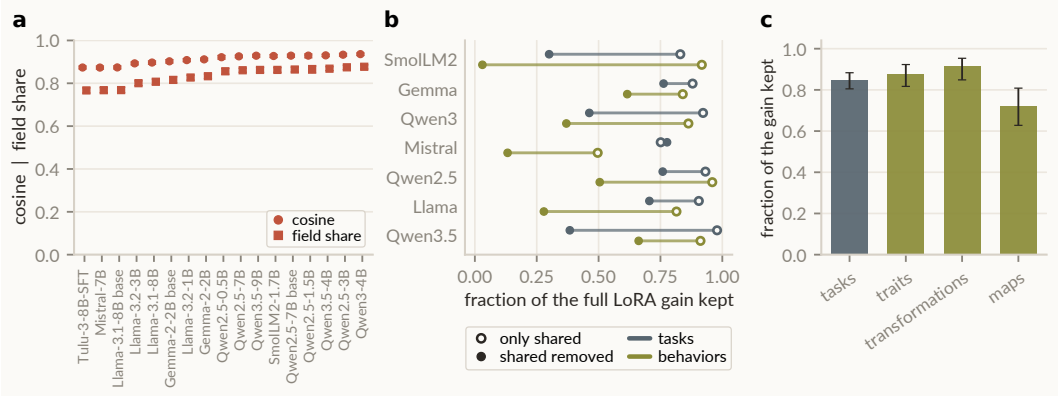

Figure 3: **LoRA adaptation mainly relies on the component shared across all inputs. (a)** Two decomposition statistics for LoRA: each input's field points along the shared component (cosine), and the shared component holds most of the field's energy (field share). Each point is the model's mean across 17 tasks, and the bars are 95% bootstrap intervals over the runs, too small to be visible. **(b)** The fraction of the full LoRA's gain kept using only the component shared across all inputs (hollow) and with the shared component removed (filled), per model on the 17 tasks and on the 9 behaviors. Rows are the largest instruction-tuned model of each family. **(c)** The fraction of LoRA's gain kept using only the component shared across all inputs, for the 17 tasks, and 9 behaviors (4 traits, 3 transformations, and 2 maps, cipher and lookup) (Appendix D.2). Each bar pools the runs of 17 models, and the error bars are 95% bootstrap intervals over models. Results for all models and tasks are in Appendix D.2 (Figure 11).

intervention, the same shared vector $\bar{a}$ replaces $a(x)$ in Equation (3) at every adapted matrix and at every position so the adapter becomes one constant vector per matrix. In the second intervention, $\bar{a}$ is subtracted at every position, so the model relies only on the input-specific component $\widetilde{a}(x)$. The former tests whether the shared component alone produces the adapted behavior on new inputs, while the latter tests whether the trained adapter depends on the shared component.

Over all 17 models, keeping only the shared component keeps on average 85% of the adapter's gain on the tasks, and removing it keeps 61% (Figure 3b, Figure 11). All models keep more with the shared component alone than without it. The only exception is Mistral-7B, which leans more on the input-specific component. The 4 models that never received instruction tuning keep 80% of the gain with only the shared component and 54% with it removed, against 86% and 63% on the instruction-tuned models, so the reliance on the shared component is not a product of instruction tuning (Appendix D.4). The same holds for behaviors, where the shared component alone keeps 84% of the gain on average, and removing it keeps 27% on the traits, 52% on the transformations, and 25% on the maps. The update matters while the model reads the input, not while it writes the answer: with the adapter switched on only for the input tokens, LoRA keeps a median of 91% of its gain, and only for the answer tokens 3%, on 17 models and 6 tasks (Appendix D.7). So most of what LoRA gains comes from one vector per matrix that is the same for every input, added while the model reads the input.

We next ask whether adaptation still relies on a shared component when producing the correct behavior requires input-specific computation. The 3 transformations, such as copying the input or returning its last word, apply a rule to the input that the base model can already follow, so the computation that varies with the input is the model's own. Surprisingly, even there, keeping only the shared component keeps 91% of the gain, as much as on the 4 traits, which ask for the same behavior on every input (87%, Figure 3c). The 2 maps apply a rule fixed by an arbitrary key, a substitution of the first letter, or a table of words, which the model can only learn during adaptation, and they keep the least, 72% of adaptation gain. This indicates that gains of the shared component fall when the base model does not hold the rule. On the maps, what adaptation adds is the table, and what the base model provides is the ability to use one: a shift adapter trained on a map answers as well as LoRA does, and the table it stores produces nothing on its own (Appendix E.5).

Unconstrained update. To investigate the influence of the low-rank update on the decomposition result, we train the same update without the low-rank factorization, so that ΔW is a full matrix at every adapted matrix, on the same 17 models, the 4 traits, and the 3 transformations. After training,

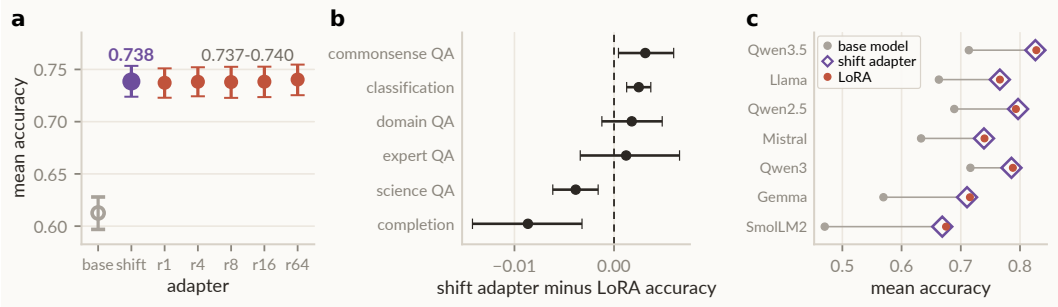

Figure 4: **Shift adapter matches LoRA's performance. (a)** Mean accuracy of the base model, shift adapter, and LoRAs with different ranks. The mean is computed across 17 models and 17 tasks, with bootstrap 95% intervals over the runs. **(b)** The paired difference between the shift and the rank-8 LoRA per task type, over 17 models and 17 tasks. Error bars are bootstrap 95% intervals over the runs. **(c)** The largest model in each family, ordered by scale: mean accuracy of the base model, the shift, and the rank-8 LoRA, over 17 tasks. All models and tasks are shown in Appendix E.1.

the update has high rank (median stable rank 45*) and raises adherence as much as LoRA does (0.37 against 0.38 above the base model), yet its shared component holds 95% of the energy of what it adds, against LoRA's 90%. Without the shared component, it keeps 17% of the adaptation gain on traits and 41% on transformations, against LoRA's 29% and 55%. With the shared component alone, it keeps 71% against LoRA's 88% on traits (Appendix D.9). The shared component is therefore not a product of the low-rank constraint: with the constraint gone, it grows from 90% of the energy to 95%, while the behavior relies on it slightly less.

Fine-tuning and ICL. We further examine whether full fine-tuning and in-context learning (ICL) rely on the shared component in the same way LoRA does. Full fine-tuning changes every weight, so its update cannot be decomposed and intervened on as LoRA's. However, we can examine the structure of its adaptation field: we find that adaptation via fine-tuning results in nearly the same shared structure as LoRA. Across the 17 tasks and 17 models, its cosine is 0.90, and its field share is 0.82 on average, compared to 0.91 and 0.83 for LoRA (Appendix E.2), suggesting that all inputs are moved in the same direction, just like we observe for LoRA. In contrast, ICL relies more on input-dependent changes: its shared component reproduces traits well but substantially less of the gain on input-dependent transformations and on the tasks, and its shared component remains nearly orthogonal to LoRA's even as their induced hidden-state changes become more aligned in later layers (Appendix D.10). This suggests that ICL reaches similar behaviors through a different, more input-dependent route. Additional results are in Appendix D.

3 Actualizing with a shift adapter

In the previous section, we showed that we can preserve most of the adaptation gain by keeping only the component shared across the inputs. If this shared component is indeed sufficient to produce the adapted behavior or solve a task, then we should be able to learn such a change directly. We therefore define a *shift adapter*: one learned vector $b_\ell \in \mathbb{R}^d$ per layer ℓ, added to the hidden state of every input at every position. Its only operation is a constant push. The shift applies the same push to every input and moves the model into a state in which its expertise for one task is actualized. We call such a state an *actualizing state*. Note that a constant parameter does not mean a constant effect: the layers above process the push together with an input-dependent hidden state, so the same push can move different inputs differently, and they can reach different actualizing states. However, the capacity to move inputs differently lies in the base model, and the shift only actualizes it via a constant push.

Shift adapter matches LoRA. We compare the shift adapter to LoRAs of rank $\{1, 4, 8, 16, 64\}$ on the 17 tasks across 17 models, with the training and evaluation protocol of Appendix E.1. The shift reaches a mean accuracy of 0.738, against 0.738 for a LoRA of rank 8 and 0.613 for the base model (Figure 4a), recovering all of the LoRA's gain, with $L \times d$ trainable parameters, 0.04M on Qwen2.5-1.5B compared to 9.2M for LoRA with rank 8. The same holds at every rank: from rank 1 to rank 64

*Stable rank is $\|W\|_F^2/\|W\|_2^2$, the squared Frobenius norm over the squared spectral norm.

the LoRA's accuracy moves by 0.003 while its parameters grow from 1.2M to 74M on that model. The shift stays within 0.013 of the rank-8 LoRA on any task and within 0.011 on any model, from 0.5B to 9.5B parameters (Figure 4b,c). Training all weights reaches 0.710 under the protocol above, so the shift adapter is above full fine-tuning on every model.

The shift and LoRA agree on the actualizing direction. The shift adapter and LoRA are trained independently, so we next wondered whether they push the inputs along the same direction and finally reach the same actualizing state. Across the 17 tasks and 17 models, the average cosine similarity of LoRA's shared component and the shift adapter is 0.65 on average, while a LoRA with random weights gives 0.06 (Appendix E.4). Across the 9 behaviors and 17 models, the cosine similarity is 0.53 on average, while a LoRA with random weights gives 0.08 compared to the shift adapter (Figure 5). Every bar sits above its random counterpart: a LoRA with random weights reaches 0.85 on the tasks and 0.67 on the behaviors for the cosine to its own shared component, and only 0.06 and 0.08 for its cosine to the trained shift (Figure 5). If, instead of weighting each layer equally as before, we weight each layer by how far the shift adapter moves it, cosine similarity rises to 0.69, and the random LoRA remains at 0.08 (Appendix C.7). The comparable performance and similar direction of the shift adapter and LoRA indicate that LoRA relies on the direction the shift adapter learns, even though it can vary its update with every input. Additional results are in Appendix E.

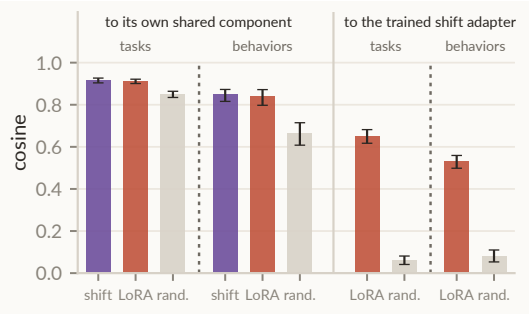

Figure 5: **Independent adaptations push the same way.** Mean over runs with 95% intervals over models. Left, the cosine of each input's field to its adapter's own shared component, for the shift, the LoRA and a LoRA with random weights, on the 17 tasks and on the 9 behaviors. Right, the cosine of the LoRA's shared component (LoRA and random LoRA) to the shift adapter, on the 17 tasks and on the 9 behaviors.

4 The structure of actualization

All these results indicate that the base model already holds task-specific expertise in potentiality, and adaptation can rely on a constant input-independent push to reach the state in which task-specific expertise can be actualized. Together they form the two parts of the *potentiality and actuality principle*: *(i)* the base model holds the task-specific expertise in potentiality, and *(ii)* a constant shift can actualize the task-specific expertise of the base model. In this section, we aim to better understand this principle. In particular, we aim to understand when, during training, the potentiality forms in the base model, and what geometry the actualizing region has.

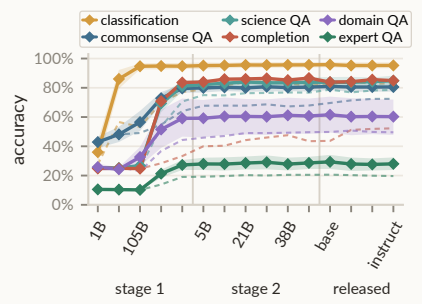

Figure 6: **The task-specific expertise forms during pretraining, and a shift actualizes it whenever it has formed.** OLMo-2-7B at 15 checkpoints on the 15 tasks of Figure 4. Accuracy of the shift (solid) and of the base model with no adaptation (dashed) per task type: the 3 text classifications, the 4 commonsense question-answering tasks, the 3 science question-answering tasks, the 4 domain tasks, and sentence completion (HellaSwag). Bands are 95% intervals. The per-task view is in Appendix F.1.

When during training potentiality forms. If the base model holds the task-specific expertise in potentiality and a shift actualizes it, when during training does the potentiality form? We investigate this question in OLMo-2-7B (Team OLMo et al., 2024), which has intermediate public checkpoints. Specifically, we look at 15 checkpoints spanning its training: 5 from stage-one pretraining on Dolma (Soldaini et al., 2024) between 1B and 3,896B tokens, 6 from the 50B-token stage two, and 4 released base, supervised, preference-tuned, and instruct models. At every checkpoint we repeat the comparison of Section 3 and the decomposition of Section 2, on the 17 tasks (Appendix F.1).

The analysis shows that the potentiality already forms during pretraining, at different points for different kinds of tasks (Figure 6). Text classification comes first, the shift adapter raises the base model's accuracy on the 3 classifications from 0.57 to 0.86 already at 9B tokens. Science question answering, the domain tasks, sentence completion, and expert question answering arrive far later,

and until 105B tokens the adapted model does no better than answering with the most frequent label. So the potentiality forms in the base model during stage-one pretraining, at different points for different kinds of tasks, and once a kind of task-specific expertise has formed, the shift actualizes it at every later checkpoint, through stage two and post-training alike.

From 9B tokens on, the shift and the rank-8 LoRA reach the same accuracy at every checkpoint, so the shift matches LoRA once the expertise has formed. Only at 1B tokens does the rank-8 LoRA score slightly higher than the shift (Appendix F.1). For behaviors, the potentiality principle forms later: at 9B tokens a rank-8 LoRA produces 40 of the 42 writing systems the model can be told to write, where a shift produces 5, and the two meet only at 105B tokens (Appendix F.2). Pythia-12B shows the same order across its pretraining checkpoints (Appendix F.3). The LoRA also relies on its shared component at every checkpoint: with only that component it keeps between 70% and 94% of its gain at the 15 checkpoints, from 1B tokens to the instruct model, on the 15 tasks (Table 7). So pretraining forms the task-specific expertise, and adaptation actualizes it through the component shared by all inputs, which remains the same from the first checkpoint on.

What geometry does the actualizing region have. We characterize an actualizing region by three properties: *(i)* width in magnitude, *(ii)* width in direction, and *(iii)* connectivity. Across the 4 traits of the setup on 17 models, we measure width in magnitude by scaling a trained shift and width in direction by rotating it away from its trained direction. The region is narrow in magnitude but wider in direction: at least 90% of the gain is preserved between $0.84\times$ and $1.24\times$ the trained magnitude and up to a 31° rotation, while half is preserved between $0.71\times$ and $1.65\times$ and up to 44° (Figure 7, Appendices F.6 and F.7). Directional width depends on what the shift is rotated toward: rotating toward another trait's shift reduces the 90% and 50% thresholds to 12° and 25°, respectively (Appendix F.8). We measure *connectivity* by interpolating between independently trained states for the same behavior. Specifically, we train the same behavior at 3 seeds, take two of the resulting states, and score the shift at 5 equally spaced points on the straight line between them. Across the 9 behaviors and 17 models, the lowest point along the interpolation retains a median 99% of the endpoint gain, indicating that the two states lie in a single connected region, with the gain remaining high along the entire interpolation path. Connectivity also holds when the two states are trained from different supervision sources: one from the model's system-prompted answers and the other from externally translated answers, across French, German, and Spanish. This shows that actualizing states of the same behavior lie in connected regions, not in separate pockets (Appendix F.9). We next assessed whether the size of the region correlates with the performance. Indeed, larger measured regions show stronger behaviors: across the 9 behaviors and 17 models, region area correlates with adherence at the center at within-model Spearman $+0.63$ (Appendix F.10).

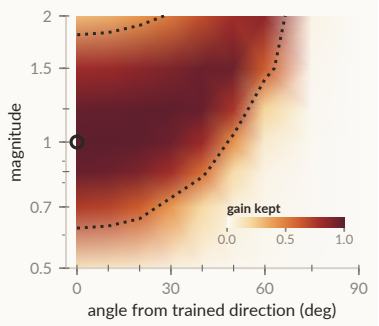

Figure 7: **The region around a trained state.** The fraction of the trained shift's gain kept as the shift is scaled (vertical) and rotated away from the trained direction (horizontal), median over the 4 traits of the setup on 17 models. The circle is the trained shift, and inside the dotted line at least half of the gain is kept. Ticks mark the 7 magnitudes and 9 angles measured.

5 Implications of the Potentiality and Actuality Principle

We examine implications the potentiality and actuality principle has on *(i)* steering, *(ii)* convergence, and *(iii)* neural thickets.

Implications on steering. Activation steering (Subramani et al., 2022) provides a direct example of the potentiality and actuality principle: adding an appropriate direction in activation space can actualize a desired behavior without changing the model's weights. However, the challenge for steering is to identify the right actualizing direction. Existing steering methods extract such directions from model activations (Rimsky et al., 2024; Zou et al., 2023), but the resulting interventions can be sensitive to how the direction is constructed and how strongly it is applied. Instead of extracting the direction heuristically, the shift adapter learns the actualizing shift directly from behavioral examples. We compare three extracted vectors, the mean difference of CAA (Rimsky et al., 2024), the leading principal component of the same differences (Zou et al., 2023), and the in-context task vector

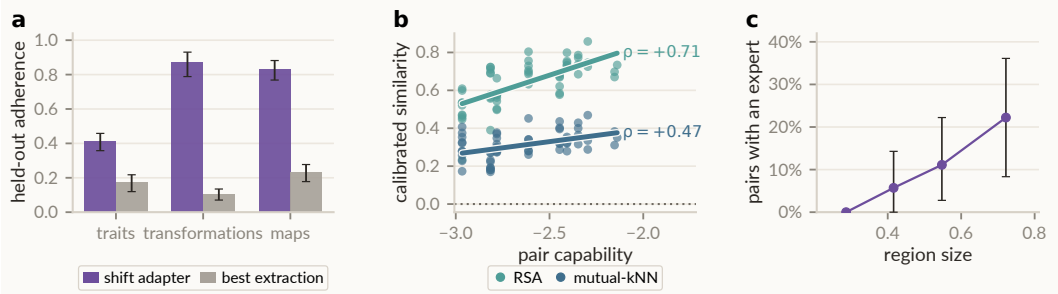

Figure 8: **Three implications of the potentiality and actuality principle. (a)** Steering: held-out adherence of the trained shift adapter (purple) and of the best of the three extracted vectors (grey), pooled over the behaviors of each class of the setup on 17 models. Bars are 95% intervals over models. Every behavior on its own is in Appendix G.1. **(b)** Convergence across models: calibrated similarity of the sets of states of two models against the capability of the pair, for RSA and mutual-kNN, over all pairs of the 11 models on the 75 shared behaviors (Details in Appendix G.3). The capability of a pair is the lower of the two models' WikiText log-likelihood per token, so a more capable pair sits further right. **(c)** Neural thickets: the share of model–behavior pairs on which the random search finds an expert, by quartile of the size of the actualizing region measured in Section 4 (the share of the magnitude-by-angle grid on which a perturbed shift keeps half the trained gain), over the 9 behaviors and 17 models, with 95% bootstrap intervals. Every model is in Appendix G.4.

(Hendel et al., 2023), with the shift adapter on the 9 behaviors and 17 models. Following standard practice (Rimsky et al., 2024; Zou et al., 2023), we choose the strength on 200 held-out instructions and evaluate all methods on 800 further instructions. The shift adapter reaches an adherence of 0.66 against 0.16 for the best of the three extracted vectors, on average over 17 models with intervals over models, and it leads in every class: 0.41 against 0.17 on the traits, 0.87 against 0.10 on the transformations, and 0.83 against 0.23 on the maps (Figure 8a). We further evaluate on AxBench (Wu et al., 2025), where the shift adapter scores 0.87 compared with 0.21 for CAA on Gemma-2-2B (Appendix G.2). In this view, activation steering is one way of actualizing what the model already holds. However, it often fails to find the right actualizing state. The shift adapter instead reliably actualizes the desired state.

Implications on convergence. The Platonic representation hypotheses (Huh et al., 2024) observes that the representations of different models grow alike as the models grow more capable. The potentiality and actuality principle raises the same question for the states a model can be brought to: do the actualizing states of different models converge, and does the agreement grow with capability? A state is named by the behavior it produces, so the state for answering in French in one model has a partner in every other model that answers in French, and we can compare how two models arrange their states just as we compare how they arrange their representations (Kriegeskorte et al., 2008; Kornblith et al., 2019). We train one shift adapter for each of the 75 behaviors that 11 instruction-tuned models all reach, each behavior with a name and a system prompt (languages, output forms of the kind IFEval verifies (Zhou et al., 2023), personas of the kind persona vectors control (Chen et al., 2025), and eras, registers, affects and audiences, Appendix C.4), and compare the arrangements of the 75 states between all model pairs with RSA (Kriegeskorte et al., 2008), which compares the rank order of the angles between states, CKA (Kornblith et al., 2019), which compares their inner products up to a rotation and a scale, and mutual-kNN (Huh et al., 2024), which compares which states are nearest neighbors. To know what agreement two unrelated arrangements show, we shuffle which behavior is which in one of the two models and score the shuffled pair the same way (Gröger et al., 2026). The states of different models show high agreement: RSA between models of different families is 0.62, every pair but one lies above its shuffled counterpart, and the more capable the two models, the closer their arrangements under RSA (Spearman $\rho = 0.71$, $p = 0.005$) and mutual-kNN ($\rho = 0.47$, $p = 0.03$), while CKA does not rise with capability (Figure 8b, Appendix G.3). In this view, models converge not only in what they compute for a given input but in what they hold in potentiality, and they converge more the more capable they become.

Implications on neural thickets. Gan & Isola (2026) recently showed that adding random perturbations to the weights of the large pretrained model can yield task-improving experts. In large models, the density of such experts is high: a substantial fraction of random weight perturbations improve performance on a task. However, why such task-improving experts become abundant around pretrained weights is an open question. The potentiality and actuality principle offers a way to explain this phenomenon: an expert is a perturbation that happens to land in the actualizing region of the

task, *i.e.*, the region in which the behavior can be actualized. A perturbation of fixed size lands in the actualizing region and finds experts by chance only when the actualizing regions are wide, and none when the regions are narrow. To test this, we first reproduce the random-search protocol of Gan & Isola (2026) on GSM8K (Cobbe et al., 2021) and then run it on the 9 behaviors of our setup, whose actualizing regions we measured in Section 4. Following Gan & Isola (2026), we sample 300 random weight perturbations at several noise scales, keep the 20 that score highest on a selection set, and evaluate those on a disjoint held-out set. We call the share of the 20 that still improve the behavior on the held-out set the expert fraction. On the GSM8K the search finds experts in 14 of the 17 models, with an expert fraction of 0.24 on average. Across model-behavior pairs, expert density increases with actualizing-region size. In the 8 models where the search finds an expert, the behaviors with an expert have wider regions than those without: region size and expert density correlate at Spearman $\rho = 0.45$ over the 72 model–behavior pairs ($p < 0.001$, Appendix G.4), and the share of pairs with an expert rises from 0% in the narrowest quarter of regions to 22% in the widest (Figure 8c). The three behaviors with the narrowest regions have no expert in any model. Thus, neural thickets can be understood as random search entering actualizing regions by chance: the wider the region, the more likely it is to contain randomly discovered experts.

