Do neuron explanations survive a sequence of narrow fine-tunes?

Will Sackfield, Open Interpretability · https://open-interpretability.com/research/sequential-tuning

Abstract

Benchmarks show that a language model fine-tuned on one narrow task after another loses general skill, but they do not show what happens to the units inside it. We fine-tuned Qwen2.5-0.5B and its instruct model through three narrow tasks in turn, SST-2 sentiment, XSum summaries and GSM8K maths. After each stage, we re-read the probes of the 362 VERIFIED neuron explanations of the base model that reproduce. After three stages, 267 explanations keep on the base model (74%) and 257 on the instruct model (71%). The first stage alone costs the base model 66 explanations, about as many as instruction tuning costs (62), and 28 of them are the same poles, against 11 expected by chance. From the first stage on, the two models keep about the same explanations, so the instruction tune neither protects them nor exposes them. The losses churn: many explanations fail at one stage and pass again at a later one. Zero-shot ARC-Easy accuracy does not track any of this. It ends 3.5 points above its start on the base model, while a quarter of the explanations fail. The stage with the largest KL divergence on its own data costs the fewest explanations.

Status: Preprint. Work left: #6367 (seeds, a control stage, a second size and recovery).

1 Introduction

A neuron explanation is useful only while the model it describes stays the same. Deployed models do not stay the same. They are fine-tuned, often more than once, on narrow data for one product after another. If explanations break at each such step, an explanation registry describes a model that no one runs.

Benchmarks have measured this setting. Luo et al. (2023) fine-tuned models of 1 to 7 billion parameters on five instruction tasks in turn, and found that general knowledge, reasoning and reading comprehension fell. They also found that a general instruction tune before the sequence reduced the fall, for one of the two pairs that they compared. Benchmark scores do not say which internal units changed, or whether the units that changed are the ones that an interpretability method had explained.

This registry has measured one step. An earlier analysis re-tested the VERIFIED explanations of Qwen2.5-0.5B on its instruct model, and found that 81% keep (`docs/analyses/base-instruct.md`). A companion paper showed that most of the further loss inside the chat template comes from the template, not the tuning (template or tuning). Neither asked what a *sequence* of narrow fine-tunes does, or whether instruction tuning changes it.

This paper's contributions are these:

- A re-test of every VERIFIED explanation of Qwen2.5-0.5B after each of three narrow fine-tunes, in the base model and in its instruct model, with paired tests at each stage.
- Evidence that the first narrow fine-tune of the base model costs about as many explanations as instruction tuning does, and that the two costs fall on an overlapping set of poles.
- Evidence that after the first stage, the two models do not differ. Instruction tuning spends its share of explanations first, and the later stages cost both models alike.
- Evidence that losses churn between stages, and that neither a general benchmark nor the KL on a stage's own data tracks them.

The paper bears on research question 4, *Do explanations carry across models?* (research questions). A fine-tuned model is a nearby model, so the question asks how far an explanation carries along a path of them.

2 Related work

Forgetting in sequences of fine-tunes. Luo et al. (2023) give the benchmark version of this paper's question. Their sequence is five generation tasks with 100,000 samples each, trained for three epochs at a learning rate of 2×10^{-5}, and they read general skill from benchmarks such as MMLU and the reasoning sets of lm-evaluation-harness. Their forgetting measure is relative to the initial score and averaged over stages. We keep their learning rate and their form of a fixed sequence, and replace the benchmarks with the registry's explanations, beside one benchmark. Kirkpatrick et al. (2017) proposed elastic weight consolidation, which slows learning on the weights that matter to earlier tasks. We do not test a remedy. In a related design, von Oswald et al. (2020) compare each task's accuracy right after it is learned with its accuracy after all tasks. Our first-failure counts take that comparison to single units and name the stage at which each one first fails.

Forgetting that is not lost knowledge. Zheng et al. (2025) split a language model's score on a task into its alignment to the task's format and the knowledge beneath it. They show that training on a new task can undo the alignment while the knowledge stays, and that a further epoch on half of the old task's data restores the score, in a synthetic setting. Kotha et al. (2024) propose that fine-tuning moves the model's inference of which task a prompt poses, more than it changes capabilities. Jain et al. (2024) find that fine-tuning at a small learning rate mostly adds a thin wrapper over a capability that stays. Their small rates are 10^{-5} and 10^{-6}, below this paper's 2×10^{-5}. All three predict that a benchmark can fall while the units beneath it do not change. This paper tests the converse as well: whether units can fail while the benchmark does not fall.