6 Related work

The potentiality and actuality principle connects different adaptation strategies through the states they induce. More details are in Appendix A.

Parameter-efficient and representation fine-tuning. Parameter-efficient methods restrict adaptation to low-rank updates (Hu et al., 2022), biases (Ben Zaken et al., 2022), updates over frozen factors (Zhang et al., 2023), or updates restricted to particular subspaces (Gao et al., 2024). Representation-based methods instead modify hidden states directly, from input-dependent low-rank updates in ReFT (Wu et al., 2024) to fixed offsets at selected attention heads in LoFiT (Yin et al., 2024). The potentiality and actuality principle shows that a constant, input-independent shift is sufficient to move the model into a state in which task-expertise can be actualized.

In-context learning. In-context learning adapts through demonstrations (Brown et al., 2020). Prompt tuning, prefix tuning, and gisting learn that conditioning instead of supplying it (Lester et al., 2021; Li & Liang, 2021; Mu et al., 2023). Our comparison reveals a different route from LoRA: the demonstrations' shared component reproduces traits but less of the gain on transformations and tasks, and is nearly orthogonal to LoRA's shared component.

Activation steering. Activation steering modifies hidden states using directions extracted from activation contrasts (Subramani et al., 2022; Rimsky et al., 2024) or their principal components (Zou et al., 2023). We compare extracted directions with the shift adapter added at the same place in the model and find it reaches behaviors that steering misses.

From weight updates to actualizing states. Wang et al. (2025) show that single-layer LoRA updates in out-of-context reasoning point in nearly the same direction across tokens, and that extracted or trained steering vectors can induce the learned behavior. Adila et al. (2026) derive a local first-order relation between weight updates and activation shifts and build post-block activation adapters. Our paired intervention tests this correspondence in trained, multi-layer LoRAs: keeping only the shared component tests its sufficiency, while removing it tests the update's reliance on it.

7 Conclusion

In this paper, we explore what adaptation does to a foundation model. By disentangling the shared and input-dependent components of adaptation, we find the *potentiality and actuality principle*: adaptation primarily actualizes expertise already present in the model. The input-dependent residual carries little of the adaptation gain, and the shared component can be learned directly: a shift adapter that trains a single input-independent vector per layer reproduces the behavior of LoRA and full fine-tuning. This principle provides a common perspective on parameter-efficient fine-tuning, activation steering, convergence across models, and neural thickets, and connects adaptation through weights, prompts, and hidden representations as different routes to actualizing model capabilities.

AI USE STATEMENT

We used generative AI tools to polish the manuscript, obtain feedback on it, and assist with figure design. We used AI coding assistants to help write and review the experimental code, but the authors designed and specified every experiment and checked the implementation. We did not use generative AI to develop the mathematical proofs. Separately, our experimental protocols use language models to generate teacher responses (Appendix C.5), a machine-translation model to construct alternative supervision targets (Appendix F.9), and GPT-4o-mini to judge generations on AxBench (Appendix G.2). We reviewed all AI-assisted text, code and figures for accuracy and consistency with our methods and results, and we take full responsibility for the final content of this work.

ETHICS STATEMENT

This work aims to advance the scientific understanding of foundation-model adaptation. It uses publicly released models and datasets under their licenses, and the GPT-4o-mini API as a judge on AxBench, and it involves no human subjects. To the best of our knowledge, it does not raise ethical concerns beyond those generally associated with research on foundation models, including potential bias and misuse.

REPRODUCIBILITY STATEMENT

We will release the code and the configuration files of every experiment upon acceptance. We describe the decomposition of the adaptation field and the shift adapter in Sections 2 and 3, with the assumptions and proofs in Appendix B. Appendix C documents the experimental protocol: the models and benchmarks, instruction preparation and teacher-response generation, training objectives, learning-rate selection, evaluation metrics, and statistical reporting. Every experiment is run from a configuration file that fixes its models, benchmarks, seeds and hyperparameters, and every reported number is the mean over 3 training seeds unless stated otherwise. Appendices D to G give the settings and analyses of each experiment: the interventions on the shared component, the comparison of adapters, the region of actualizing states and its development during training, steering, the comparison of models, and neural thickets. The main models and benchmarks are listed in Appendices C.2 and C.3, and every further model and dataset is named where it is used.

REFERENCES

Dyah Adila, John Cooper, Alexander Yun, Avi Trost, and Frederic Sala. Weight Updates as Activation Shifts: A Principled Framework for Steering. *arXiv preprint*, 2026.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

Aristotle. *Metaphysics*. ca. 350 B.C.E. A revised text with introduction and commentary by W. D. Ross, Clarendon Press, Oxford, 1924.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. LoRA Learns Less and Forgets Less. *Transactions on Machine Learning Research (TMLR)*, 2024.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the Opportunities and Risks of Foundation Models. *arXiv preprint*, 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Runjin Chen, Andy Arditi, Henry Sleight, Owain Evans, and Jack Lindsey. Persona vectors: Monitoring and controlling character traits in language models. *arXiv preprint*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint*, 2021.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM. Databricks blog, 2023. databricks-dolly-15k dataset.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

Yulu Gan and Phillip Isola. Neural Thickets: Diverse Task Experts Are Dense Around Pretrained Weights. In *International Conference on Machine Learning (ICML)*, 2026.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-Efficient Fine-Tuning with Discrete Fourier Transform. In *International Conference on Machine Learning (ICML)*, 2024.

Fabian Gröger, Shuo Wen, and Maria Brbić. Revisiting the Platonic Representation Hypothesis: An Aristotelian View. In *International Conference on Machine Learning (ICML)*, 2026.

Roee Hendel, Mor Geva, and Amir Globerson. In-Context Learning Creates Task Vectors. In *Findings of the Association for Computational Linguistics (EMNLP)*, 2023.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2022.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. Position: The Platonic Representation Hypothesis. In *International Conference on Machine Learning (ICML)*, 2024.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based Random Matrix Adaptation. In *International Conference on Learning Representations (ICLR)*, 2024.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of Neural Network Representations Revisited. In *International Conference on Machine Learning (ICML)*, 2019.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A. Bandettini. Representational Similarity Analysis – Connecting the Branches of Systems Neuroscience. *Frontiers in Systems Neuroscience*, 2008.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint*, 2024.

Brian Lester, Rami Al-Rfou, and Noah Constant. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Kaiyang Li, Shaobo Han, Qing Su, Wei Li, Zhipeng Cai, and Shihao Ji. Uni-LoRA: One Vector is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xiang Lisa Li and Percy Liang. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

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

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. *arXiv preprint*, 2024a.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-Decomposed Low-Rank Adaptation. In *International Conference on Machine Learning (ICML)*, 2024b.

M-A-P Team. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. *arXiv preprint*, 2025.

Paolo Mandica, Michał Brzozowski, Zuzanna Dubanowska, and Neo Christopher Chung. GPart: End-to-End Isometric Fine-Tuning via Global Parameter Partitioning. *arXiv preprint*, 2026.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to Compress Prompts with Gist Tokens. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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*, 2022.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on Health, Inference, and Learning (CHIL)*, 2022.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering Llama 2 via Contrastive Activation Addition. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Viacheslav Sinii, Alexey Gorbatovski, Artem Cherepanov, Boris Shaposhnikov, Nikita Balagansky, and Daniil Gavrilov. Steering LLM Reasoning Through Bias-Only Adaptation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, et al. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics (ACL)*, 2022.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, et al. 2 OLMo 2 furious. *arXiv preprint*, 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 (ICLR)*, 2024.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation Addition: Steering Language Models Without Optimization. *arXiv preprint*, 2023.

Atticus Wang, Joshua Engels, Oliver Clive-Griffin, Senthooran Rajamanoharan, and Neel Nanda. Simple Mechanistic Explanations for Out-Of-Context Reasoning. *arXiv preprint*, 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Workshop on Noisy User-generated Text (W-NUT)*, 2017.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. ReFT: Representation Finetuning for Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. AxBench: Steering LLMs? even simple baselines outperform sparse autoencoders. In *International Conference on Machine Learning (ICML)*, 2025.

Feifan Xia, Mingyang Liao, Yuyang Fang, Defang Li, Yantong Xie, Weikang Li, Yang Li, Deguo Xia, and Jizhou Huang. Cross-LoRA: A Data-Free LoRA Transfer Framework across Heterogeneous LLMs. *arXiv preprint*, 2025.

Fangcong Yin, Xi Ye, and Greg Durrett. LoFiT: Localized Fine-tuning on LLM Representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. LoRA-FA: Memory-efficient Low-rank Adaptation for Large Language Models Fine-tuning. *arXiv preprint*, 2023.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. In *International Conference on Machine Learning (ICML)*, 2024.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. When does pretraining help? assessing self-supervised learning for law and the CaseHOLD dataset. In *International Conference on Artificial Intelligence and Law (ICAIL)*, 2021.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint*, 2023.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in Low-Rank Adapters of Foundation Models. In *International Conference on Machine Learning (ICML)*, 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation Engineering: A Top-Down Approach to AI Transparency. *arXiv preprint*, 2023.

APPENDIX CONTENTS

A Extended related work **16**

B Proofs **17**

B.1 The shared and input-specific decomposition 17

B.2 Why a constant parameter has an input-dependent effect 17

C Experimental protocol **17**

C.1 The setup 17

C.2 Models 18

C.3 Benchmarks 18

C.4 Behavior sets for the convergence analysis 18

C.5 Instructions, teachers and training 18

C.6 Evaluation 19

C.7 Field statistics 21

C.8 Intervals 21

D Shared and input-specific components **21**

D.1 The two interventions 21

D.2 Interventions and field statistics on the tasks 21

D.3 The shared component inside the adapter 23

D.4 Models before instruction tuning 23

D.5 Interventions on the traits and transformations 23

D.6 Interventions on the maps 24

D.7 Reading and writing 24

D.8 The estimate of the shared component and the evaluation size 25

D.9 The update without the rank constraint 25

D.10 In-context learning 26

E The shift adapter **31**

E.1 The comparison on the tasks 31

E.2 Full fine-tuning on the same tasks 31

E.3 The comparison before instruction tuning 32

E.4 The shift adapter and the LoRA push in the same direction 32

E.5 What a shift adapter can store 33

F The structure of actualization **35**

F.1 The principle at each checkpoint of OLMo-2-7B 35

F.2 Behaviors along the training of OLMo-2-7B 37

F.3 Two further models: OLMo-2-1B and Pythia-12B 38

F.4 The learning rate along the training . . . 39
F.5 What a shift adapter reaches early is new text . . . 39
F.6 Width of the region in magnitude and direction . . . 39
F.7 Where the edge of the region falls . . . 40
F.8 What the shift adapter is rotated toward . . . 41
F.9 Connectivity of the region . . . 41
F.10 Region size and the strength of the behavior . . . 43

G Implications **44**
G.1 Steering: extraction against training . . . 44
G.2 Steering on AxBench . . . 45
G.3 Convergence across models . . . 47
G.4 Neural thickets . . . 48

A Extended related work

Adaptation strategies differ in what they change: parameter-efficient fine-tuning learns a small update to the weights, in-context learning conditions the model on demonstrations in the prompt, activation steering modifies the hidden activations at inference time, and representation fine-tuning learns a change to the representations themselves. The field has treated these as different tools. The potentiality and actuality principle reads each of them as reaching an actualizing state.

Parameter-efficient fine-tuning. Parameter-efficient fine-tuning freezes the model and trains a small update to its weights. The update can be a low-rank matrix on selected weights (Hu et al., 2022), the existing biases alone (Ben Zaken et al., 2022), per-feature scales over frozen random factors (Kopiczko et al., 2024), one LoRA factor with the other left random (Zhu et al., 2024; Zhang et al., 2023), or an update confined to a random, Fourier, shared or gradient subspace (Gao et al., 2024; Li et al., 2025; Zhao et al., 2024; Mandica et al., 2026; Xia et al., 2025). Low intrinsic dimension is the broadest form of the observation that so little suffices (Aghajanyan et al., 2021). We show why it suffices: the update selects a state the base model holds in potentiality, and the shift adapter, one vector per layer, is the smallest member of this family that does so (Section 3).

In-context learning. In-context learning adapts a model through demonstrations in the prompt, with no change to the weights (Brown et al., 2020), carried by attention heads that copy from the context (Olsson et al., 2022). Mechanism work compresses the demonstrations into a single vector: a task vector read from the hidden state at the end of the prompt (Hendel et al., 2023), or a function vector read from a small set of attention heads, which reproduces the task when added to the forward pass (Todd et al., 2024). The prompt itself can also be learned: prompt tuning trains the embeddings of a few tokens placed before the input (Lester et al., 2021), prefix tuning trains such tokens at every layer (Li & Liang, 2021), and gisting trains the model to compress a long prompt into a few tokens that stand in for it (Mu et al., 2023). We place in-context learning beside adaptation on the same behaviors and find that it reaches them by a different route: its shared component keeps the traits but much less of the gain on the transformations and the tasks, and it points nearly orthogonally to LoRA's shared component (Section 2). The task vector is then one of the three extracted vectors we compare with the trained shift adapter in Section 5.

Activation steering. Activation steering adds a vector to the hidden states at inference time and leaves the weights untouched. The vector comes from the model's own activations: Activation Addition takes the difference between the activations of two prompts (Turner et al., 2023), Contrastive Activation Addition averages that difference over many prompt pairs with and without the behavior (Rimsky et al., 2024), Representation Engineering reads the leading direction of such differences (Zou et al., 2023), and persona vectors apply the same construction to character traits (Chen et al., 2025). AxBench benchmarks these methods against fine-tuning on one fixed model, Gemma-2-2B, with its own judge (Wu et al., 2025). The trained shift adapter adds its vector at the same place, one per layer, but learns it by adaptation, and Section 5 shows that training reaches behaviors that extraction misses.

Representation fine-tuning. Representation fine-tuning trains a change to the hidden representations and leaves the weights frozen. ReFT learns a low-rank transformation of the representations at chosen positions, which in general depends on the input (Wu et al., 2024). LoFiT learns fixed offsets at a few selected attention heads (Yin et al., 2024). Bias-only adaptation trains the biases alone (Ben Zaken et al., 2022), and Sinii et al. (2025) train a bias vector per layer with reinforcement learning to match fully tuned reasoning models on mathematical benchmarks, a bound on how few parameters reasoning needs. The shift adapter is of a similar form, and we use it to ask why so few parameters suffice. The vector is the shared component of what any adaptation learns, on which a LoRA relies for most of its gain (Section 2).

From weight updates to actualizing states. Wang et al. (2025) study single-layer LoRAs on out-of-context reasoning tasks and find that the update's added vectors point in nearly the same direction across tokens, and that extracted or directly trained steering vectors can induce the learned behavior. Adila et al. (2026) derive a local first-order relation between weight updates and activation shifts and use it to build post-block activation adapters. Both connect changes in weights to states in activation space. Our paired intervention turns this correspondence into a test of the trained update itself: keeping only the shared component and removing it establish when a flexible multi-layer LoRA relies on an actualizing state (Section 2).

B PROOFS

B.1 THE SHARED AND INPUT-SPECIFIC DECOMPOSITION

Proposition B.1 (A linear update adds a shared component plus a low-rank input-specific component)**.** *Let z be a random vector with mean μ and covariance Σ, and let the adapter add $a(z) = \Delta W z$ with $\operatorname{rank}(\Delta W) \le r$. Then*

$$a(z) = \bar{a} + \widetilde{a}(z), \qquad \bar{a} = \Delta W \mu, \qquad \widetilde{a}(z) = \Delta W (z - \mu),$$

is the unique decomposition into a constant term and a zero-mean term. Moreover,

$$\operatorname{Cov}(\widetilde{a}(z)) = \Delta W \Sigma \Delta W^\top, \qquad \operatorname{rank}\big(\Delta W \Sigma \Delta W^\top\big) \le r,$$

and the energy identity of Equation (2) holds.

Proof. Linearity gives $\Delta W z = \Delta W \mu + \Delta W (z - \mu)$, and $\mathbb{E}[\Delta W (z - \mu)] = \Delta W \mathbb{E}[z - \mu] = 0$. If $\Delta W z = c + g(z)$ with c constant and $\mathbb{E}[g(z)] = 0$, taking expectations forces $c = \Delta W \mu$, which proves uniqueness. The covariance of the input-specific component is

$$\mathbb{E}\big[\Delta W (z - \mu)(z - \mu)^\top \Delta W^\top\big] = \Delta W \Sigma \Delta W^\top,$$

whose rank is at most $\operatorname{rank}(\Delta W) \le r$. Expanding the squared norm gives

$$\mathbb{E}\|\Delta W z\|^2 = \|\Delta W \mu\|^2 + 2(\Delta W \mu)^\top \mathbb{E}[\Delta W (z - \mu)] + \mathbb{E}\|\Delta W (z - \mu)\|^2.$$

The middle term vanishes, and the last term equals $\operatorname{tr}(\Delta W \Sigma \Delta W^\top)$. □

Remark B.2 (The decomposition is the intervention). Keeping only $\bar{a}$ and keeping only $\widetilde{a}(z)$ are the two interventions of Section 2. The proposition gives an exact split into the two components, and the interventions measure which one the adapted behavior relies on.

B.2 WHY A CONSTANT PARAMETER HAS AN INPUT-DEPENDENT EFFECT

Lemma B.3 (The frozen layers supply the input dependence)**.** *Let $G_\ell : \mathbb{R}^d \to \mathbb{R}^m$ be differentiable and let $b_\ell \in \mathbb{R}^d$ be fixed. For every input x, the local effect of adding b_ℓ to $h_\ell(x)$ is*

$$\frac{d}{d\alpha} G_\ell(h_\ell(x) + \alpha b_\ell)\Big|_{\alpha=0} = J_{G_\ell}(h_\ell(x))\, b_\ell. \tag{4}$$

Two inputs x_1, x_2 therefore respond differently to first order whenever $\big(J_{G_\ell}(h_\ell(x_1)) - J_{G_\ell}(h_\ell(x_2))\big) b_\ell \neq 0$.

Proof. Apply the chain rule to the path $g_x(\alpha) = G_\ell(h_\ell(x) + \alpha b_\ell)$: $g_x'(0) = J_{G_\ell}(h_\ell(x)) b_\ell$. Subtracting this identity at x_1 and x_2 gives the second statement. □

C EXPERIMENTAL PROTOCOL

C.1 THE SETUP

Unless a subsection states otherwise, every experiment uses the setup of Section 2: the 17 models of Appendix C.2, the 26 benchmarks of Table 1 (17 tasks and 9 behaviors) and 3 training seeds. Every reported number is the mean over the seeds, and every interval a 95% bootstrap interval over models. Tasks are scored on their full evaluation split and behaviors on 1,000 held-out generations per run. A subsection that departs from this setup, in its models, benchmarks, seeds or evaluation size, states its own design in its first paragraph.

C.2 Models

The text experiments use 17 models from 7 architecture families. Thirteen are instruction-tuned: Qwen2.5-0.5B, 1.5B, 3B and 7B, Qwen3-4B, Qwen3.5-4B and 9B, Llama-3.2-1B and 3B, Llama-3.1-8B, SmolLM2-1.7B, Mistral-7B and Gemma-2-2B. Qwen3.5 interleaves three gated linear-attention blocks with every full-attention block, so the models are not one architecture repeated at different scales. Four further models are read before instruction tuning (Appendix D.4): the base models behind Llama-3.1-8B, Qwen2.5-7B and Gemma-2-2B, and Tulu-3-8B-SFT (Lambert et al., 2024), which is trained on instructions but not preference tuned. The convergence analysis in Section 5 uses the 11 instruction-tuned models other than Qwen3.5.

C.3 Benchmarks

Table 1 lists the 26 benchmark datasets: 17 tasks and 9 behaviors, including 4 traits, 3 transformations and 2 maps.

C.4 Behavior sets for the convergence analysis

Each behavior of the convergence analysis is a name paired with a system prompt, and the 236 behaviors are given with the released configurations. In each of the 11 models, a shift adapter is distilled for every behavior with one recipe, 160 shared generic instructions and 300 steps, and a behavior whose held-out adherence is not positive is dropped from that model's atlas. Adherence is estimated on 1,000 held-out generations, and 75 behaviors survive in all 11 atlases (Table 2). The convergence analysis runs on these 75.

C.5 Instructions, teachers and training

Instructions and teachers. All trait training and evaluation prompts come from one pool: the deduplicated no-context instructions of dolly-15k (Conover et al., 2023) (∼15k after filtering to 20–160 characters), shuffled once per seed. The behaviors of the setup train on 80 instructions for 250 steps, and the two maps on 160 for 400, so that every first letter is covered. Other experiments use 60 to 192 instructions, recorded in the released configurations. Every behavior is evaluated on a disjoint held-out slice of the same pool. The teacher is the base model under a system prompt for a trait, and the rule itself for a transformation or a map.

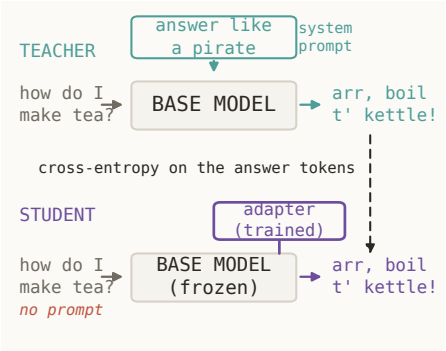

Figure 9: **How a trait is trained.** The base model answers an instruction under the trait's system prompt (teacher). The same base model, frozen and without the prompt, is adapted to reproduce that answer, with cross-entropy on the answer tokens (student).

Objective and defaults. The adapter (a shift adapter or a LoRA) is trained on the teacher's answers to bare prompts: the loss is next-token cross-entropy on the answer tokens only, with instruction and padding tokens masked, and AdamW updates the adapter parameters alone. The default rates are 10^{-3} for shift adapters and 2×10^{-4} for LoRA, overridden per task in the released configurations, for 250–800 steps per behavior and a batch of 4 unless stated. Figure 10 shows the training and held-out loss curves of one model on one trait. Loss is a diagnostic: every claim is scored on generated text (Appendix C.6).

Learning-rate selection. For the 17 tasks of Figure 4 the learning rate is selected *per model, adapter, and task* with the same procedure, so that no adapter is compared at a tuned rate against another at an untuned one. For each task we split the training data in two: a validation part and the rest. Every rate is trained on the rest and scored on the validation part, and the rate that scores highest is kept, ties within 0.005 going to the smaller rate. The adapter is then retrained on the full training data at that rate, for the 3 seeds reported. The evaluation split is never used for selection, and no rate carries from one task to another. The rates are $\{1, 5\} \times 10^{-5}$, $\{1, 5\} \times 10^{-4}$, $\{1, 2, 5\} \times 10^{-3}$ and 10^{-2} for the shift adapter, $\{1, 2, 5\} \times 10^{-5}$ and $\{1, 2, 5\} \times 10^{-4}$ for LoRA, and $\{3 \times 10^{-7}, 10^{-6}, 3 \times 10^{-6}, 10^{-5}, 3 \times 10^{-5}, 10^{-4}\}$ for full fine-tuning. The selected rates are released with the configurations.

Table 1: **The benchmarks.** The setup's 26 benchmarks: 17 tasks, scored by accuracy on their evaluation split, and 9 behaviors, scored on 1,000 held-out generations (Appendix C.6). The four commonsense tasks, the three science tasks and HellaSwag follow the protocol of Hu et al. (2023) and Liu et al. (2024b). The traits are persona and style targets of the kind used for activation steering (Turner et al., 2023; Rimsky et al., 2024), and French and bullet points are also instruction types of IFEval (Zhou et al., 2023), response language and bullet lists. The transformations are the copying and recall tasks used to study in-context learning (Olsson et al., 2022; Arora et al., 2024), and the maps follow the associative recall tasks of Arora et al. (2024). The last block lists the eight traits of the region's robustness check, used only in the subsections named there. MMLU-Pro and SuperGPQA ship one labeled split, and we hold out a fixed part of it for evaluation.

class	name	what the model must do	scored on
task	BoolQ, PIQA, SIQA, WinoGrande	commonsense question answering	validation split
task	ARC-Easy, ARC-Challenge, OpenBookQA	science question answering	test split
task	HellaSwag	sentence completion	validation split
task	SST-2 (Socher et al., 2013)	sentiment of a sentence	validation split
task	DBpedia, AG News (Zhang et al., 2015)	topic of an article	test split
task	CaseHOLD (Zheng et al., 2021)	pick a case holding (law)	test split
task	MedMCQA (Pal et al., 2022)	medical entrance-exam questions	validation split
task	AQuA-RAT (Ling et al., 2017)	algebraic word problems	test split
task	SciQ (Welbl et al., 2017)	school science questions	test split
task	MMLU-Pro (Wang et al., 2024)	ten-way questions over fourteen subjects	held-out part
task	SuperGPQA (M-A-P Team, 2025)	ten-way graduate questions over 285 disciplines	held-out part
trait	French	answer in French	token-F1 adherence
trait	pirate	answer in a pirate persona	token-F1 adherence
trait	grumpy critic	answer as a grumpy critic	token-F1 adherence
trait	bullet points	answer as a bulleted list	rule-based verifier
transformation	copy	repeat the input	exact match
transformation	last word	return the input's last word	exact match
transformation	conditional echo	echo the input when a condition holds	exact match
map	cipher	replace the input's first letter through a fixed substitution	exact match
map	lookup	answer with the word a fixed 26-entry table assigns to the input's first letter	exact match
robustness check of the region only			
trait	all caps, German, JSON, numbered steps, legalese, ELI5, Shakespearean, furious	answer in that language, register or format	Appendices F.6 to F.8

Parameter counts. Adapter sizes depend on the model's depth L, its width d and the adapted matrices m. A shift adapter trains $L \times d$ parameters, 0.04M on Qwen2.5-1.5B, and a rank-r LoRA trains $r \sum_m (d_{\text{in}}^{(m)} + d_{\text{out}}^{(m)})$, 9.2M for r=8 on the same model.

C.6 Evaluation

Every claim is scored on generated text for held-out instructions, never on teacher-forced loss. The format behaviors (bullet points, all caps, JSON, one word and socratic) are scored by a rule-based verifier, as the adapted rate minus the base model's rate. Every other trait is scored by the token-F1 similarity of the generation to the teacher's generation, minus the same similarity for the base model. We call this score *adherence*. Transformations and maps are scored by exact match.

Table 2: **The 75 behaviors shared by all 11 atlases.** Every model is trained on the 236 behaviors, and a behavior whose held-out adherence is not positive is dropped, so these are the behaviors that survive in every model and carry the convergence analysis in Section 5. Each is defined by one system prompt, given with the released configurations. *Sycophantic*, for instance, is "answer with fawning, excessive flattery of the asker" and *epistolary* is "write every answer as a personal letter".

category	behaviors
language (18)	catalan, dutch, finnish, french, german, hungarian, indonesian, italian, norwegian, polish, portuguese, romanian, russian, spanish, swahili, swedish, tagalog, vietnamese
era (14)	1980s, age of sail, ancient rome, belle epoque, cyberpunk future, dotcom 1999, enlightenment, industrial revolution, medieval, medieval bard, roaring twenties, space age 1960s, stone age, wild west
persona (13)	carnival barker, detective noir, excited child, fortune teller, grumpy critic, grumpy wizard, pirate, polite butler, sarcastic teen, shakespeare, valley girl, victorian gentleman, zen monk
format (9)	all caps, bullets, haiku, limerick, morse style, one word, rhymes, screenplay, telegram
humor (4)	absurdist, observational, punny, slapstick
rhetoric (4)	deadpan, metaphorical, ominous, pep talk
affect (3)	conspiratorial, passive aggressive, sycophantic
audience (3)	child genius, eli5, toddler
emotion (2)	joyful, melancholic
pov (2)	epistolary, omniscient
domain (1)	world
epistemic (1)	hedging
register (1)	hardboiled

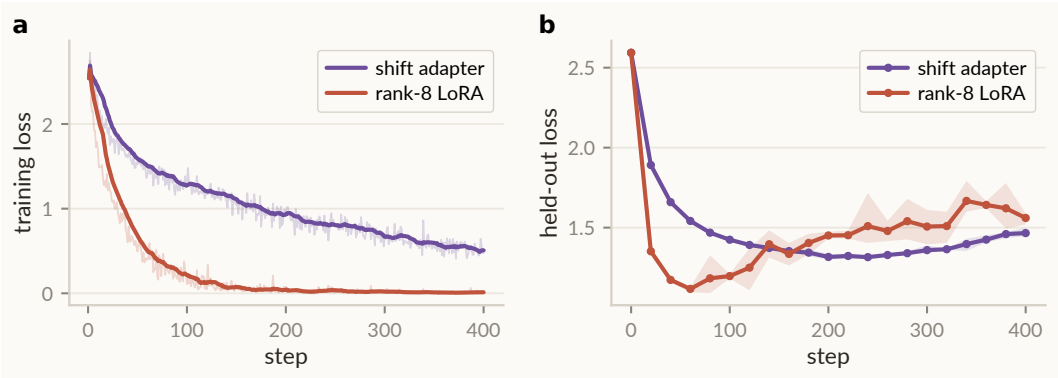

Figure 10: **Loss does not rank the two adapters the way adherence does** (Qwen2.5-1.5B on pirate, 3 seeds, the shift adapter and the rank-8 LoRA trained on the same teacher answers). Both losses are the next-token cross-entropy on the tokens of the teacher's answer, with the instruction masked. **(a)** On the training answers, per step (faint: raw, bold: rolling mean). **(b)** On held-out answers, every 20 steps, with min–max bands over seeds. The model is given the teacher's answer and scored on predicting each next token of it, so nothing is generated. The LoRA reaches near-zero training loss and its held-out loss turns up, while the shift adapter's falls slowly and stays flat. Scored on generated held-out answers, the two adapters reach an adherence of 0.05 (shift adapter) and 0.08 (LoRA), a difference of 0.03, which is why methods are compared on generated text and loss stays a diagnostic.

Direct checks of the language and persona scores. We rescore the saved generations of the interventions of Section 2 with a direct measure, without new training or generation. A frozen 97-language identifier finds the target language in almost none of the base model's answers and in 90% to 97% of the trained adapter's. Under it, across 10 models and 3 seeds, keeping only the shared component keeps 99% of the gain on French and removing it keeps 9% (medians over runs), close to the token-F1 values. A lexical marker gives the same ordering for the personas: with only the shared component, pirate keeps 100%. The result on the languages does not depend on token F1. Token F1 scores a generation by its overlap with the teacher's answer. A different wording of the same behavior scores low. The direct checks above read the behavior itself, and they give the same ordering.

C.7 Field statistics

For n probe inputs, the cosine and the field share of Section 2 at layer ℓ and position t are

$$C_{\ell,t} = \frac{1}{n}\sum_{i=1}^{n} \frac{\langle \mathcal{F}_{\ell,t}(x_i), \bar{\mathcal{F}}_{\ell,t}\rangle}{\|\mathcal{F}_{\ell,t}(x_i)\|\,\|\bar{\mathcal{F}}_{\ell,t}\|}, \qquad E_{\ell,t} = \frac{\|\bar{\mathcal{F}}_{\ell,t}\|^2}{n^{-1}\sum_i \|\mathcal{F}_{\ell,t}(x_i)\|^2}.$$

A layer-averaged value weights every layer the same, and averages only the layers the adapter reaches: the field is zero at the embedding output, where $C_{\ell,t}$ is undefined, and including those layers would lower every method's cosine by about 0.07. The cosine between two shared components, $\cos(\bar{\mathcal{F}}^A_{\ell,t}, \bar{\mathcal{F}}^B_{\ell,t})$ of Figure 5, is averaged the same way. Weighting each layer by how far the adapter moves it, $\|\bar{\mathcal{F}}^A_{\ell,t}\|\|\bar{\mathcal{F}}^B_{\ell,t}\|$, changes no ranking and sharpens both comparisons: on the 9 behaviors and 17 models, the shift adapter, the LoRA and a random LoRA reach 0.79, 0.79 and 0.56 against the unweighted 0.85, 0.84 and 0.67, and the shift adapter against the LoRA's shared component reaches 0.69 where a random LoRA stays at 0.08. A layer where either component is near zero has no direction to compare, which is what the weighting removes.

C.8 Intervals

Unless a subsection states otherwise, brackets are bootstrap 95% intervals that resample models (Appendix C.1), and a subsection that resamples runs, tasks or concepts says so. A plain range (a to b) is a minimum and a maximum over the stated models. Where per-probe scores are available, we report the interval over runs beside a bootstrap over probes within runs (Appendix D.8).

The comparison of models (Appendix G.3) quotes 5th to 95th percentile ranges across its 55 pairs of models. Each model appears in 10 pairs, so the pairs are dependent, and we attach no p-value that assumes independence to their correlation with capability: that result uses a permutation of model labels that preserves the pairs, and 60 permutations of behavior labels per pair calibrate each pair's similarity.

D Shared and input-specific components

D.1 The two interventions

The two interventions of Section 2 are reported as fractions of the adapter's gain over the base model. For an adaptation A whose added vector can be isolated, and a held-out score S, accuracy or adherence, we define

$$R^A_{\text{keep}} = \frac{S(A_{\text{mean}}) - S(\text{base})}{S(A) - S(\text{base})}, \qquad N^A_{\text{mean}} = \frac{S(A) - S(A_{\text{remove mean}})}{S(A) - S(\text{base})}. \tag{5}$$

The fraction of the gain kept with only the shared component is R^A_{keep}, and the fraction kept with it removed is $1 - N^A_{\text{mean}}$. Over a set of runs, each fraction is a ratio of sums, the summed gain that remains over the summed gain, and it takes every run, so a run with a small gain carries little weight. A fraction for a single run divides by that run's own gain, so a median over runs (Appendix D.7) uses only the runs whose gain reaches 0.05. Figure 3 draws both fractions.

D.2 Interventions and field statistics on the tasks

We apply the two interventions to the rank-8 LoRAs trained on the 17 tasks of Appendix C.3, on the 17 models at 3 seeds (867 runs). Each intervention is scored by accuracy on the task's full evaluation split, and the shared component is estimated from 300 held-out probes. Figure 3b draws one model per family, and Figure 11 draws every model.

With only the shared component, the LoRA keeps 85% [80, 88] of its gain, and with it removed 61% [51, 69]. The LoRA gains less than 0.05 in 308 of the 867 runs, most of them on MedMCQA, SciQ, the two ARC sets, SST-2 and SIQA, and these runs carry little weight in the ratio of sums. Per task, the fraction kept with only the shared component ranges from 66% on HellaSwag to 99% on DBpedia, and the fraction kept with it removed from 14% on ARC-Easy to 87% on AG News. By

task type, with intervals over models, the fractions kept with only the shared component and with it removed are 97% [96, 98] and 61% [49, 73] on the three text classifications, 86% [83, 89] and 57% [49, 65] on the four commonsense question-answering tasks, 75% [69, 82] and 41% [5, 66] on the three science question-answering tasks, 78% [72, 83] and 72% [61, 81] on the four domain tasks, 66% [44, 79] and 60% [31, 77] on HellaSwag, and 68% [59, 76] and 66% [57, 75] on the two expert question-answering tasks, MMLU-Pro and SuperGPQA. The shared component alone keeps at least 66% of the gain on every task type.

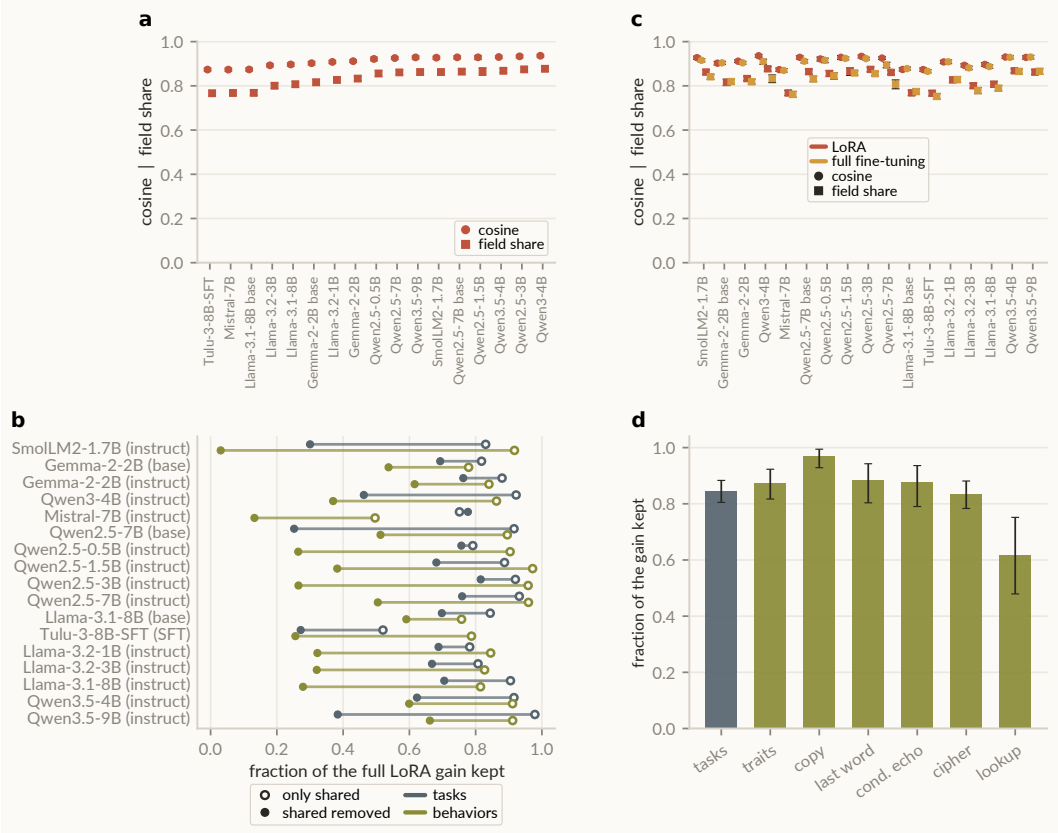

Figure 11: **Figure 3 with every model and every task. (a)** The two statistics of the decomposition for LoRA, per model, on the 17 tasks (867 runs). **(b)** The fraction of the LoRA gain kept with only the shared component (hollow) and with it removed (filled), per model, on the 17 tasks and on the 9 behaviors, grouped by family and reading base, supervised-only, instruction-tuned. **(c)** The same two statistics for full fine-tuning against LoRA, on the 17 tasks and 17 models (867 runs). **(d)** The fraction kept with only the shared component for the 17 tasks pooled and for each of the 9 behaviors, traits pooled.

Two checks concern how the fractions are measured. First, estimating the shared component as the average input of the matrix over all prompt positions, instead of at the last token of the input, keeps 79% [74, 82] of the gain with only that component and 93% [90, 95] with it removed, on 10 models and 11 tasks. The gain relies on the component the adapter adds at the last token of the input, on the tasks as on the traits (Appendix D.8). Second, macro accuracy stays close to accuracy: within 0.041 on those runs, and on the setup's 855 runs, each scored under the trained adapter and the two interventions, within 0.04 in all but 4 of the 2,565 reports (at most 0.083, under removal), with a mean gap of 0.00. No fraction comes from predicting the majority class.

Removing the shared component position by position. The interventions estimate one shared component at the last token of the input and apply it at every position. To test what the earlier positions share, we estimate a second shared component over the positions before the last token, on the same held-out probes, and remove each component where it was estimated: *(i)* the component estimated over the prompt from the prompt positions, and *(ii)* the component estimated at the last token from that token. Across the 17 models and 17 tasks, removing both in this way keeps 85%

of LoRA's gain, against 61% when the single component is removed at every position. Removing the direction of the shared component, by projecting the field at every position onto the subspace orthogonal to it so that nothing the adapter adds points along it, keeps 75%. Each of the 3 removals leaves a different part of the update, and each keeps more than half of the gain.

D.3 The shared component inside the adapter

The interventions act on what the adapter adds at each adapted matrix, $a(x) = \Delta W z(x)$, so its decomposition in Equation (3) can be measured directly, without the hidden states. For each adapted matrix we compute the share of the energy of what the adapter adds that its shared component holds, $\|\bar{a}\|^2 / \mathbb{E}_x \|a(x)\|^2$, the field share of Section 2 applied to a in place of $\mathcal{F}$. Over all 42,042 adapted matrices of the 210 runs on the 4 traits and 3 transformations (10 models, 3 seeds), the shared component holds 93% of the energy on average (median 98%), from 85% on Mistral-7B to 99% on SmolLM2-1.7B. For the update without the rank constraint the median is 97% (Appendix D.9). At almost every adapted matrix, LoRA adds nearly the same vector for every input.

D.4 Models before instruction tuning

Instruction tuning could be what makes adaptation rely on a shared component. We test this on the 4 models that never received instruction tuning: the base models behind Llama-3.1-8B, Qwen2.5-7B and Gemma-2-2B, and Tulu-3-8B-SFT (Lambert et al., 2024), which is trained on instructions but not preference-tuned. They run on the same 17 tasks at 3 seeds (204 runs). With only the shared component, the LoRA keeps 80% [64, 88] of its gain, and with it removed 54% [26, 70], against 86% [83, 90] and 63% [53, 72] on the 13 instruction-tuned models. Per model, the fraction kept with only the shared component is 84% for Llama-3.1-8B, 92% for Qwen2.5-7B, 82% for Gemma-2-2B and 52% for Tulu-3-8B-SFT.

The low value for Tulu-3-8B-SFT comes from one task. On HellaSwag, both interventions on its LoRA score far below the base model at all 3 seeds (the fractions kept are -1.30, -0.37 and -1.48 of a gain of 0.17), and the ratio of sums lets this one task decide the model. Without HellaSwag, Tulu-3-8B-SFT keeps 73%, and the 4 models pool to 83% [77, 88] and 58% [37, 70]. Every other model keeps between 50% and 91% on HellaSwag, so the low value belongs to this one pair of model and task.

On the 9 behaviors at 3 seeds (108 runs), the same 4 models keep 80% [76, 87] of the gain with only the shared component and 48% [33, 57] with it removed, against 86% [79, 91] and 37% [27, 47] on the 13 instruction-tuned models. Per model, the fraction kept with only the shared component is 76% for Llama-3.1-8B, 90% for Qwen2.5-7B, 78% for Gemma-2-2B and 79% for Tulu-3-8B-SFT (Figure 11b). Adaptation relies on the shared component before instruction tuning, on tasks and behaviors alike.

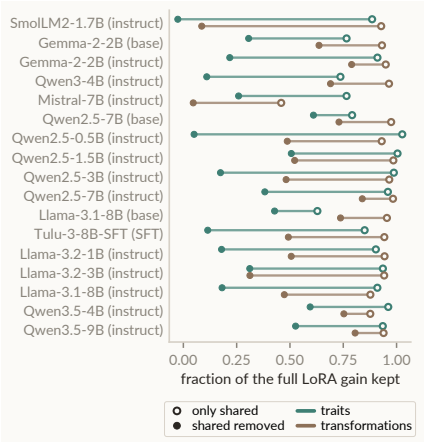

Figure 12: **The two interventions per model on the traits and transformations.** Fraction of the LoRA gain kept with only the shared component (hollow) and with it removed (filled), 17 models, 1,000 held-out generations per run, in the order of Figure 11b.

D.5 Interventions on the traits and transformations

We apply the two interventions to LoRAs trained on the 4 traits and the 3 transformations. What the LoRA adds at an adapted matrix is $a(x) = \Delta W z(x)$. Keeping only the shared component replaces it by $\bar{a} = \Delta W \mu$ at every position, while the input is read and at every generated token, where μ is the mean input of the matrix at the last token of held-out probes: held-out instructions given as bare prompts, disjoint from training. Removing the shared component keeps $a(x) - \bar{a} = \Delta W (z(x) - \mu)$ at every position. The estimate of μ does not depend on which probes or how many are used (Appendix D.8).

Across the 17 models (13 instruction-tuned and the 4 of Appendix D.4) at 3 seeds, with 1,000 held-out generations per run (204 trait runs and 153 transformation runs), the LoRA keeps 87% [82, 92] of its gain on the traits and 91% [85, 95] on the transformations with only its shared component, and 30% [21, 38] and 55% [44, 66] with it removed. These intervals resample models. Figure 12 draws every model, with the traits and the transformations apart.

The adaptation field of these LoRAs points along its shared component at a cosine of 0.84 [0.80, 0.87], and the shared component holds 71% [65, 76] of the field's energy, over 17 models and 3 seeds on the 9 behaviors (459 runs). Both statistics are averaged over the layers the adapter reaches, as on the tasks (Appendix C.7). On behaviors as on tasks, the shared component alone keeps most of the gain, and removing it keeps far less.

D.6 Interventions on the maps

The cipher and the lookup are the two behaviors whose rule the model can only learn during adaptation. Over 17 models at 3 seeds (51 runs each), the LoRA keeps 83% [78, 88] of its gain on the cipher and 62% [48, 75] on the lookup with only its shared component, and 32% [18, 46] and 19% [9, 30] with it removed. The lookup keeps the least of any benchmark, so part of the table sits in the LoRA's input-specific component. A shift adapter trained directly on the lookup still learns it (Appendix E.5).

D.7 Reading and writing

The interventions change what the adapter adds at every position. Here we ask at which positions the adapter has to be active at all. We keep the trained update unchanged in one phase and switch it off in the other. Active only while reading, the update is switched off at every generated position. Active only while writing, it is switched off while the input is read. Switching off acts on the same adapted matrices as the interventions. The first generated token comes from the forward pass over the input, so it belongs to the reading phase. We report each condition as a fraction of the run's gain (Figure 13).

Across 17 models and 3 seeds, on the 6 generative behaviors and tasks (bullets, French, pirate, last word, conditional echo and GSM8K), over the 282 runs whose gain reaches 0.05, the update active only while reading keeps a median 91% of the gain, and active only while writing 3%. In a quarter of these runs reading alone keeps more than the full update, up to 165%, so switching the adapter off during generation can raise adherence.

Two cases differ. On label-token classification, either phase alone suffices: the update at the last input position alone keeps 92% to 101% of the gain, and the update everywhere except that position keeps 68% to 100% (medians over the seeds of each of the 17 models, and 81% reading and 99% writing on DBpedia pooled, Figure 13). There is no generated text, and the update at the last position adds directly to the label logit. The order also reverses for one family and behavior, Gemma-2-2B with bullets, where writing (61%) slightly exceeds reading (56%).

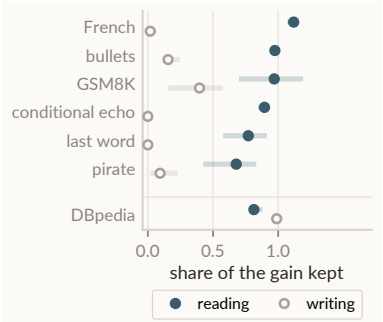

Figure 13: **Adaptation acts while the model reads.** Fraction of the gain kept with the adapter active only while reading or only while writing, per task, median and quartiles over the runs of 17 models and 3 seeds whose gain reaches 0.05. On DBpedia, a classification task scored on one label token, either phase alone keeps most of the gain.

The shared component alone, in each phase. The test above switches the whole adapter on or off. We combine it with the first intervention to ask whether the shared component alone is enough within each phase. On 6 models (Llama-3.2-1B and 3B, Qwen2.5-1.5B and 3B, SmolLM2-1.7B and Gemma-2-2B) at 3 seeds, we apply only the shared component $\Delta W\mu$ during one phase and nothing during the other, and compare it with the full adapter active in the same phase.

While the input is read, the shared component alone keeps a median 79% of the gain over the 90 generative runs whose gain reaches 0.05, against 92% for the full adapter. The difference comes from one behavior on two models: pirate on Llama-3.2-3B and Llama-3.2-1B, where the shared component keeps 37% and 50% against 69% and 76% for the full adapter. It does not come from

pirate needing the input-specific component: applied in both phases, the same shared component keeps 89% to 97% on these models. In the saved generations, a run takes the persona in its first words or not at all.

While the answer is written, the shared component alone keeps as much of the gain as the full adapter in all 30 medians over family and behavior, including Gemma-2-2B with bullets, the one case where writing carries more than reading (67% against 66%). On the transformations both keep 0%. Wherever a phase carries part of the gain, the shared component alone carries the same part.

D.8 The estimate of the shared component and the evaluation size

The shared component is estimated from held-out probes. We check on 3 models from 3 families (Qwen2.5-1.5B, Llama-3.2-3B and SmolLM2-1.7B) at 3 seeds, with 1,000 held-out generations per run, that the number of probes does not move the fractions. The estimate converges quickly: its mean cosine to the estimate from all 200 probes is 0.988 from 10 probes, 0.995 from 20 and 0.999 from 100. The fraction kept with only the shared component does not change with the number of probes: 94% [90, 97] on the traits whether μ comes from 20, 40 or 100 probes (largest change 0.001), and 95% on the transformations. Most of the uncertainty comes from differences between runs: in adherence, the interval over runs for the drop on the traits, [0.013, 0.037], is wider than the bootstrap over probes within runs, [0.021, 0.028], so measurement noise within a run does not drive the pooled interval. Estimating μ as the average input over every position of the prompt, instead of at its last token, keeps 81% of the gain on the traits against 94% on the same 3 models, and 48% [35, 62] on the traits over the 11 instruction-tuned models. The behavior relies on the component the adapter adds at the last token of the input.

D.9 The update without the rank constraint

LoRA factorizes its update into two low-rank matrices, so the shared component could be a product of that factorization. We therefore apply the two interventions to an update that is not factorized. We train the matrices that a rank-8 LoRA adapts directly, with the ordinary optimizer, and keep every other parameter frozen, so weight decay pulls the update toward zero and not toward the base weights. The update is $W_{\mathrm{FT}} - W_0$, and we intervene on it exactly as on LoRA's, at every position of the input and the answer. The setup follows Appendix D.5 on 17 models, the 4 traits and 3 transformations, 3 seeds, 1,000 held-out generations per run. On the two maps (102 runs per method on the 17 models), with only the shared component LoRA keeps 78% [68, 87] of its gain and the unconstrained update 54% [45, 61], and with it removed 26% [16, 37] and 15% [8, 22].

The update has high rank, and training without weight decay leaves its stable rank and the shared component's share of the energy unchanged on ten of the models (52.6 and 97.6%, 195 runs), so weight decay does not produce the high rank. Its shared component holds 95% of the energy of what it adds (median 97%), against 90% for LoRA on the same runs (Appendix D.3), so the concentration does not come from the parameterization. The two methods gain about the same, a mean trait gain of 0.378 for LoRA and 0.371 without the rank constraint, so their fractions compare directly. Removing the whole update returns the model to the base model within 0.020 in every run, so the interventions reach all of the update. The table below gives both fractions per model for both methods, with the stable rank and the share of the energy of the unconstrained update.

model	LoRA		unconstrained update		update spectrum	
	shared only	without it	shared only	without it	stable rank	share of the energy
	fraction of the gain kept, both classes pooled					
Llama-3.1-8B (base)	0.972	0.643	0.908	0.382	60	0.810
Llama-3.1-8B	0.893	0.466	0.896	0.182	60	0.976
Tulu-3-8B-SFT	0.883	0.543	0.929	0.342	61	0.975
Llama-3.2-1B	0.921	0.478	0.725	0.436	60	0.967
Llama-3.2-3B	0.920	0.378	0.876	0.196	61	0.970
Mistral-7B	0.474	0.183	0.128	−0.146	34	0.944
Qwen2.5-0.5B	0.958	0.402	0.793	0.428	45	0.965
Qwen2.5-1.5B	0.975	0.512	0.785	0.382	64	0.978
Qwen2.5-3B	0.960	0.369	0.543	0.346	53	0.983
Qwen2.5-7B (base)	0.953	0.623	0.920	0.548	52	0.980
Qwen2.5-7B	0.952	0.738	0.687	0.505	51	0.987
Qwen3-4B	0.869	0.455	0.845	0.036	52	0.979
SmolLM2-1.7B	0.929	0.012	0.021	0.542	113	0.982
Gemma-2-2B (base)	0.869	0.641	0.766	0.462	42	0.818
Gemma-2-2B	0.897	0.504	0.851	0.245	39	0.962

Pooled over runs as fractions of each method's own gain, with only the shared component LoRA keeps 88% [84, 93] on the traits and the unconstrained update 71% [57, 83], and on the transformations 90% [80, 95] against 72% [56, 84]. With the shared component removed, LoRA keeps 29% [20, 40] against 17% [5, 29] on the traits, and 55% [44, 64] against 41% [31, 50] on the transformations. These intervals resample models. Resampling runs, the shared component alone keeps 88% [85, 91] against 71% [65, 77] on the traits and 90% [86, 93] against 72% [66, 77] on the transformations. With only the shared component, the run-level intervals of the two methods separate in both classes, and the model-level intervals separate on the traits and overlap on the transformations. The gap varies by model: with only the shared component, the two methods are close on Llama-3.1-8B and Qwen3-4B and far apart on SmolLM2-1.7B.

Without the rank constraint, the shared component holds more of the energy of the update and the behavior relies on it somewhat less, so the reliance does not come from the low rank.

D.10 In-context learning

In-context learning adapts without training, through demonstrations in the prompt. We apply the two interventions to the demonstrations and to a LoRA trained in the same runs, and call each of these two ways of adapting a *route*. A shift adapter trained in the same runs is a third route.

Experimental setup. We use 10 of the setup's models (Table 3), 3 seeds and the 9 behaviors of the setup: the traits French, bullet points, pirate and grumpy critic, the transformations copy, last word and conditional echo, and the maps cipher and lookup (270 runs). In each run the demonstrations are four question–answer pairs before the query. For the traits, the answers are the model's own answers under the behavior's system prompt, each passing the behavior's rule-based verifier, and the prompt uses the chat template. For the transformations and maps, the answers are the task's targets, and the prompt is plain text behind a preamble that asks the model to continue the pattern, because an instruction-tuned model under its template answers the last question instead. The maps get their whole 26-word table before the 4 examples, since 4 examples cover only 4 of its 26 entries. The field of the demonstrations is the change they make to the hidden state at the last token of the input, $h(\text{demonstrations} + x) - h(x)$, and its shared component is the mean of that change over 40 probe queries. To keep only the shared component, we add it to the bare query, one vector per layer at every position of the input and the answer, scaled by $\alpha \in \{0.5, 1, 2\}$ chosen per family and class on seed 0. To remove it, we keep the demonstrations and subtract it at every position. As a control, a random vector of the same norm per layer is added and subtracted in the same way. A rank-8 LoRA and a shift adapter are trained in the same run on the same 160 answers under the same prompt format, and the LoRA is run as trained, with only its shared component, and without it (Appendix D.1). Every route is scored on the same 48 held-out queries, by the behavior's rule-based verifier or, for pirate and grumpy critic, by token F1 against the model's own answers under the system prompt. Fractions pool every run, and intervals resample models. The demonstration, probe, training and evaluation queries are disjoint.

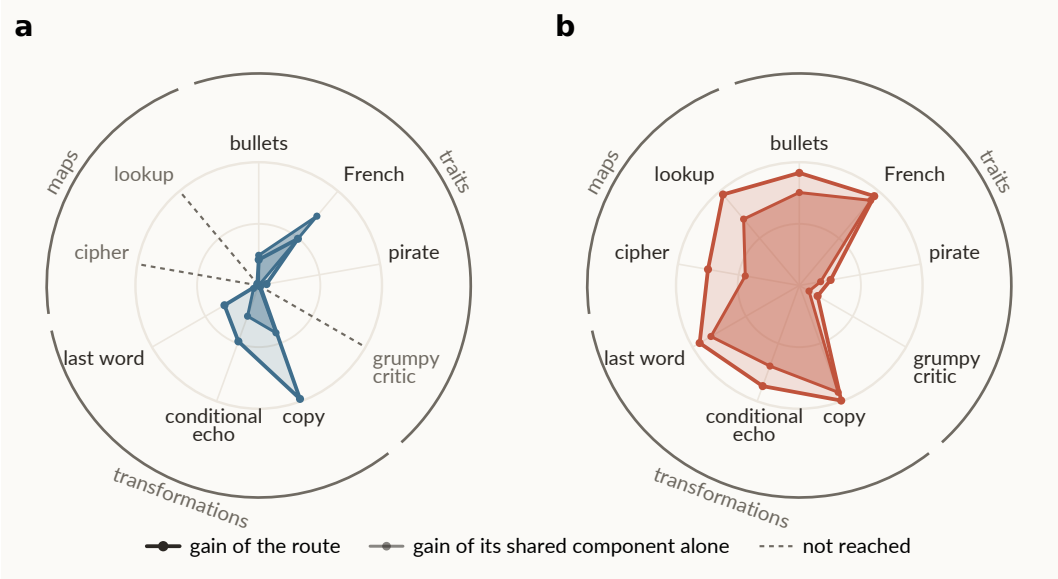

Figure 14: **The two routes behavior by behavior.** Gain over the base model of **(a)** the demonstrations and **(b)** the LoRA trained in the same runs (outer line), and of each route's shared component alone (inner line), mean over the 10 models and 3 seeds. A dashed spoke marks a behavior the route does not reach (gain below 0.05).

Results. Figure 14 compares the two routes behavior by behavior. The demonstrations gain 0.20 on the traits, 0.59 on the transformations and 0.01 on the maps, where the LoRA gains 0.57, 0.93 and 0.86, and the shift adapter 0.57, 0.89 and 0.64. With only their shared component, the demonstrations keep 96% [63, 170] of their gain on the traits and 41% [33, 48] on the transformations, where the LoRA keeps 84% [78, 90] and 87% [75, 96]. On the maps the LoRA keeps 67% [50, 81], and the demonstrations gain 0.01, too little for their fraction to mean anything (2% [−54, 60]). Removing the shared component of the demonstrations keeps none of their gain: −49% [−76, −36] on the traits and −2% [−3, −1] on the transformations. Removing a random vector of the same norm keeps 9% [−25, 53] on the traits, so there the removal is specific to the shared component. On the transformations it keeps 6% [0, 13], so a random direction of that size is as destructive as the shared component, and there the result rests on keeping only the shared component. Removing the shared component of the LoRA keeps 31% [20, 44] on the traits and 52% [34, 68] on the transformations.

Table 3 gives the values per model. The shared component of the demonstrations keeps at least 94% of their gain on the traits in seven of the 10 models, and 60% on Mistral-7B, 41% on Qwen3-4B and 30% on Llama-3.2-3B. Where the demonstrations gain little on the traits (0.04 on Llama-3.2-1B and Qwen2.5-3B), the fraction is large (224% and 841%) and says little. Mistral-7B is the one model whose LoRA keeps little of its gain on the transformations with only its shared component (35%): in three of its 9 transformation runs, the shared component alone sends over 90% of the answers into one repeated token, which no other model does in any run.

Pirate and grumpy critic. On these two traits the score and the generations disagree. The demonstrations gain 0.06 on pirate and 0.01 on grumpy critic, and with only their shared component they fall below the base model on both (gains of −0.15 and −0.08), which lowers the pooled trait numbers above. Under the demonstrations, the generations of Mistral-7B, SmolLM2-1.7B, Gemma-2-2B and Qwen3-4B carry a pirate marker as often as under the shift adapter (87% to 99% of them, against 98% to 100%), yet their F1 gain over the base model is 63% to 106% of the trained one (Mistral 0.42 against 0.41 for the shift adapter, SmolLM2 0.39 against 0.46), because the trained routes reproduce the teacher's sentences and the demonstrations write their own. The three Qwen2.5 models carry the marker in 36% to 41% of their generations. Llama-3.2-1B refuses ("I can't help with that", F1 0.04 against 0.20 for the base model), and Llama-3.2-3B and Llama-3.1-8B continue the demonstration transcript ("I'll answer your questions one by one, matey. Q: …", markers in 49% of their generations). The short pirate and grumpy critic spokes of Figure 14 therefore measure how closely the

Table 3: **The two routes per model.** Gain over the base model (mean over the model's runs) and the fraction of that gain its shared component alone keeps (ratio of sums over every run), for the demonstrations and for the LoRA trained in the same runs, on the 4 traits and the 3 transformations of the setup.

	traits				transformations			
	demonstrations		LoRA		demonstrations		LoRA	
model	gain	kept	gain	kept	gain	kept	gain	kept
Llama-3.1-8B	0.23	1.02	0.65	0.83	0.69	0.43	0.94	0.91
Llama-3.2-1B	0.04	2.24	0.61	0.68	0.52	0.35	0.93	0.80
Llama-3.2-3B	0.31	0.30	0.65	0.88	0.62	0.38	0.93	0.92
Mistral-7B	0.41	0.60	0.57	0.73	0.60	0.59	0.93	0.35
Qwen2.5-1.5B	0.14	1.15	0.45	1.01	0.55	0.31	0.91	0.97
Qwen2.5-3B	0.04	8.41	0.54	0.97	0.58	0.28	0.93	0.96
Qwen2.5-7B	0.11	1.25	0.56	0.96	0.61	0.22	0.92	0.99
Qwen3-4B	0.33	0.41	0.56	0.77	0.63	0.37	0.96	0.94
SmolLM2-1.7B	0.27	0.94	0.55	0.79	0.53	0.62	0.93	0.95
Gemma-2-2B	0.09	1.89	0.59	0.82	0.62	0.51	0.94	0.95

demonstrations reproduce the teacher's answers, and the generations carry the pirate marker in 36% to 99% of cases in nine of the 10 models.

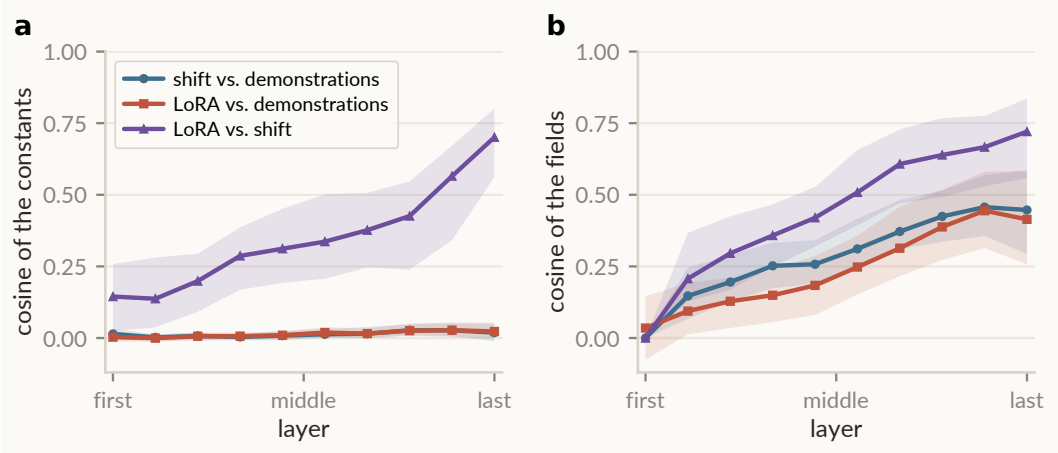

Figure 15: **Shared components and their fields through depth.** Median cosine per layer decile over the 270 runs on the 9 behaviors of the setup, quartiles shaded, for 3 pairs of routes: the shift adapter and the demonstrations, the LoRA and the demonstrations, the LoRA and the shift adapter. **(a)** Their shared components: the shift adapter's vector per layer, the vector per layer that keeping only the LoRA's shared component adds, and the demonstrations' shared component. **(b)** Their fields.

Shared components and fields through depth. Figure 15 compares the shared components of the three routes, and their fields, layer by layer. The shift adapter's vector and the shared component of the demonstrations are orthogonal at every layer: the median cosine per layer decile lies between -0.01 and 0.03 on the traits and transformations and is at most 0.07 on the maps, and at most 0.27 in any run and layer. The shared component of the LoRA, the vector per layer that keeping only it adds, is also orthogonal to that of the demonstrations (median cosine per layer decile at most 0.04 on the traits and transformations and 0.08 on the maps, at most 0.26 in any run and layer). The two trained routes agree more with depth, from a median cosine of 0.16, 0.18 and 0.04 in the first layer decile on the traits, transformations and maps to 0.61, 0.71 and 0.84 in the last. The fields are not orthogonal. By the ninth layer decile, the fields of the shift adapter and the demonstrations reach a cosine of 0.44 on the traits and 0.51 on the transformations, the fields of the shared component of the LoRA and of the demonstrations 0.41 and 0.57, and the fields of the two trained routes 0.76 and 0.61, rising to 0.86 and 0.62 in the last decile. The trained routes add vectors orthogonal to the shared component of the demonstrations, and their fields meet the field of the demonstrations in the late layers.

On the tasks. We run the same two interventions on 4 of the tasks of Section 3, AG News, SST-2, OpenBookQA and HellaSwag, over the 17 models at 3 seeds (204 runs). Each run puts four label-balanced demonstrations before the query, takes their shared component as the mean change at the last token of the input over 300 probe queries, and chooses the scale $\alpha \in \{0.05, 0.1, 0.25, 0.5, 1, 2\}$ on a held-out slice of 300 rows. Every route is then scored on the task's full evaluation split. A rank-8 LoRA trained in the same run on the task's training rows, never on the demonstrations, is scored the same way.

Four demonstrations change these tasks little: the median gain over the 204 runs is 0.033 (quartiles 0.011 to 0.078), and one run (Llama-3.2-1B on AG News) ends 0.267 below the bare prompt (Figure 16b). The demonstrations gain 0.05 or more on average only on AG News and SST-2. With only their shared component, the demonstrations keep 48% [36, 62] of their gain over every run, where the LoRA trained in the same runs keeps 82% [75, 88] of its own, and a random vector of the same norm keeps −15% [−36, −1]. On AG News and SST-2 the demonstrations keep 51% and 54%, where the LoRA keeps 94% and 96% on the same runs (Figure 16a). On OpenBookQA and HellaSwag their mean gain is below 0.05, too little for a fraction, while the LoRA keeps 74% and 66%. The fractions of the LoRA on every task are in Appendix D.2. With 16 demonstrations (198 runs), the shared component alone keeps 48% [34, 62] and removing it keeps 44% [30, 62], against 48% and 34% with four.

Removing the shared component from the prompt with the demonstrations keeps 34% [13, 51], against 78% [69, 86] for removing a random vector of the same norm, so the direction matters. The scale is chosen for keeping only the shared component and reused for the removal, and in 34 of the 204 runs the removal sends the model to one class, which widens the interval. At full strength it does so in every run, as on the behaviors. Through depth, the shared components of the two routes are orthogonal, with a median cosine per layer decile between 0.00 and 0.02 over all 204 runs, while their fields converge, from 0.00 in the first decile to 0.75 in the last (quartiles [0.62, 0.85], Figure 16c). Table 4 gives the values per model. On behaviors and tasks, the shared component of the demonstrations is orthogonal to LoRA's and keeps less of the gain on the transformations and the tasks, so in-context learning reaches similar behaviors by a different route from LoRA.



Figure 16: **The two routes on standard tasks.** Color is the route, shade the number of demonstrations: solid for four, pale for sixteen. The LoRA is trained on the run's training rows, so it has one bar. **(a)** The fraction of each route's own gain that its shared component alone keeps, per task, as a ratio of sums over every run, with whiskers resampling models, and the tick is a random vector of the same norm in place of the demonstrations' shared component. Where the demonstrations gain less than 0.05 on average, there is no gain to divide ("no gain"). **(b)** Every run's gain, per task, with the mean marked and a line at 0.05. **(c)** Through depth, the cosine between the two routes' shared components and between the changes they make to the hidden states, median per layer decile over all 204 runs with quartile bands.

Table 4: **The two routes per model on the tasks.** Gain over the base model (mean over the model's 12 runs) and the fraction of that gain its shared component alone keeps (ratio of sums over every run), for the demonstrations and for the LoRA trained in the same runs, over AG News, SST-2, OpenBookQA and HellaSwag. Where a route gains little, its fraction is large or negative and says little.

model	runs	demonstrations		LoRA	
		gain	kept	gain	kept
Gemma-2-2B	12	+0.14	+0.38	+0.23	+0.92
Gemma-2-2B (base)	12	+0.21	+0.36	+0.40	+0.83
Llama-3.1-8B	12	+0.07	+0.53	+0.14	+0.90
Llama-3.1-8B (base)	12	+0.06	+0.15	+0.18	+0.90
Llama-3.2-1B	12	−0.01	−0.69	+0.19	+0.74
Llama-3.2-3B	12	+0.06	+0.00	+0.15	+0.78
Mistral-7B	12	+0.07	+0.59	+0.15	+0.68
Qwen2.5-0.5B	12	+0.08	+1.13	+0.24	+0.79
Qwen2.5-1.5B	12	+0.05	+0.91	+0.15	+0.85
Qwen2.5-3B	12	+0.04	+0.31	+0.11	+0.86
Qwen2.5-7B	12	+0.03	+0.24	+0.08	+0.95
Qwen2.5-7B (base)	12	+0.03	+0.37	+0.10	+0.94
Qwen3-4B	12	+0.01	+0.22	+0.05	+0.89
Qwen3.5-4B	12	+0.15	+0.37	+0.21	+0.97
Qwen3.5-9B	12	+0.05	+0.52	+0.09	+0.96
SmolLM2-1.7B	12	+0.17	+0.68	+0.35	+0.86
Tulu-3-8B-SFT	12	+0.03	+0.41	+0.10	−0.02
all	204	+0.07	+0.48	+0.17	+0.82

E THE SHIFT ADAPTER

E.1 THE COMPARISON ON THE TASKS

We compare the shift adapter (Figure 17) with LoRAs of rank 1, 4, 8, 16 and 64. The shift adapter and the LoRAs are trained on the same teacher pairs, each at the learning rate selected for it (Appendix C.5), and every task is scored by accuracy on its whole labeled evaluation split. These splits are unbalanced, so the majority-class rate is reported beside every result.

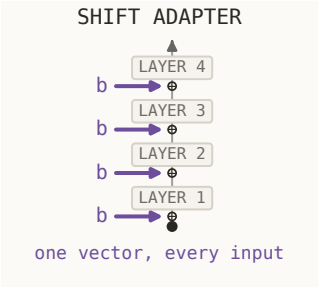

Figure 17: **The shift adapter.** One trained vector b_ℓ per layer, added to the hidden state at every position, the same for every input.

Table 5 gives every task and every method, each with a bootstrap interval over the 51 combinations of model and seed behind it, and the paired difference between the shift adapter and the rank-8 LoRA with its own interval. Pooled over the 867 runs of 17 models, the shift adapter and the rank-8 LoRA both reach 0.738, against 0.613 for the base model, a paired difference of $+0.0006$ [-0.0007, $+0.0019$]. Against the other ranks the difference ranges from $+0.0013$ [$+0.0001$, $+0.0024$] for rank 1 to -0.0018 [-0.0030, -0.0006] for rank 64. Per model it ranges from -0.006 on SmolLM2-1.7B and Gemma-2-2B to $+0.011$ on the base Gemma-2-2B, so no single model carries the result. Per task, the LoRA leads most on HellaSwag (-0.009), ARC-Challenge (-0.007) and MMLU-Pro (-0.007), and the shift adapter leads on AQuA-RAT ($+0.013$), SuperGPQA ($+0.010$), SIQA ($+0.006$) and AG News ($+0.005$). Figure 18 draws every task and every model. The selected learning rates of the shift adapter span a factor of 1,000 across models and tasks, from 10^{-5} to 10^{-2}, against 50 for the rank-8 LoRA, which is why each method trains at its own selected rate.

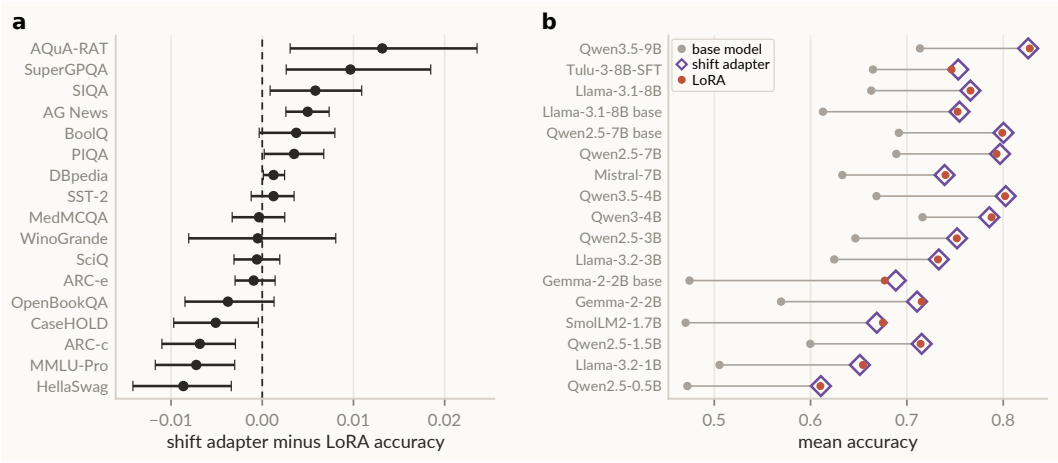

Figure 18: **Figure 4 with every task and every model. (a)** The paired shift-minus-rank-8 difference per task, with bootstrap 95% intervals over the runs behind each. **(b)** Per model, ordered by scale: mean accuracy of the base model, of the shift adapter and of the rank-8 LoRA over the 17 tasks.

E.2 FULL FINE-TUNING ON THE SAME TASKS

Full fine-tuning updates every weight. We train it with AdamW in bfloat16 for 500 steps, the budget of the LoRA, at a learning rate selected per model and task from the 6 rates of Appendix C.5, on the 17 tasks and 17 models, scored on the same evaluation splits. The shift adapter is retrained in the same run at its selected rate, so every difference is paired within a run. The retrained shift adapter reproduces the shift adapter of Table 5 to a mean absolute difference of 0.004.

Pooled over the 867 runs, full fine-tuning reaches 0.710, the shift adapter and the rank-8 LoRA 0.738, against 0.613 for the base model, a paired difference between the shift adapter and full fine-tuning of $+0.029$ [$+0.026$, $+0.031$] (Table 6). The shift adapter is at or above full fine-tuning

Table 5: **Every task and every method.** Generated-answer accuracy on the full labeled evaluation split of the 17 tasks, 17 models and 3 seeds per entry (867 runs). Brackets are bootstrap 95% intervals, and *floor* is the majority-class rate. The last column is the paired shift-minus-rank-8 difference.

benchmark	floor	shift	r1	r4	r8	r16	r64	shift − r8
BoolQ	0.622	0.860 [0.850, 0.870]	0.854 [0.841, 0.866]	0.854 [0.841, 0.866]	0.857 [0.844, 0.868]	0.857 [0.845, 0.868]	0.859 [0.849, 0.869]	0.004 [−0.000, 0.008]
PIQA	0.505	0.822 [0.804, 0.838]	0.816 [0.798, 0.834]	0.817 [0.799, 0.834]	0.818 [0.801, 0.834]	0.819 [0.801, 0.836]	0.821 [0.803, 0.837]	0.003 [0.000, 0.007]
SIQA	0.336	0.742 [0.728, 0.755]	0.739 [0.724, 0.754]	0.737 [0.722, 0.752]	0.736 [0.722, 0.749]	0.741 [0.726, 0.754]	0.744 [0.729, 0.757]	0.006 [0.001, 0.011]
WinoGrande	0.504	0.731 [0.709, 0.753]	0.730 [0.708, 0.751]	0.733 [0.710, 0.755]	0.731 [0.708, 0.753]	0.734 [0.712, 0.754]	0.738 [0.716, 0.759]	−0.000 [−0.008, 0.008]
ARC-e	0.268	0.905 [0.887, 0.922]	0.906 [0.888, 0.923]	0.906 [0.889, 0.922]	0.906 [0.887, 0.924]	0.909 [0.892, 0.925]	0.909 [0.892, 0.925]	−0.001 [−0.003, 0.001]
ARC-c	0.264	0.781 [0.748, 0.811]	0.784 [0.751, 0.814]	0.786 [0.755, 0.814]	0.788 [0.757, 0.817]	0.786 [0.755, 0.815]	0.790 [0.759, 0.819]	−0.007 [−0.011, −0.003]
HellaSwag	0.257	0.825 [0.795, 0.852]	0.832 [0.804, 0.857]	0.837 [0.809, 0.861]	0.834 [0.805, 0.858]	0.834 [0.806, 0.859]	0.830 [0.800, 0.856]	−0.009 [−0.014, −0.003]
OpenBookQA	0.276	0.840 [0.818, 0.861]	0.847 [0.826, 0.867]	0.847 [0.825, 0.867]	0.843 [0.822, 0.865]	0.833 [0.801, 0.860]	0.848 [0.825, 0.870]	−0.004 [−0.009, 0.001]
SST-2	0.509	0.946 [0.942, 0.949]	0.943 [0.938, 0.948]	0.942 [0.938, 0.947]	0.945 [0.941, 0.948]	0.946 [0.942, 0.949]	0.949 [0.946, 0.952]	0.001 [−0.001, 0.003]
DBpedia	0.071	0.986 [0.985, 0.986]	0.985 [0.984, 0.985]	0.984 [0.983, 0.985]	0.984 [0.983, 0.985]	0.984 [0.983, 0.985]	0.984 [0.983, 0.985]	0.001 [0.000, 0.002]
AG News	0.250	0.900 [0.899, 0.902]	0.896 [0.894, 0.898]	0.893 [0.888, 0.897]	0.895 [0.893, 0.898]	0.897 [0.894, 0.900]	0.898 [0.895, 0.901]	0.005 [0.003, 0.007]
CaseHOLD	0.202	0.721 [0.708, 0.732]	0.723 [0.712, 0.733]	0.724 [0.713, 0.734]	0.726 [0.715, 0.736]	0.722 [0.711, 0.732]	0.719 [0.706, 0.731]	−0.005 [−0.010, −0.000]
MedMCQA	0.322	0.531 [0.509, 0.552]	0.532 [0.509, 0.554]	0.531 [0.508, 0.554]	0.531 [0.507, 0.554]	0.532 [0.509, 0.554]	0.530 [0.508, 0.552]	−0.000 [−0.003, 0.003]
AQuA-RAT	0.248	0.376 [0.350, 0.405]	0.366 [0.336, 0.397]	0.367 [0.338, 0.398]	0.363 [0.334, 0.395]	0.370 [0.341, 0.402]	0.373 [0.344, 0.404]	0.013 [0.003, 0.024]
SciQ	0.269	0.912 [0.901, 0.923]	0.912 [0.901, 0.922]	0.913 [0.902, 0.924]	0.913 [0.901, 0.924]	0.913 [0.901, 0.924]	0.911 [0.899, 0.923]	−0.001 [−0.003, 0.002]
MMLU-Pro	0.112	0.408 [0.382, 0.435]	0.406 [0.378, 0.434]	0.414 [0.387, 0.442]	0.415 [0.388, 0.444]	0.410 [0.382, 0.440]	0.414 [0.389, 0.440]	−0.007 [−0.011, −0.003]
SuperGPQA	0.105	0.268 [0.250, 0.287]	0.263 [0.243, 0.284]	0.265 [0.244, 0.286]	0.258 [0.238, 0.279]	0.264 [0.244, 0.285]	0.269 [0.250, 0.289]	0.010 [0.003, 0.019]

on every task except ARC-Challenge (−0.002), and leads most on WinoGrande (+0.067), CaseHOLD (+0.061) and HellaSwag (+0.060). Per model its lead ranges from +0.016 on Qwen3.5-9B to +0.069 on the base Gemma-2-2B. Over the same runs, the field of full fine-tuning has a cosine of 0.90 to its shared component and a field share of 0.82, against 0.91 and 0.83 for the rank-8 LoRA, paired differences of −0.008 and −0.015 (Figure 11c). Full fine-tuning moves every input in nearly the same direction, as LoRA does.

E.3 The comparison before instruction tuning

We repeat the comparison on the 4 models that never received instruction tuning (Appendix D.4), on the 17 tasks at 3 seeds (204 runs), each method at its selected rate. The shift adapter reaches 0.749 and the rank-8 LoRA 0.744, against 0.611 for the base model, a paired difference of +0.005 [+0.003, +0.009], and the shift adapter is at or above every rank from 1 to 64. Full fine-tuning on the same models and tasks reaches 0.711, below the shift adapter by +0.038 [+0.032, +0.045]. The parity with LoRA holds before instruction tuning, as the reliance on the shared component does (Appendix D.4).

E.4 The shift adapter and the LoRA push in the same direction

The shift adapter and the LoRA are trained independently on the same teacher pairs, so we ask whether they push the inputs in the same direction. On the 17 tasks and 17 models of Appendix E.1, with both trained in the same run at their selected rates, the shared component of the LoRA aligns with the shift adapter at a cosine of 0.65 [0.62, 0.68], against 0.06 [0.04, 0.08] for a rank-8 LoRA with random weights and no training, measured in the same 867 runs. On the 9 behaviors and 17 models of Appendix D.5 (459 runs), the two align at 0.53 [0.50, 0.56], against 0.08 [0.05, 0.11] for

Table 6: **Full fine-tuning against the shift adapter on every task.** Accuracy on the evaluation split, across 17 models and 3 seeds per entry (867 runs), on the 17 tasks. The last column is the paired shift-minus-full-fine-tuning difference with its bootstrap 95% interval.

benchmark	majority-class rate	base	shift adapter	rank-8 LoRA	full FT	shift − full FT
BoolQ	0.622	0.764	0.859	0.857	0.834	+0.025 [+0.021, +0.029]
PIQA	0.505	0.737	0.823	0.818	0.801	+0.021 [+0.017, +0.027]
SIQA	0.336	0.682	0.743	0.736	0.716	+0.027 [+0.020, +0.035]
WinoGrande	0.504	0.581	0.730	0.731	0.663	+0.067 [+0.053, +0.084]
ARC-e	0.268	0.883	0.905	0.906	0.899	+0.006 [+0.004, +0.010]
ARC-c	0.264	0.759	0.781	0.788	0.783	−0.002 [−0.005, +0.002]
HellaSwag	0.257	0.615	0.824	0.834	0.764	+0.060 [+0.048, +0.074]
OpenBookQA	0.276	0.741	0.840	0.844	0.808	+0.031 [+0.023, +0.040]
SST-2	0.509	0.851	0.945	0.945	0.936	+0.009 [+0.006, +0.013]
DBpedia	0.071	0.512	0.985	0.984	0.984	+0.001 [−0.000, +0.003]
AG News	0.250	0.626	0.900	0.895	0.887	+0.013 [+0.010, +0.016]
CaseHOLD	0.202	0.497	0.722	0.726	0.662	+0.061 [+0.054, +0.068]
MedMCQA	0.322	0.503	0.531	0.531	0.516	+0.015 [+0.012, +0.019]
AQuA-RAT	0.248	0.282	0.375	0.363	0.328	+0.046 [+0.036, +0.056]
SciQ	0.269	0.894	0.912	0.913	0.900	+0.012 [+0.008, +0.016]
MMLU-Pro	0.112	0.292	0.408	0.415	0.360	+0.048 [+0.038, +0.058]
SuperGPQA	0.105	0.194	0.268	0.259	0.225	+0.043 [+0.037, +0.049]
all		0.613	0.738	0.738	0.710	+0.029 [+0.026, +0.031]

the random LoRA (Figure 5). There, each input's field points along its adapter's shared component at a mean cosine of 0.85 [0.82, 0.87] for the shift adapter and 0.84 [0.80, 0.87] for the LoRA, against 0.67 [0.61, 0.72] for the random LoRA. Weighting each layer by how far the two adapters move it raises the agreement between the shift adapter and the LoRA to 0.69, while the random LoRA stays at 0.08 (Appendix C.7). A LoRA that can vary what it adds with every input pushes along the direction the shift adapter learns.

E.5 What a shift adapter can store

With only its shared component, LoRA keeps the least of its gain on the maps, whose rule the model can only learn during adaptation (Appendix D.6). We therefore ask what a shift adapter trained directly can store: whether it learns the maps, how much content fits in it, and how that content relates to the shared component.

A shift adapter learns the maps. We train a shift adapter directly on the cipher and on a lookup table whose entries are assigned at random, on 5 families at 3 seeds, and score its held-out generations by exact match. Each map has seven possible answers, so we read the score against the majority-class rate, the score of always giving the most common answer. The shift adapter exceeds that rate on both maps in every family, by 0.08 to 0.38, with its answers spread over three to five of the seven, at the level of the LoRA. As a control, a third map, the number of words of the input modulo three, stays at or below the majority-class rate in every family. A shift adapter can therefore store content that pretraining never showed the model. The lower fraction on the maps in Appendix D.6 comes from the trained LoRA, which puts part of the map in its input-specific component, and a shift adapter trained directly on the map does without that component.

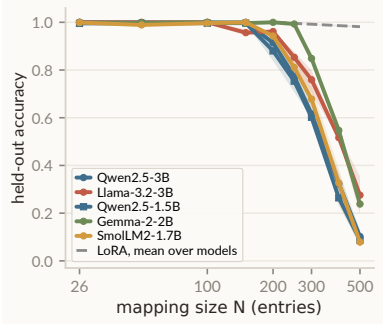

Figure 19: **An actualizing state carries a finite payload.** Held-out lookup accuracy stays high through 200 entries and falls by 500 (5 families, 3 seeds).

How much fits. We train a shift adapter on key–value tables of 100 to 500 entries and count the entries it answers on held-out queries (5 families, 3 seeds). The count rises with the table, 100, 149, 185, then 201 at 250 entries and 199 at 300, and then falls, to 112 at 400 entries and 43 at

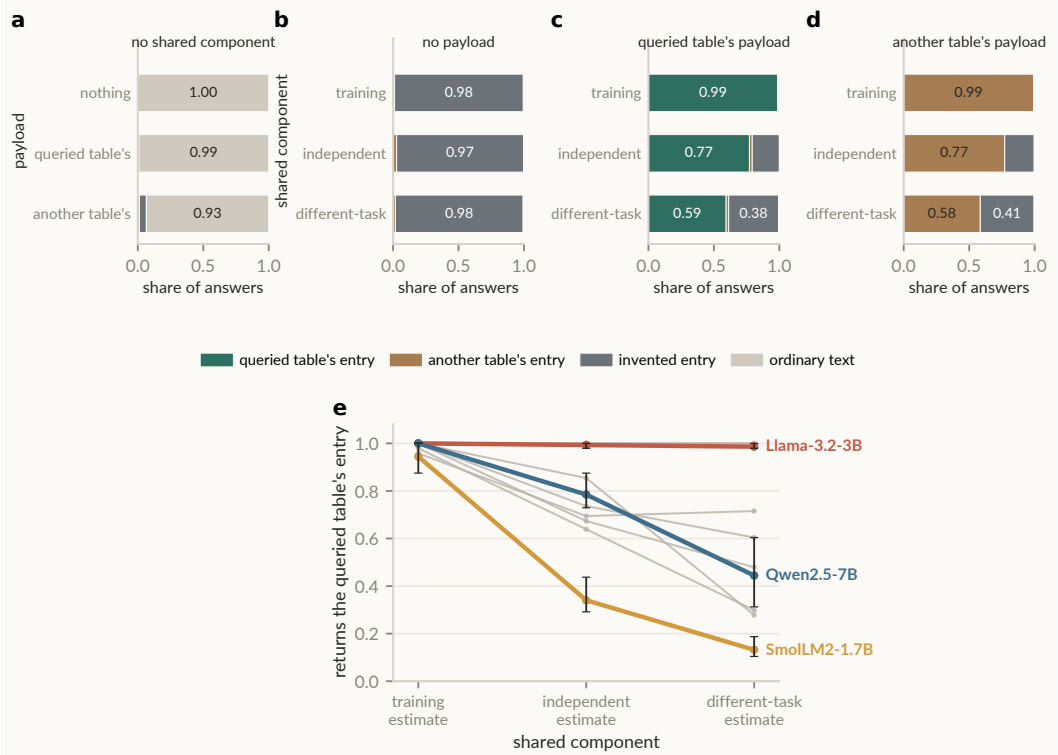

Figure 20: **The shared component makes the model look an entry up, and the payload decides which table it reads.** Every bar splits the generated answers into four kinds, pooled over 10 models, 3 seeds and four payload tables. **(a)** A payload without a shared component leaves the model's answers ordinary text. **(b)** A shared component without a payload makes the model answer with an invented entry, whichever source the shared component comes from: the training estimate (the mean of the 24 shift adapters under which the payload was trained), an independent estimate from 24 other tables, or an estimate from a different task. **(c)** With the queried table's payload, the shared component returns that table's entries, fewer of them the further its source is from the training estimate. **(d)** With another table's payload, it returns that table's entries at the same rates as in (c). **(e)** The rate of returning the queried table's entry per model, over the three sources, three models in color with bootstrap 95% intervals and the other seven in grey.

500 (Figure 19). At 200 entries the 5 families answer 0.88 to 1.00 of the table. A shift adapter therefore holds about 200 entries, and a larger table lowers the count. The field does not change at the limit: from 100 to 500 entries, its magnitude (728, 824, 729), the share of its energy in the shared component (0.43, 0.46, 0.45) and the effective rank of its input-specific component (59, 54, 55) stay the same. Rank does not move the limit: a rank-1 LoRA holds 500 entries as well as a rank-64 one (0.950 against 0.949). With only its shared component, a trained LoRA keeps 16% [9, 22] of its gain at 500 entries, against nearly all of it at 100, so past the shift adapter's limit the content sits in the LoRA's input-specific component. Documents exceed the limit as well. On SQuAD a shift adapter recovers 54% [52, 57] of the gain of reading the document in context, against 62% [60, 64] for the LoRA, and on CoQA 13% [10, 16] against 17% [14, 21] (30 held-out documents each, 5 families, 3 seeds).

The shared component invokes the lookup, and a payload supplies the entries. On the lookup, one trained shift adapter holds two things: the operation of looking an entry up, and the entries of one table. We separate them by construction. The shared component is estimated as the mean of 24 shift adapters, each trained on a different table over one set of 50 keys. With that estimate frozen in the model, a new per-layer vector, the *payload*, is trained on the entries of a table outside the set. We then combine shared components and payloads from different sources and sort each generated answer into four kinds: the queried table's entry, another table's entry, an invented entry, and ordinary text (10 models, 3 seeds, four payload tables, 48 held-out prompts each). The payload alone leaves the model's behavior unchanged. The shared component alone makes the model look an entry up with nothing to read, so it invents one (0.98 [0.97, 0.99] of answers). The shared component with

another table's payload returns that table's entries (0.99 [0.99, 1.00]). A shared component estimated from 24 other tables still reads the payload at 0.77 [0.70, 0.84], from 1.00 on every Llama model to 0.13 on SmolLM2-1.7B (Figure 20). Under a shared component trained for the reverse lookup, from value to key, the forward payload returns 0.000 [0.000, 0.001] of its entries, while a random direction at the same distance returns 0.76 [0.72, 0.81], so the payload stores the table in the direction the forward lookup reads it. The shared component carries the operation, and the payload carries the content it reads.

F The structure of actualization

F.1 The principle at each checkpoint of OLMo-2-7B

To see when the two parts of the principle appear during training, we repeat the comparison of Appendix E.1 and the interventions of Appendix D.1 along the training of OLMo-2-7B, which AI2 publishes with more than 900 intermediate checkpoints. We use 15: five of stage-one pretraining on Dolma, at 1B, 9B, 105B, 449B and 3,896B tokens, six of the 50B-token stage two that follows, at 5B, 13B, 21B, 30B, 38B and 50B tokens into it, from the first of the 3 stage-two runs whose average is the released base, and the released base, supervised, preference-tuned and instruct models. Each checkpoint follows the protocol of Appendix C.5 on the 17 tasks: for each task and method a rate is selected from six on a validation slice of the training split at seed 0, and 3 seeds are trained on the whole training split at that rate and scored on the full evaluation split, 765 runs in all. The accuracy of the base model is the zero-shot accuracy of the checkpoint on the same split, read beside the split's majority class. A run where neither method scores more than 0.02 above the majority class counts as a failure of adaptation and stays in every average. The rank-8 LoRA of each run is then scored under the two interventions, with every run pooled and intervals clustered by task, since one checkpoint at 3 seeds has 17 independent units, the tasks, and not 51. Table 7 lists every checkpoint, Figure 6 draws it, and Figure 21 draws the task types with the LoRA beside the shift adapter.

Table 7: **The principle at each checkpoint of OLMo-2-7B.** Mean accuracy over the 17 tasks and 3 seeds: the majority class, the base model with no adaptation, the shift adapter and the rank-8 LoRA at their selected rates, and the paired shift-minus-rank-8 difference. *Fails* counts the runs of 51 that neither method lifts more than 2% above the majority class. The last two columns are the fraction of the rank-8 LoRA's gain kept with only its shared component and with that component removed, pooled over every run. At 1B and 9B tokens the LoRA gains little on most tasks, so these two fractions are wide. Brackets are 95% intervals clustered by task.

checkpoint	majority	base	shift	rank-8	shift − rank-8	fails	kept, shared only	kept, shared removed
stage 1								
1B	0.30	0.28	0.30	0.32	−0.022 [−0.062, +0.003]	44/51	0.91 [0.46, 2.20]	0.04 [−0.23, 0.20]
9B	0.30	0.34	0.39	0.40	−0.004 [−0.015, +0.005]	42/51	0.70 [−1.82, 2.50]	0.31 [−0.31, 0.70]
105B	0.30	0.33	0.45	0.45	+0.002 [−0.019, +0.020]	28/51	0.70 [0.20, 0.94]	0.25 [0.06, 0.43]
449B	0.30	0.47	0.65	0.65	+0.000 [−0.015, +0.018]	3/51	0.81 [0.70, 0.90]	0.41 [0.28, 0.53]
3,896B	0.30	0.56	0.72	0.72	−0.007 [−0.015, +0.001]	0/51	0.80 [0.72, 0.87]	0.47 [0.20, 0.71]
stage 2								
5B	0.30	0.59	0.72	0.73	−0.005 [−0.012, +0.001]	1/51	0.85 [0.77, 0.91]	0.70 [0.56, 0.82]
13B	0.30	0.59	0.73	0.73	−0.000 [−0.006, +0.006]	0/51	0.93 [0.88, 0.96]	0.51 [0.39, 0.67]
21B	0.30	0.61	0.73	0.73	−0.001 [−0.009, +0.007]	0/51	0.90 [0.86, 0.94]	0.60 [0.35, 0.77]
30B	0.30	0.61	0.73	0.74	−0.004 [−0.010, +0.002]	0/51	0.87 [0.78, 0.93]	0.63 [0.52, 0.76]
38B	0.30	0.61	0.73	0.74	−0.005 [−0.012, +0.003]	0/51	0.88 [0.79, 0.94]	0.57 [0.40, 0.72]
50B	0.30	0.61	0.73	0.73	−0.001 [−0.009, +0.007]	0/51	0.91 [0.86, 0.96]	0.51 [0.29, 0.70]
released								
base	0.30	0.62	0.74	0.74	−0.003 [−0.012, +0.008]	0/51	0.85 [0.72, 0.92]	0.61 [0.45, 0.74]
SFT	0.30	0.63	0.73	0.74	−0.008 [−0.014, −0.002]	0/51	0.81 [0.73, 0.91]	0.61 [0.43, 0.75]
DPO	0.30	0.63	0.73	0.73	−0.001 [−0.007, +0.005]	0/51	0.88 [0.81, 0.94]	0.59 [0.41, 0.76]
instruct	0.30	0.63	0.73	0.73	−0.002 [−0.008, +0.004]	0/51	0.87 [0.81, 0.94]	0.64 [0.48, 0.78]

By task type (Figure 21), the base model scores 0.57 on the three text classifications at 9B tokens and 0.82 at the released base, where the shift adapter reaches 0.86 and 0.96 and the LoRA 0.88 and 0.96. On the four commonsense question-answering tasks the base model rises from 0.42 at 1B tokens to 0.64 at the end of stage one and 0.70 at the released base, where the shift adapter and the LoRA both reach 0.81. On the three science question-answering tasks it stays at the majority class through 105B tokens and reaches 0.53 at 449B and 0.79 at the released base, where the shift adapter reaches 0.84 and the LoRA 0.86. On the four domain tasks it stays at the majority class through

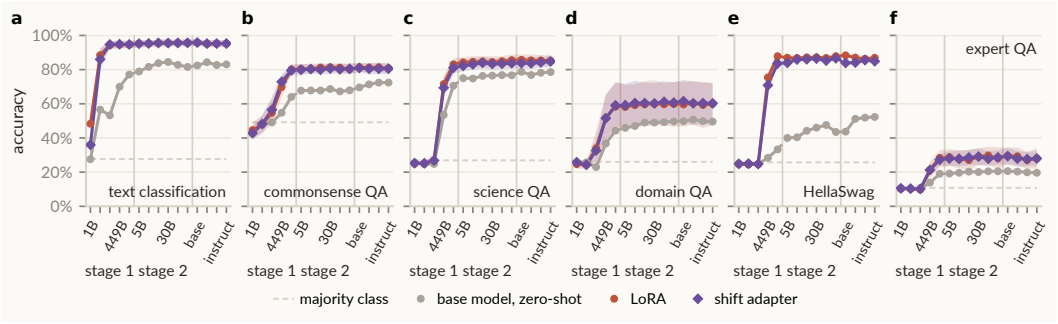

Figure 21: **Accuracy along the training of OLMo-2-7B, by task type.** OLMo-2-7B at 15 checkpoints, 3 seeds: **(a)** the three text classifications, **(b)** the four commonsense question-answering tasks, **(c)** the three science question-answering tasks, **(d)** the four domain tasks, **(e)** HellaSwag and **(f)** the two expert question-answering tasks. Each panel draws the shift adapter (purple diamonds), the LoRA (red), the base model with no adaptation (grey), and the majority class (dashed line), mean over the type's tasks and seeds with 95% intervals over those runs.

105B tokens and reaches 0.37 at 449B and 0.50 at the released base, where the shift adapter reaches 0.52 and 0.62 and the LoRA 0.52 and 0.60. On HellaSwag it stays at the majority class through 105B tokens and reaches 0.28 at 449B, 0.44 at the released base and 0.52 at the instruct model, where the shift adapter reaches 0.71, 0.84 and 0.85 and the LoRA 0.75, 0.88 and 0.87. On the two expert question-answering tasks it stays at the majority class through 105B tokens and reaches 0.14 at 449B and 0.21 at the released base, where the shift adapter reaches 0.21 and 0.29 and the LoRA 0.21 and 0.30. From 449B tokens on, the two adapters are within 0.04 of each other in every task type, and the LoRA leads clearly only at 1B tokens on the classifications, 0.48 against 0.36.

At the early checkpoints only some tasks are lifted above the majority class. At 1B tokens the LoRA lifts DBpedia to 0.51 and AG News to 0.41, where the shift adapter reaches 0.23 and 0.35 against majority classes of 0.07 and 0.25, and the LoRA alone lifts SST-2 slightly (0.53 against a majority class of 0.51). At 9B tokens the base model already scores 0.55 to 0.58 on these 3 tasks, and both methods lift them, DBpedia to 0.97, AG News to 0.88 and 0.87, and SST-2 to 0.74 and 0.81, while the other 14 tasks stay at the majority class. At 105B tokens BoolQ, SIQA, SciQ and CaseHOLD join them. At 449B tokens both methods lift every task but AQuA-RAT, and from 3,896B tokens every task. Across all 255 pairs of task and checkpoint, only three are lifted by one method alone: SST-2 at 1B and ARC-Easy at 105B tokens by the LoRA (0.31 against 0.26, majority class 0.27), and CaseHOLD at 105B tokens by the shift adapter (0.28 against 0.22, majority class 0.20).

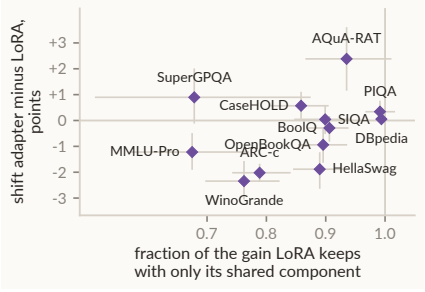

Figure 22: **The gap to the LoRA against the LoRA's reliance on its shared component.** Per task, pooled over the 10 stage-two and released checkpoints of OLMo-2-7B: the difference between the shift adapter and the rank-8 LoRA against the fraction of the LoRA's gain kept with only the shared component. Every run is pooled. The 5 tasks whose mean gain is below 0.05 over these checkpoints (ARC-Easy, SciQ, SST-2, AG News and MedMCQA) have no gain to divide and are not drawn. Bars are 95% intervals.

With only the shared component, the LoRA keeps between 70% and 93% of its gain at every checkpoint, and with that component removed 4% at 1B tokens, 25% to 47% over the rest of stage one and 51% to 70% from stage two on (Table 7). At 1B and 9B tokens the LoRA gains little on most tasks and both intervals are wide. At every checkpoint the shared component alone keeps more of the gain than the rest of the update.

Figure 22 pools the 10 stage-two and released checkpoints per task. The shift adapter trails the LoRA most on WinoGrande, ARC-Challenge and HellaSwag, by about 0.02 each, and by 0.012 or less on the other tasks. On those 3 tasks the shared component of the LoRA keeps 76%, 79% and 89% of its gain, and 67% to 99% on the other tasks with a gain. Over the 12 tasks with a fraction, the gap between the two methods and the fraction of the LoRA's gain its shared component keeps correlate

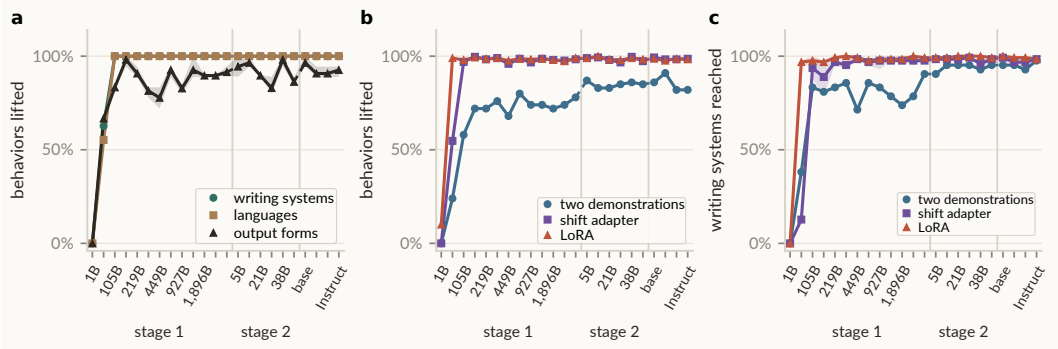

Figure 23: **Three ways of producing a behavior along the training of OLMo-2-7B, at 23 checkpoints. (a)** The share of each class of behavior a trained shift adapter lifts, where lifted means it raises the model's own rate on the behavior by more than 0.05 over 40 held-out answers, of the behaviors the training target itself produces: 42 writing systems, 38 languages and 20 output forms. **(b)** The three routes over the 100 behaviors scored: two demonstrations in the prompt, a trained shift adapter and a rank-8 LoRA. **(c)** The 42 writing systems under one criterion for every route, produced in a majority of the held-out answers. The learning rate is selected per checkpoint, class and method at seed 0, and the band spans the 3 seeds read at that rate. The horizontal axis changes at each ruled boundary, from stage-one tokens to stage-two tokens to released models, so a segment that crosses one is dashed. Appendices F.2 and F.4 state the criteria and the rate selection.

at Spearman $+0.41$ ($p = 0.19$). On every task, the two methods differ by at most 0.024, the shift adapter's lead on AQuA-RAT.

F.2 Behaviors along the training of OLMo-2-7B

Beside the accuracy on the tasks, we count at each checkpoint how many of a fixed set of behaviors the model can be made to produce, by three routes: two demonstrations in the prompt, a trained shift adapter and a rank-8 LoRA, each adapter trained on 160 questions and read on 40 held-out ones.

Behaviors. Three classes of behavior can be scored without a teacher or a judgment. There are 49 non-Latin writing systems, scored by Unicode script property, 45 Latin-script languages, scored by a language identifier, and 42 output forms, scored by the predicate each instruction states, such as answering only in questions or only in bullets. The output forms follow the verifiable-instruction construction of IFEval (Zhou et al., 2023). Several are its instruction types directly: no commas, a word or sentence count, bullets or a JSON object, capital letters, a forbidden word or letter, and a response language. The others, such as a sonnet or a pros-and-cons table, are built the same way. Every check refuses a degenerate answer, one that is too short, too repetitive or a phrase repeated in a loop, since such an answer can pass a predicate without following the instruction: a repeated character of a script passes a check that reads the share of letters in that script.

Criteria. A behavior is *lifted* when the adapter raises the model's own rate on it by more than 0.05 over the 40 held-out answers (Figure 23a, b), and *reached* when the model produces it in a majority of its answers, the criterion the prompt is held to and the one Figure 23c uses for both adapters.

Training target and learning rate. A pretraining checkpoint's own answers under a system prompt do not contain the behavior: at 9B tokens they hold the requested script in 0 of 49 writing systems. We therefore train every checkpoint toward the instruct model's answers to the same 160 questions, one target for every checkpoint, and score every answer against the instruction itself. That target produces 42 of the 49 writing systems, 38 of the 45 languages and 20 of the output forms, and those are the counts below. The learning rate is selected per checkpoint, class and method, from 4 rates for the shift adapter and 3 for the LoRA, with a higher rate added wherever the highest one won, as the rate that reaches the most behaviors at seed 0, and all 3 seeds are read at it (Appendix F.4).

Results. At 1B tokens OLMo-2-7B produces none of the 42 writing systems by any route. At 9B tokens two demonstrations reach 16 of them, a rank-8 LoRA 40 to 41 across the 3 seeds, and a trained shift adapter 5 to 6. From 105B tokens on, the three routes stay close to the whole set: the prompt between 30 and 41, the shift adapter between 33 and 42, and the LoRA between 40 and 42 at every later checkpoint (Figure 23c). At 7 further checkpoints between 9B and 105B tokens, a

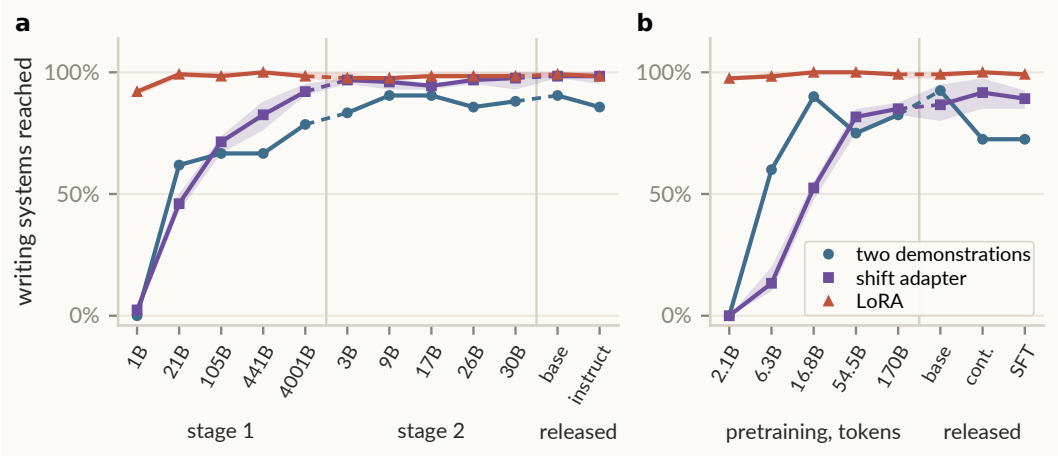

Figure 24: **The same three routes along the training of two further models, on the writing systems each training target produces. (a)** OLMo-2-1B, a second model with two pretraining stages, at 12 checkpoints over 42 writing systems. **(b)** Pythia-12B, another corpus and another laboratory, with no second pretraining stage, at 8 checkpoints over 40. A behavior counts when the model produces it in a majority of the held-out answers, the criterion Figure 23c uses. The learning rate is selected per checkpoint and method at seed 0, and the band spans the 3 seeds read at that rate. The horizontal axis changes at each ruled boundary, so a segment that crosses one is dashed.

trained shift adapter reaches 6, 10, 22, 21, 29, 30, 32, 39 and 41 of the 42 writing systems at 9B, 13B, 21B, 30B, 42B, 55B, 68B, 84B and 105B tokens, while a rank-8 LoRA stays between 37 and 42. The model holds these writing systems in potentiality at 9B tokens, where a rank-8 LoRA already actualizes almost all of them, and the shift adapter reaches them gradually between 9B and 105B tokens. The other two classes follow the same order. On the 38 languages the prompt reaches 3 at 9B tokens, the shift adapter 0 to 1 and the LoRA 31 to 34, and at 105B tokens the three reach 11, 16 to 26 and 35 to 37. On the 20 output forms the LoRA reaches 12 to 13 at 9B tokens, against 3 to 4 for the shift adapter and 5 for the prompt, and from 105B tokens on all three sit between 9 and 15. Over all 100 behaviors the LoRA reaches 84 at 9B tokens, the shift adapter 9 to 11 and the prompt 24, and at the instruct model the three reach 90 to 91, 88 to 93 and 82 (Figure 23b). The distance between the LoRA and the shift adapter is widest at the earliest checkpoints.

The prompt along the training. Over all 100 behaviors the prompt falls from 91 at the supervised model to 82 after preference tuning and at the instruct model. Open-Assistant's supervised chain from Pythia-12B shows the same fall, from 37 of the 40 writing systems at the base model to 29. On both recipes the instruction-tuned model follows a bare sequence of demonstrations less closely than the base model it came from, while both adapters reach the same behaviors. From 105B tokens on, the share of the prompt's behaviors that a shift adapter reaches does not depend on how strongly the prompt reaches them: it is within 6 points whether the prompt reaches a behavior in all eight of its answers or in fewer (AUC 0.30 to 0.51, permutation $p \geq 0.469$). At 9B tokens, the few behaviors a shift adapter reaches are those the prompt reaches best, 5 of 5 against 16 of 23 (AUC 0.76, $p = 0.023$).

F.3 Two further models: OLMo-2-1B and Pythia-12B

OLMo-2-1B, a second model with two pretraining stages, repeats the measurement at 12 checkpoints with the same target and the same rate rule (Figure 24a). At 1B tokens a rank-8 LoRA already reaches 38 to 39 of the 42 writing systems, where two demonstrations reach none and a trained shift adapter 0 to 2. At 21B tokens the prompt reaches 26 and the shift adapter 18 to 21, and from 105B tokens on the three routes stay between 28 and 42, reaching 38, 41 to 42 and 41 to 42 at the released base model. Pythia-12B, trained on the Pile by EleutherAI without a second pretraining stage, has no instruct model of its own, so its adapters train toward the answers of Llama-3.1-8B-Instruct to the same questions. A rank-8 LoRA reaches 39 to 40 of the 40 writing systems that target produces at every one of its 8 checkpoints, including the first at 2.1B tokens, where the prompt and a trained

shift adapter reach none (Figure 24b). A trained shift adapter reaches 4 to 8 at 6.3B tokens, 19 to 22 at 16.8B, 30 to 34 at 54.5B and 32 to 39 from 170B tokens on, against 24 to 37 for the prompt over the same range. In all 3 models a rank-8 LoRA reaches these behaviors at the first checkpoint we measure, and a shift adapter reaches them later, in Pythia-12B after about 50 times more tokens. The potentiality and the shift adapter that actualizes it are not acquired together.

F.4 The learning rate along the training

The selected rate varies along the training and between the classes of behavior. Over the 92 runs of a checkpoint and a class, the shift adapter takes 3×10^{-3} in 66, 10^{-3} in 13, 10^{-2} in 12 and 3×10^{-2} in 1, and the rank-8 LoRA 2×10^{-4} in 51, 6×10^{-4} in 28, 6×10^{-5} in 12 and 2×10^{-3} in 1. Wherever the highest rate of the range wins, a higher rate is also trained and loses. The separation at 9B tokens does not depend on the selection: no rate of the shift adapter reaches more than 6 writing systems there, while the LoRA reaches 40 at three of its rates.

F.5 What a shift adapter reaches early is new text

A behavior a shift adapter reaches early could be text the checkpoint has seen and repeats. We count exact n-grams of the shift adapter's own generations for the writing systems in the Dolma corpus (Soldaini et al., 2024) that OLMo-2 was trained on, through the infini-gram index (Liu et al., 2024a). The share found falls with length, 58% of bigrams, 11% of 5-grams and 5% of 10-grams, which shows that the index finds text where it exists. A behavior's corpus overlap and the checkpoint at which its shift adapter first reaches it correlate at $\rho = -0.15$ over 99 behaviors (Figure 25). A shift adapter that reaches a behavior early makes the model write new text in it.

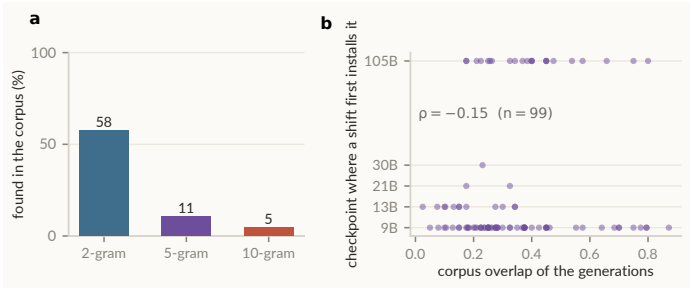

Figure 25: **Corpus overlap of the generations. (a)** The share of the n-grams of a shift adapter's own generations that appear in Dolma, by length, over 99 behaviors. **(b)** The corpus overlap of each behavior against the checkpoint at which its shift adapter first reaches it, at the rate that checkpoint selects.

F.6 Width of the region in magnitude and direction

We train a shift adapter for each of the setup's 4 traits on the 10 instruction-tuned models at 3 seeds, and vary its magnitude and its angle to the trained direction, scoring 256 held-out probes at every point (40 pairs, 120 runs). Each pair of model and trait trains at its own rate, since the rate that maximizes trained adherence spans a factor of 40 across the 10 models and varies as widely between the traits of one model. We select it from 5 rates between 5×10^{-5} and 2×10^{-3}, as the rate with the highest trained adherence, with ties going to the smaller rate.

Pooled adherence is 0.015 [0.010, 0.021] at 0.25× the trained magnitude, 0.057 [0.037, 0.080] at 0.5×, 0.432 [0.382, 0.485] at the trained magnitude, 0.131 [0.075, 0.191] at 2×, and 0.010 [−0.046, 0.072] at 4×, and a 90° rotation leaves −0.003 [−0.004, −0.001] (Figure 26a). Rotated by 45°, the shift adapter keeps more at twice the trained magnitude (0.267) than at the trained magnitude (0.204): at twice the magnitude its component along the trained direction is 1.41 times the trained one, the component that sets the edge (Appendix F.7). At twice the trained magnitude, bullets keep 52% of their peak and French 45%, while pirate and grumpy critic fall below the base model (−10% and −62%, Figure 26b). The peak reproduces across the 3 seeds of a pair: the standard deviation of its adherence over the seeds has a median of 0.010 and is at most 0.085 over the 40 pairs.

Distance from the base model's states. We also measure how far the adapted state at the last input token lies from the base model's, on 64 held-out prompts over the second half of the layers: the median over prompts of the mean distance to the 8 nearest base states, divided by the same distance among the base states, so that 1 is as close as the base states lie to each other. The distance rises with

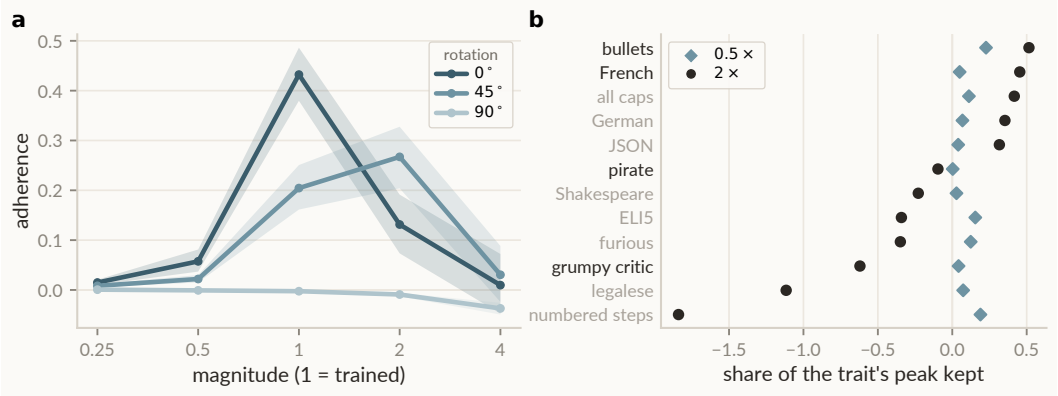

Figure 26: **A region is narrow in magnitude.** Ten models and 3 seeds at the selected rates. **(a)** Adherence against the magnitude of the shift adapter (log axis, 1 is the trained magnitude) with the shift adapter rotated by $0°$, $45°$ or $90°$ toward a random orthogonal direction, mean over the 120 runs of the setup's 4 traits with bootstrap 95% intervals. **(b)** Per trait, the share of its own peak kept at half and at twice the trained magnitude, without rotation: the setup's 4 traits in black and the 8 traits of the robustness check in grey (109 pairs in all). The rotation toward the three kinds of direction is in Figure 27.

magnitude, from 0.91 at $0.25\times$ and 1.04 at $0.5\times$ to 1.52 [1.47, 1.58] at the trained magnitude, 1.84 at $2\times$ and 2.41 at $4\times$. Rotated by $90°$ at the trained magnitude, the state lies at 0.95 [0.92, 0.98]. Below half the trained magnitude, the state stays among the base model's states and the behavior is not produced. Above the trained magnitude, the state moves further from them while adherence falls.

Robustness on 8 further traits. The same measurement on 8 further traits (all caps, German, JSON, numbered steps, legalese, ELI5, Shakespearean and furious) gives the same shape. In 11 of their 80 pairs no rate trains the trait, so there is no region to measure: *furious* on 4 models, *numbered steps*, *JSON* and *legalese* on 2 each, and *all caps* on Qwen2.5-1.5B. Pooled over all 12 traits (109 pairs, 327 runs), adherence is 0.009 at $0.25\times$, 0.039 at $0.5\times$, 0.335 at the trained magnitude, 0.046 at $2\times$ and -0.068 at $4\times$, and 0.001 after a $90°$ rotation. At twice the trained magnitude, all caps (42%), German (35%) and JSON (32%) keep a third or more of their peak, and the other five fall below the base model, down to -184% of the peak for numbered steps (Figure 26b, grey). The collapse below the trained magnitude and the disappearance at $90°$ hold for every one of the 12 traits.

F.7 Where the edge of the region falls

To locate the edge, we measure the same 10 models, 12 traits and 3 seeds at 9 angles and 7 magnitudes near the trained scale, at the selected rates (327 runs, 321 with a trained gain). Adherence halves $41.1°$ from the trained direction (quartiles 35.6 to 47.2, range 18.9 to 80.4 over the 109 pairs). An edge is read per run as the point where the straight line between two neighboring grid points crosses the threshold, and we report the median over runs. On the 4 traits of the setup, on 17 models (201 runs, the ten instruction-tuned models and the base models, Qwen2.5-0.5B and Qwen3.5 on the same grid), the edge at 90% of the trained shift adapter's gain runs from 0.84 to 1.24 times the trained magnitude and to $31.1°$, and the edge at half the gain from 0.71 to 1.65 times and to $43.6°$. As a robustness check on the 10 instruction-tuned models, the 4 traits give 0.86 to 1.27 times and $29.9°$, and 0.75 to 1.67 times and $41.4°$, and all 12 traits give 0.86 to 1.21 times and $28.1°$, and 0.74 to 1.55 times and $41.2°$, so the 8 further traits change none of them. The same grid on the 12 verifiable instruction types of IFEval (Zhou et al., 2023), on 11 models (262 runs with a trained gain above 0.05), puts the edge at 90% of the gain from 0.91 to 1.05 times the trained magnitude and at $15.1°$, and the edge at half the gain from 0.57 to 1.24 times and at $57.3°$. There a program decides compliance with no teacher, and the shape is the same: a narrow band in magnitude and a wide one in direction. On the 5 rule behaviors of the setup, on 17 models at 3 seeds (255 runs), the edge at 90% of the gain runs from 0.69 to 1.39 times the trained magnitude and to $42.6°$, and the edge at half the gain spans the whole measured range of magnitudes, 0.5 to 2 times, and reaches $58.3°$. Per behavior the half-gain angle runs from $52.3°$ (lookup) to $67.6°$ (copy), wider than on

any trait. The per-trait medians span only $36.6°$ (*grumpy critic*) to $46.7°$ (*bullets*), and the widest run of magnitudes holding half the peak is $1.76\times$ from end to end. A magnitude above the trained one and a magnitude below it do not cost the same, and the median and the mean disagree in sign: the median pair keeps 0.071 more of its peak at $1.5\times$ than at $0.7\times$ and 59 of 109 pairs favor $1.5\times$, while the mean runs the other way at -0.186 because the minority that fails when overdriven loses the behavior outright. A larger magnitude usually costs less of the peak than a smaller one, and in a minority of pairs it removes the behavior.

Write a perturbed state in the coordinates the perturbation itself uses, a component $\alpha\cos\theta$ along the trained direction and a component $\alpha\sin\theta$ across it. The second is exact, since the perturbation orthogonalizes the added direction against the trained one before rotating. The first alone sets the edge. One shared curve in $\alpha\cos\theta$ accounts for a median 0.439 of the variation within one run, against 0.185 for the distance from the trained state, 0.152 for the angle and 0.073 for the magnitude (20,223 measurements over 321 runs, with repeated seeds putting the reproducibility ceiling at 0.95). On the traits, the band of $\alpha\cos\theta$ that holds half the peak runs from 0.73 to 1.56 (0.66 to 1.71 with the rule behaviors), and that fall-off barely moves as the across component grows to twice the trained norm (largest spread 0.18 of the peak at any fixed $\alpha\cos\theta$, 0.16 inside the band). The angle at which adherence halves therefore climbs with magnitude, from $28.3°$ at $0.85\times$ to $41.1°$ at the trained magnitude and $66.4°$ at $2\times$, which is what a constant $\alpha\cos\theta$ predicts. A region is therefore a band in that one component: what a state keeps depends on how much of the trained direction it retains.

F.8 What the shift adapter is rotated toward

We measure how far the edge depends on the kind of direction the shift adapter is rotated toward. The magnitude is held at the trained one and the added direction is drawn three ways: at random, inside the span of the other traits' shift adapters, and along another trait's own shift adapter, on the same 10 models and 12 traits at the selected rates (327 runs, 321 with a trained gain). On the 4 traits of the setup, on 17 models (201 runs), the shift adapter keeps 90% of its gain up to $31.4°$ toward a random direction and $12.1°$ toward another trait's shift adapter, and half of it up to $43.7°$ and $24.7°$. Toward a direction inside the span, measured on the 10 instruction-tuned models only (120 runs), it keeps 90% up to $24.6°$ and half up to $38.9°$. As a robustness check, over all 12 traits on those 10 models, adherence halves at $41.9°$, $35.3°$ and $21.7°$, and keeps 90% up to $30.2°$, $21.5°$ and $10.7°$. On the 5 rule behaviors (255 runs), turning toward another behavior's state halves the gain at $35.9°$ and keeps 90% up to $22.1°$, against $58.3°$ and $42.6°$ toward a random direction (Appendix F.7). A region is about twice as wide across the directions the model does not use for another trait as across the directions it does, and the random edge exceeds the other-trait edge in 313 of the 321 runs and in all 12 traits. The in-span edge sits below the random edge in 10 of the 12 traits and above it for *French* and *German*, whose span holds the other languages. The random edge agrees with that of Appendix F.7, $41.9°$ against $41.1°$, on separate runs (Figure 27).

F.9 Connectivity of the region

We score adherence along the straight segment between two states of one behavior trained at different seeds, on 6 models. We summarize each path by its lowest interior point, as a share of the mean adherence of its two endpoints. The path stays adherent along its whole length: over 126 segments on 5 traits, the lowest interior point keeps a median 100% of the endpoint gain (100% on bullets and French, 54 segments, and 95% on one word, the lowest trait), and the worst per-model mean barrier, over the segments scored at two evaluation seeds, is -0.019 (Mistral-7B), at endpoint cosines from 0.16 on Llama-3.1-8B to 0.91 on Gemma-2-2B (Figure 28). On the 5 rule behaviors of the setup, on 17 models at 3 seeds, the lowest interior point on the line between two trained states keeps a median 99% of their gain (255 lines), from 90% on the cipher and 96% on the lookup to 100% on copy, while the same line toward a random direction of the same norm keeps 0% (85 lines). Two states that share a sixth of their direction still connect without a trough, so the states that produce one behavior form one connected region. The two states need not share their supervision. A state trained on external gold answers (the base model's own answers translated by a separate machine-translation model) and one distilled from a system prompt stay connected in all 11 models, and this holds in each of 3 languages: the lowest interior point keeps a median 98% of the endpoint gain in French and German and 100% in Spanish, never below 86%, with the target language above 0.87 at every midpoint, 33 of 33 model–language pairs. Both routes reach the language they were trained

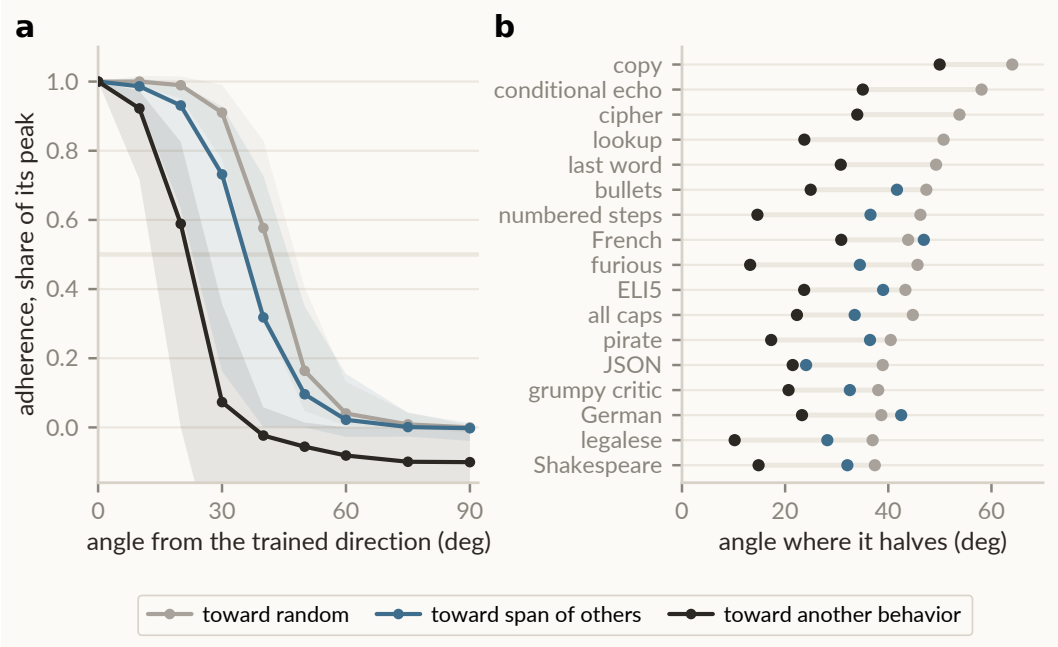

Figure 27: **A region is wide only across the directions the model does not use. (a)** Adherence as a share of each run's peak against the angle from the trained direction, at the trained magnitude, for each kind of added direction, median over 10 models and 12 traits with quartile bands. All three start from the same trained state and are orthogonalized against it, so only the kind of direction differs: turning toward another behavior's own state costs half the peak by 22°, toward a random direction not until 42°. **(b)** The angle at which adherence halves, per behavior (12 traits and 5 rule behaviors), median over models, toward each kind of direction. The ordering holds in every behavior. Figure 7 draws the region itself over magnitude and angle.

for, so no path is scored on a state that never produced it (lowest rate over the 33 runs per language 0.92, 0.88 and 0.88, medians 1.00, 1.00 and 0.96). A random direction of the same norm connects in none of the 33. The two states connect although a nearest-neighbor ranking of their fields places them far apart, 249th of a model's 441 to 782 trait states in French, and 13th to 20th in German.

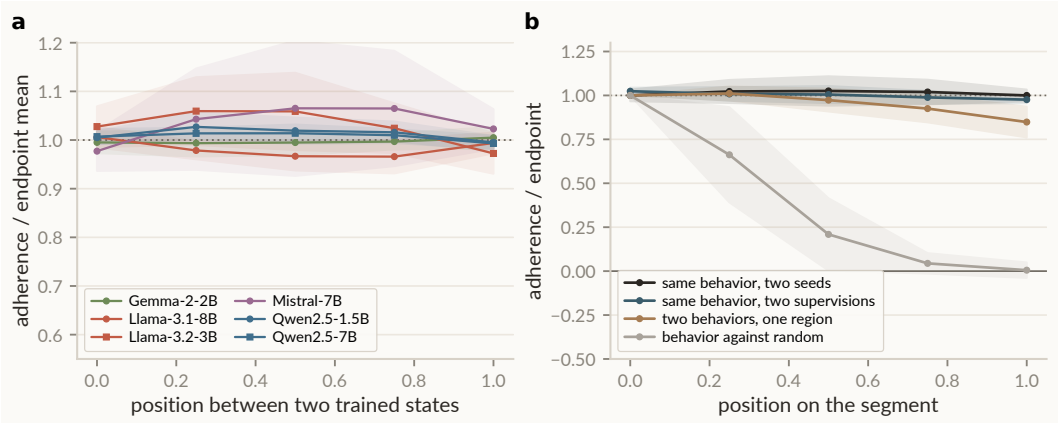

Figure 28: **Independently trained states of one behavior form a single connected region. (a)** Adherence along the straight segment between two independent states of one behavior, normalized to the endpoint mean, for 6 models: the path stays flat. **(b)** The same instrument on four kinds of endpoint pair. Two states of one behavior stay flat whether they differ by seed or by supervision (a gold-supervised French shift adapter against a prompt-distilled one, 11 models), two behaviors that are behaviorally one region blend, and a behavior against a norm-matched random direction falls to the base model.

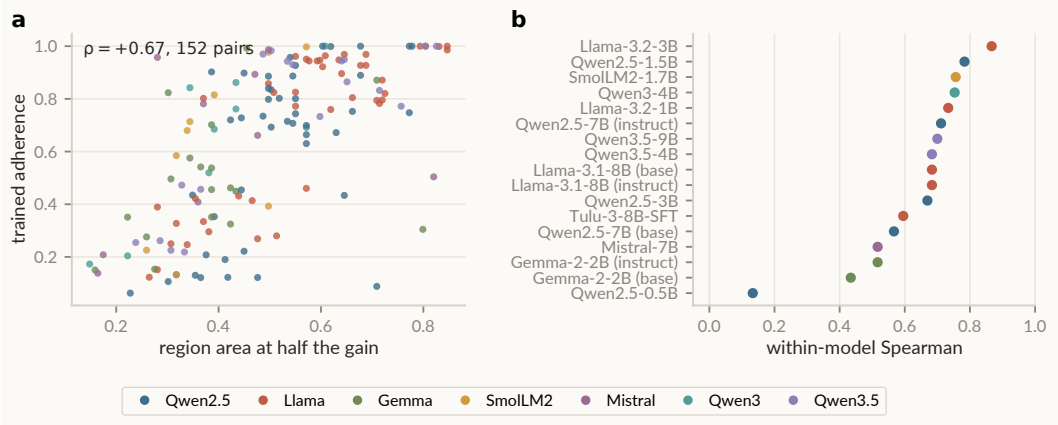

Figure 29: **A larger region gives a stronger behavior, in every model. (a)** Per pair of model and behavior, the area of the region, the share of the magnitude-by-angle grid of Appendix F.7 on which a perturbed shift adapter keeps half the trained gain, against the adherence of the trained state, each the mean over runs. 152 pairs on the 17 models, colored by family. **(b)** Per model, the Spearman correlation of the two over its behaviors.

F.10 Region size and the strength of the behavior

On the setup. For each pair of model and behavior, we take the area of the region of actualizing states, the share of the magnitude-by-angle grid of Appendix F.7 on which a perturbed shift adapter keeps half the trained gain, and the adherence of the trained state, each as the mean over runs. The pairs are the 4 traits and 5 rule behaviors of the setup on the 17 models, 152 in all. A larger region gives a stronger behavior (Figure 29). Within a model, the area correlates with the adherence at Spearman $+0.63$ on average (median $+0.68$) over the 17 models, and the correlation is positive in every one of them, from $+0.13$ on Qwen2.5-0.5B to $+0.87$ on Llama-3.2-3B. Over all 152 pairs it is $+0.67$ ($p < 10^{-20}$).

What a region looks like. Figure 30 slices the shift space of 10 models on the plane through 4 independently trained states of each of 5 behaviors. French, pirate and excited child occupy bounded regions whose size differs across models, while bullets and one word fill the whole measured plane, so their regions exceed the window and their reported area is a lower bound. The mean area over the 5 behaviors runs from 0.48 on SmolLM2-1.7B to 0.84 on Qwen2.5-7B.

Along training. At each of 6 stages of OLMo-2-7B, we measure the region on the plane through 3 independently trained states of each of 10 behaviors that a shift adapter reaches by 449B tokens, as the share of the plane whose adherence stays at or above half its peak. The area rises from 0.13 at 449B tokens and 0.17 at the end of stage one to 0.40 at the released base and 0.78, 0.80 and 0.79 after supervised, preference and instruction tuning (Figure 31). Across these 60 planes, the area predicts the adherence at the center at Spearman $+0.84$, with a positive correlation within every stage.

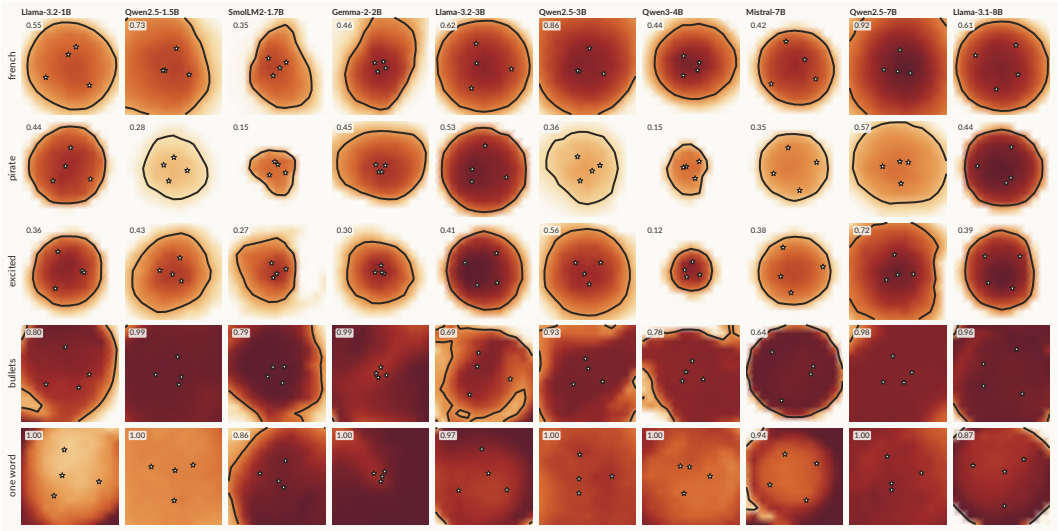

Figure 30: **The region of actualizing states, across behaviors and models.** Each panel is a plane through the shift space of one model, spanned by 4 independently trained states of one behavior (the stars). Every pixel is one shift adapter, added to the model and scored on generated text, so the surface is measured, not fitted. The bold line is the edge of the region at half the peak, and the number in each panel is the area above that edge. Columns are the 10 models from the smallest to the largest, and each row is scaled to its own peak. The top 3 behaviors occupy bounded islands whose size differs across models, while the two format behaviors fill the frame.

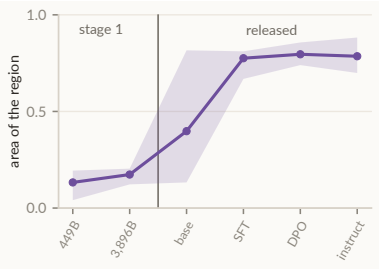

Figure 31: **Post-training makes the regions larger.** Area of the region of actualizing states on the plane through 3 trained states of each behavior, median and quartiles over 10 behaviors trained at 449B tokens, at 6 stages of OLMo-2-7B (Section 4).

G Implications

G.1 Steering: extraction against training

Steering adds a vector to the hidden states of the model. We compare vectors extracted without training with the shift adapter trained on the same behavior, on the 9 behaviors of the setup, 17 models and 3 seeds.

Experimental setup. Three extraction methods give one steering vector per behavior: the mean difference of CAA (Rimsky et al., 2024), the leading principal component of the same differences (Zou et al., 2023), and the in-context task vector (Hendel et al., 2023). Each is added at the single best layer or at every layer, 6 variants in all, each added the way the shift adapter adds its vectors, one vector per layer. An extracted vector gives a direction but not a strength, so the strength of each variant is chosen from 5 values on 200 held-out instructions, and every method is scored on 800 further held-out instructions. The shift adapter trains on 400 instructions for 200 steps. For each run, the best extracted direction is the variant with the highest adherence on the 800 instructions, a choice that favors extraction. Scores follow Appendix C.6: adherence on the traits, where 0 is the base model, and the exact-match rate on the transformations and maps.

Table 8: **Extracted directions against the trained shift adapter on the setup.** Held-out score on 800 instructions per run (adherence on the traits, exact match on the transformations and maps), mean over runs with a bootstrap 95% interval over models, 17 models and 3 seeds. The best extracted direction is the best of the 6 extracted variants in each run.

behavior	shift adapter	best extracted direction	runs
bullet points	0.81 [0.67, 0.92]	0.76 [0.57, 0.92]	51
French	0.39 [0.33, 0.44]	0.01 [−0.03, 0.04]	51
pirate	0.29 [0.23, 0.34]	−0.03 [−0.08, 0.04]	51
grumpy critic	0.16 [0.13, 0.20]	−0.06 [−0.10, −0.02]	51
copy	0.99 [0.97, 1.00]	0.07 [0.03, 0.12]	51
last word	0.85 [0.73, 0.94]	0.01 [0.01, 0.02]	51
conditional echo	0.78 [0.67, 0.86]	0.22 [0.16, 0.28]	51
cipher	0.80 [0.74, 0.85]	0.42 [0.32, 0.51]	51
lookup	0.86 [0.79, 0.92]	0.04 [0.01, 0.07]	51
traits	0.41 [0.36, 0.46]	0.17 [0.12, 0.22]	204
transformations	0.87 [0.79, 0.93]	0.10 [0.07, 0.13]	153
maps	0.83 [0.77, 0.88]	0.23 [0.18, 0.28]	102
all 9	0.66 [0.60, 0.70]	0.16 [0.13, 0.19]	459

Results. Pooled over the 9 behaviors, the trained shift adapter reaches a score of 0.66 [0.60, 0.70], against 0.16 [0.13, 0.19] for the best extracted direction (Table 8). Per class, the shift adapter reaches 0.41 on the traits against 0.17, 0.87 on the transformations against 0.10, and 0.83 on the maps against 0.23. The extracted directions come close to the trained shift adapter only on bullet points (0.76 against 0.81), and stay at or below the base model on French, pirate and grumpy critic, where the shift adapter reaches 0.16 to 0.39. On the transformations and maps, the best extracted direction reaches at most 0.42, on the cipher, against 0.78 to 0.99 for the shift adapter.

G.2 Steering on AxBench

AxBench (Wu et al., 2025) steers on 500 concepts drawn from a sparse autoencoder of Gemma-2, each a text description with positive and negative examples. We train a shift adapter and a rank-8 LoRA on Gemma-2-2B for every concept, 6 rates for the shift adapter and 4 for the LoRA, 3 seeds, and score them on the two axes of the benchmark. Steering is judged by GPT-4o-mini, the benchmark's judge model, on each generation with its prompts unchanged: the presence of the concept, the relevance to the instruction and the fluency, each from 0 to 2, combined by their harmonic mean. Detection is the concept-against-negative separation the trained state gives when read as a probe. The rate of each is the one with the best steering score, 3×10^{-2} for the shift adapter and 6×10^{-4} for the LoRA, and both ranges continue past their best rate. Each method's rate is selected and scored on the same held-out instructions, as AxBench selects its steering strength. Each concept is judged on its own held-out instructions at 64 generated tokens, where the benchmark's leaderboard judges 10 fixed instructions at 128 tokens, so our levels are not its leaderboard numbers and the comparison is between the two methods under one protocol. On adherence to the reference answer, the score of the main text, the shift adapter reaches 0.064 against 0.073 for the LoRA, a paired difference of -0.009 $[-0.011, -0.007]$ (Figure 32a). At the rate the judge selects, the shift adapter puts the concept into more of its answers and keeps less of the reference wording, and at 3×10^{-3} its adherence is 0.090, above the LoRA's. On the benchmark's steering score the shift adapter reaches 0.867 and the LoRA 0.914, a paired difference of -0.047 $[-0.072, -0.021]$ over the 500 concepts, -0.05, -0.06 and -0.04 on the 3 seeds (Figure 32b). On the presence of the concept alone the shift adapter scores 0.965 and the LoRA 0.983, and on the other two judged scores it is below the LoRA, by 0.09 on relevance to the instruction and by 0.02 on fluency. On detection the shift adapter reaches an area under the curve of 0.571, above chance on 374 of the 500 concepts, and the LoRA 0.686, above chance on 464, a difference of -0.115 $[-0.126, -0.105]$ (Figure 32c). The rate moves the three scores apart: from 3×10^{-3} to 3×10^{-2} the steering score rises from 0.851 to 0.867, while adherence falls from 0.090 to 0.064 and the area under the curve from 0.607 to 0.571. A shift adapter steers within 5% of the LoRA, and the state of the LoRA separates the concept from its negatives better. Programming and the web make up 20% of the concepts and science 7%, and none of the 500 is one of our behaviors: none is a writing system, a language or an output form, and the one that names a language we

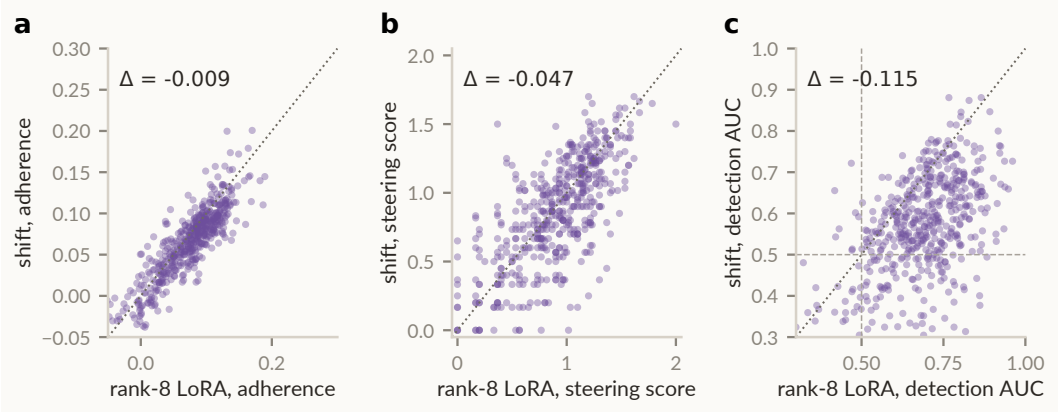

Figure 32: **A shift adapter steers close to the LoRA on AxBench.** The 500 concepts of AxBench on Gemma-2-2B, a shift adapter against a rank-8 LoRA, paired per concept, at the rates the judged steering score selects. **(a)** Adherence, the score of the main text: the overlap of the generation with the reference answer, as a gain over the base model. **(b)** Steering score, the benchmark's judge on each generation. **(c)** Concept detection, the area under the curve of the trained state read as a probe, with chance as dashed lines. In each panel, Δ is the mean over concepts of the shift adapter's score minus the LoRA's.

test names it as a topic. The parity of Section 3 therefore holds across two concept sets that do not intersect.

Extracted directions on the same protocol. We compare the trained methods with 3 directions extracted without training, as in Section 5, on the same concepts and under the same protocol. The mean difference is, at every decoder block, the mean hidden state over the response tokens of the concept's 48 training examples minus the mean over those of the 64 shared negatives, AxBench's DiffMean and the contrast of CAA. The principal component is the leading component of the same positives centered on the negative mean, sign-aligned with the mean difference (Zou et al., 2023). The in-context direction is the last-token displacement that a prompt of 4 of the concept's examples induces over the bare prompt, averaged over 16 of its instructions (Hendel et al., 2023). Each direction is added the way the shift adapter adds its vectors, one vector per block, the unit direction at a block scaled by a multiple of the base model's activation norm there, at every block or, for the mean difference, at AxBench's block 20 alone. The multiples bracket the best value (0.005 to 0.08 at every block, 0.02 to 0.8 at one block), the multiple of each direction is the one with the best steering score over the concepts, the rule that selects the trained methods' rates, and each direction is judged on the same two held-out instructions per concept as the trained methods. Table 9 gives the result. The best extracted direction, the mean difference at every block, reaches a steering score of 0.207 at multiple 0.04 and 0.017 at 0.08, where the generations degenerate. At block 20 it reaches 0.180, the principal component 0.170 and the in-context direction 0.166. The trained shift adapter reaches 0.867 and the LoRA 0.914 on the same concepts, a paired difference of -0.660 $[-0.697, -0.623]$ between the best extracted direction and the shift adapter. On the concept sub-score the extracted directions reach 0.16 to 0.26 against 0.97 for the shift adapter, and they raise the steering score above zero on 79 to 132 of the 500 concepts against 435 for the shift adapter and 446 for the LoRA. Their relevance sub-scores are above those of the trained methods, and so is their fluency, except for the mean difference at every block (0.79). The level of the mean difference at block 20, 0.180, is the level the benchmark's own board reports for it, 0.178 (Wu et al., 2025), under a protocol that differs in the instructions and the length judged.

Table 9: **Extracted directions against the trained shift adapter on AxBench.** The 500 concepts on Gemma-2-2B, each direction at the multiple of the base activation norm (extracted) or the learning rate (trained) its steering score selects, steering score and its three sub-scores from 0 to 2, mean over concepts. The column > 0 counts the concepts on which the score is above zero, and the last column gives the paired difference to the trained shift adapter with a 95% bootstrap interval over concepts. The extracted directions have one seed, the trained methods three.

direction	scale	score	concept	relevance	fluency	> 0	minus the shift adapter
mean difference, every block	0.04	0.207	0.26	1.73	0.79	132	-0.660 [-0.697, -0.623]
mean difference, block 20	0.4	0.180	0.18	1.92	1.19	90	-0.687 [-0.727, -0.646]
principal component, every block	0.005	0.170	0.16	1.94	1.32	79	-0.698 [-0.736, -0.657]
in-context direction, every block	0.01	0.166	0.17	1.94	1.34	80	-0.701 [-0.740, -0.661]
shift adapter	3×10^{-2}	0.867	0.97	1.49	1.12	435	—
rank-8 LoRA	6×10^{-4}	0.914	0.98	1.58	1.14	446	—

G.3 Convergence across models

Conventions. The 54-of-55 permutation count in Section 5 is a pairwise comparison and carries no family-wise correction. Every comparison of the 11 atlases uses the 75 behaviors that all of them keep (Appendix C.4). The association with capability is unchanged ($\rho = 0.71$) when capability is measured in bits per byte instead of log-likelihood per token. RSA and mutual-kNN rise with the capability of the pair and CKA does not (Figure 33): over the 55 pairs, RSA at Spearman $\rho = 0.71$ ($p = 0.005$), mutual-kNN at $\rho = 0.47$ ($p = 0.033$) and CKA at $\rho = -0.29$ ($p = 0.83$), each p from a one-sided permutation of the model labels. The 3 measures differ in what they cannot see. RSA ranks the cosines between states, so it does not see one state rescaled or the cosines changed in any order-preserving way. CKA centers the atlas first, so it does not see the atlas moved as a whole or rescaled. Mutual-kNN reads neighbor sets, so it does not see any change that leaves each state's neighbors in place. A rotation of the coordinates is invisible to all three: a similarity between two atlases reads the pattern of distances among their states, not their coordinates, which differ from model to model. Figure 34 draws the similarity over the 55 pairs and through depth.

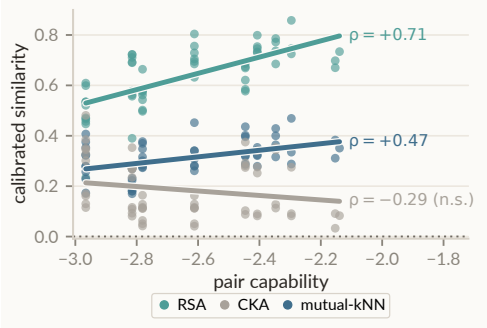

Figure 33: **RSA and mutual-kNN rise with capability, CKA does not.** Calibrated similarity of the atlases of two models against the capability of the pair, for the 55 pairs of the 11 models on the 75 shared behaviors, with a least-squares line per measure and its Spearman ρ at the line's end. The capability of a pair is the lower of the two models' WikiText log-likelihood per token. Figure 8b draws the same pairs for RSA and mutual-kNN.

A larger set of behaviors and models. Every model is offered the same 946 behaviors, covering writing systems, subject matter, personas, historical periods, output forms, emotions, senses and hard constraints. Each is distilled into a shift adapter on 160 shared instructions for 300 steps at one seed, the recipe of Appendix C.4, and a behavior whose held-out adherence is not positive is dropped from that model's atlas, so the number of entries is measured, not set: from 429 (Tulu-3-8B-SFT) to 927 (Qwen2.5-72B), median 705. Each entry is read as the per-layer change the trained shift adapter makes to the hidden state on held-out instructions shared by every behavior. The 11 models of Section 5 contain no base model, no intermediate training stage and nothing above 8B, so this part uses 18 models: those 11, Qwen2.5-14B, 32B and 72B, which extend the Qwen2.5 sizes, and Llama-3.1-8B with the three Tulu-3 stages, one pretrained model followed through supervised,

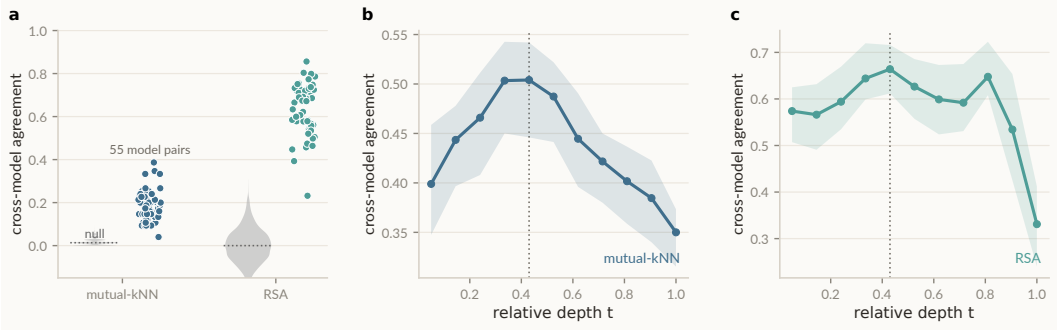

Figure 34: **The arrangement is shared and agrees most in the middle layers** (11 single-seed atlases, 75 shared behaviors). **(a)** Cross-model agreement under mutual-kNN and RSA over the 55 model pairs, against the permutation null (60 permutations per pair). **(b)** Mutual-kNN and **(c)** RSA against relative depth, mean over the pairs with quartile bands: both peak at relative depth 0.43 and fall at the readout.

preference and reinforcement tuning. 5 of the 18 are one pretrained model at different stages and are not independent draws.

The value a similarity can reach. Section 5 reports a similarity of 0.621 between models of different families. Two atlases of the same model already differ, because training the behaviors is itself stochastic, and that difference bounds every comparison between two models. We retrain the same 201 behaviors at two further seeds and compare the atlases under the same protocol. The bound depends strongly on the model: Llama-3.1-8B agrees with itself at 0.55, Qwen2.5-7B at 0.73 and Gemma-2-2B at 0.96, each tight across its 3 seed pairs (within 0.008 for Llama and 0.024 for Qwen). A comparison is therefore read against the bound of the model in question. How far a comparison moves when both atlases are retrained is measured by repeating the full atlas at two further seeds for 10 of the models: over the 8 pairs from different families measured at all 3 seeds, the widest range is 0.091, every pair without Mistral-7B lies within 0.044, and by size tier the 3 seeds agree within 0.052. The comparisons below are reproducible to about 0.05. The bound is measured on 201 behaviors while the comparisons use the larger shared sets, so it gives the right order, not an exact denominator.

Size and family. Two models of different families at 7 to 8B agree at 0.50 over 3 pairs, 91% of the Llama bound and 68% of the Qwen bound. Across the Qwen2.5 sizes, neighboring sizes agree more the larger the pair: 0.35 at 0.5B against 1.5B, then 0.46, 0.56, 0.58, and 0.68 at 14B against 32B, falling to 0.60 at 72B. Across the 3 Llama-3 sizes the agreement rises the same way, 0.33 then 0.44. Across families the same measurement stops rising early, at 0.22 in the 0.5 to 1B tier, then 0.44, 0.43 and 0.50 at 7 to 8B, flat from about 2B on. Models of one family converge on each other as they grow, while models of different families reach a shared arrangement early and then stay at a constant distance. This parallels the convergence reported for representations (Huh et al., 2024; Gröger et al., 2026), here measured on trained shift adapters.

Post-training. On the Tulu-3 chain from Llama-3.1-8B, the supervised step agrees with its base at 0.38 [0.31, 0.45], well below that model's bound of 0.55, while the two preference steps agree with their predecessors at 0.68 [0.65, 0.72] and 0.71 [0.69, 0.73], above it. Supervised tuning moves the arrangement, and preference tuning moves it less than retraining the same model at a new seed. Meta's own instruction tuning of the same pretrained model agrees with it at 0.22 [0.18, 0.25], below the 0.50 at which two models of different families agree.

G.4 Neural thickets

Gan & Isola (2026) add Gaussian noise of a fixed size to every weight of a pretrained model, score each perturbed model on a selection split, and keep the best, which they call experts. Under the principle, a perturbation finds an expert only where the region of actualizing states of the behavior is wide enough to be hit by chance (Section 5). We repeat their search on the setup and relate where it finds experts to the size of the region measured in Appendix F.7.

Experimental setup. For each model we draw 300 perturbed models at each noise size, 6 sizes for GSM8K, the task of Gan & Isola (2026), and 3 for each of the 9 behaviors of the setup, with the sizes placed per model in units of the displacement they cause in the hidden state. Each perturbed model is scored on a selection split of 32 items, and the 20 best are scored again on a disjoint held-out split of 64 items, at 3 seeds. A perturbed model is an *expert* when it still answers fluently (distinct-2 of at least 0.5) and gains at least 0.10 over the base model on the held-out split. For each model, task and seed, the *expert fraction* is the share of the 20 re-scored models that are experts, at the noise size where that share is highest. The search covers the 17 models on GSM8K (49 runs) and on the 9 behaviors (432 runs).

Where the search finds experts. On GSM8K the expert fraction is 0.24 on average over the 17 models. It is highest on Qwen2.5-3B (0.87), Qwen3.5-9B (0.85), the Qwen2.5-7B base model (0.68) and Qwen2.5-7B (0.47), followed by Qwen3.5-4B (0.38), Tulu-3-8B-SFT (0.32) and Llama-3.1-8B (0.25), and at most 0.05 on the other 10 models. On the behaviors of the setup it is 0.015 on average: the search finds experts on 14 pairs of model and behavior in 8 models (copy and the conditional echo on the base models of Llama-3.1-8B, Gemma-2-2B and Qwen2.5-7B and on Qwen2.5-3B, the cipher on Qwen2.5-3B, the conditional echo and bullets on Qwen2.5-0.5B, and bullets on Qwen2.5-1.5B, Qwen2.5-7B and SmolLM2-1.7B), and none in the other models.

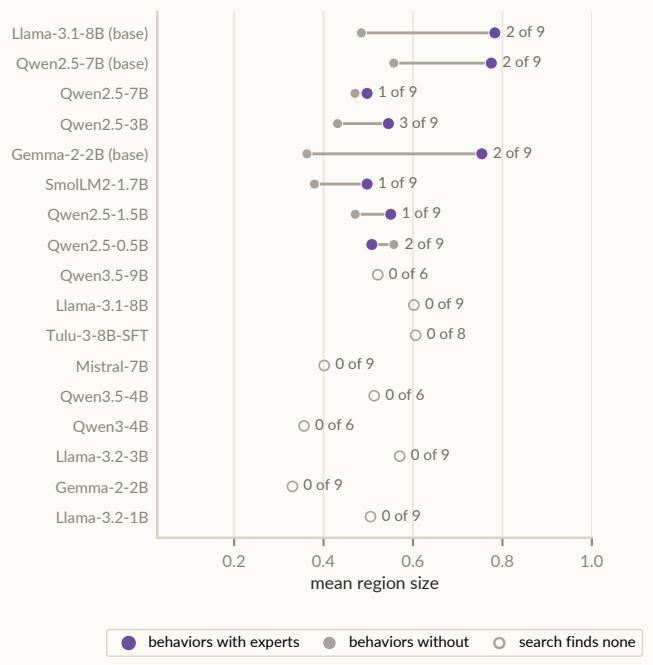

Figure 35: **Random search finds experts on the widest regions, model by model.** Per model, the mean size of the region of actualizing states (the share of the magnitude-by-angle grid on which a perturbed shift adapter keeps half the trained gain, Appendix F.7) over the behaviors of the setup on which the search finds an expert (purple) and over those on which it finds none (grey), with the number of behaviors with an expert. A hollow mark is a model in which the search finds none. All 17 models.

Experts and the size of the region. In the 17 models, the behaviors on which the search finds an expert have wider regions than those on which it finds none (Figure 35). Over the 72 pairs of model and behavior in the 8 models where the search finds any expert, the region size separates the two at an AUC of 0.82 ($p < 0.001$) and correlates with the expert fraction at Spearman $+0.45$ ($p < 0.001$). Over all 143 pairs in the 17 models, the same values are 0.76 and $+0.28$ ($p < 0.001$ each). Split into quartiles of region area, the search finds an expert in 0% of the pairs of the narrowest quartile (mean area 0.29), then in 6% and 11%, and in 22% of the widest (mean area 0.72). The 14 expert pairs have regions covering 0.39 to 0.85 of the grid, and French, pirate and grumpy critic hold no expert in any model.

Relation to trained states. A second search, on 8 models and GSM8K at 3 seeds, with the same splits and the same criterion for an expert, relates the experts to trained states. The fields of its 280 experts align with the field of the trained LoRA at a median signed cosine of 0.25, positive for 278 of them. Against references trained to the same level, an expert's field aligns with a trained shift adapter at a median cosine of 0.24 and with the shared component of a rank-8 LoRA at 0.23, where a random field of the same norm reaches 0.00.

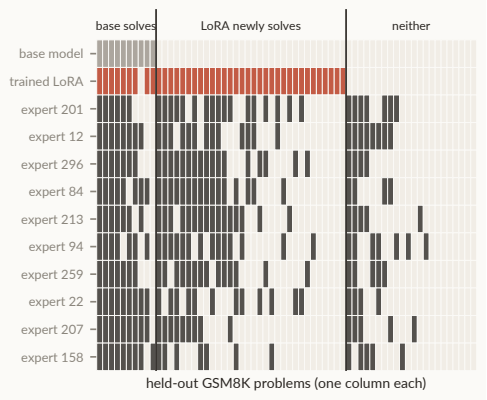

Figure 36: **Experts solve the problems training newly solves.** Held-out GSM8K problems of Qwen2.5-3B (columns), grouped into those the base model solves, those the trained LoRA newly solves, and neither, against the 10 experts of highest gain (rows).

Which problems the experts solve. If an expert reaches the state training reaches, it should newly solve the same held-out problems. For each expert we count the problems it newly solves that the trained LoRA, the reference, also newly solves, against the count expected if it newly solved problems at random among those the base model fails (Figure 36). Over 111 experts on 6 models, the observed count is a median 1.35 times the expected one, and the median ratio is above 1 in every model (1.14 on Llama-3.1-8B to 3.14 on SmolLM2-1.7B).

What a shift adapter recovers. A shift adapter trained on an expert's own answers keeps a median 82% of the expert's gain on Qwen2.5-3B (47 experts) and 63% on Qwen2.5-7B (26 experts), the two models where a shift adapter trained on the answers of the trained reference itself recovers 81% to 88% and 72% to 94% of its gain (Figure 37). Random search reaches states that point the way trained states point.

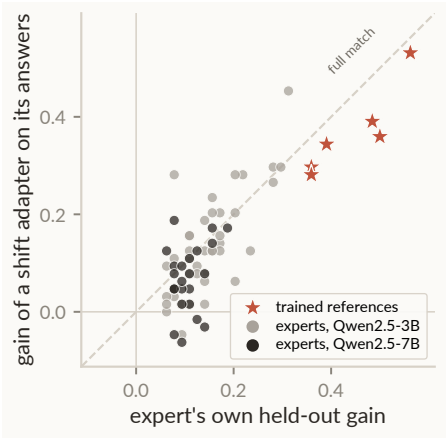

Figure 37: **A shift adapter carries what an expert does.** The gain of a shift adapter trained on each expert's answers against the expert's own held-out gain, on Qwen2.5-3B and Qwen2.5-7B. The dashed diagonal is a full match, and stars are the trained references, whose answers a shift adapter recovers the same way.