Which layers change. Ramasesh et al. (2021) trained image networks on two tasks in turn, and found that the layers nearest the output change most and cause most of the forgetting. A single-author preprint on continual fine-tuning of language models reports instead that the MLP blocks of the middle layers change their representations most (Laitinen-Fredriksson Lundstrom-Imanov, 2026). Its review page notes that several of its references do not support their claims, so we cite its result as a report, not a finding. Chen et al. (2026) propose a mechanism at the level of features: narrow supervised fine-tuning pushes correlated features in superposition onto a shared direction, so that general

inputs give spurious activations. Biderman et al. (2024) find that LoRA forgets less than full fine-tuning, which is why this paper's full fine-tune is the harder case for explanations.

What predicts forgetting. Shenfeld et al. (2026) find that forgetting after fine-tuning is predicted by the KL divergence between the old and new models, measured on inputs from the new task. Their strongest evidence is from a toy setting, and the fit is weaker on language models. We read this KL for each stage, on the stage's own held-out data, and compare it with the explanations that the stage costs.

3 Method

Models and units. The two starting models are Qwen2.5-0.5B and Qwen2.5-0.5B-Instruct (Yang et al., 2024). Both have 24 layers of MLP neurons at the same positions. A *pole* is one sign of a neuron's activation, positive or negative. A pole is *VERIFIED* when its natural-language explanation passed the registry's adversarial review. In that review, a reviewer model wrote 10 to 30 probes meant to break the explanation: *hard negatives*, which should not fire the pole, and *holdouts*, which should. A probe fires the pole when its activation reaches 0.25 of the pole's scale. The pole passes when its *specificity*, the share of hard negatives that stay quiet, is at least 0.80, and its *sensitivity*, the share of holdouts that fire, is at least 0.60. On 2026-11-23, Qwen2.5-0.5B had 363 VERIFIED poles.

The keep test. After each stage, each pole's own probes are read on the stage's model, as plain text, and the pole *keeps* when it passes both gates. The pole's scale on a changed model is its recorded scale times a ratio: the pole's largest activation on the changed model over its largest on the base model, on the same 20,000 tokens of the registry's scan corpus. This is the keep test of the earlier analysis. Only poles that keep on the base model before any stage count. That is 362 of the 363, so every share in this paper is out of 362. The same sample gives Pearson's r between each pole's activations on the base model and on the stage's model.

The sequence. Each model is fine-tuned on three narrow tasks, in one fixed order:

1. SST-2, sentiment of a sentence, answered "positive" or "negative" (Socher et al., 2013).
2. XSum, a one-sentence summary of a news article, cut to its first 1,500 characters (Narayan et al., 2018).
3. GSM8K, a grade-school maths problem, answered with the dataset's worked solution (Cobbe et al., 2021).

The order puts a classification task, a generation task and a reasoning task in turn. Each task uses one instruction template with an instruction, an input and a response, the same for both models. So the two models' sequences differ only in where they start. Each dataset is read at one fixed Hub commit (Appendix A).

Training. Each stage draws 2,000 training pairs with a fixed seed and trains every weight for two epochs. The optimiser is AdamW (Loshchilov & Hutter, 2019) at a constant learning rate of 2×10^{-5}, the rate of Luo et al. (2023), with no weight decay. Each step takes 8 pairs, so each stage is 500 steps. The loss is the cross-entropy of the response tokens only. Training ran in bfloat16 autocast with float32 weights on one GPU.

Measures at each stage. Beside the keep test, each stage gives three measures:

- *ARC-Easy accuracy.* All 2,376 test questions of ARC-Easy (Clark et al., 2018), zero-shot. Each question is a prompt, and the model's answer is the choice with the highest mean log-probability per token. The interval is the 95% Wilson interval.
- *Task losses.* The mean loss on the response tokens of 200 held-out pairs of each task, in nats.
- *Stage KL.* For each fine-tune stage, the mean KL divergence from the model before the stage to the model after it,

$$\mathrm{KL}^{(s)} = \frac{1}{|T|} \sum_{t \in T} \sum_{v} p^{(s-1)}(v \mid x^{<t}) \log \frac{p^{(s-1)}(v \mid x^{<t})}{p^{(s)}(v \mid x^{<t})},$$

where $p^{(s)}$ is the model after stage s, $x^{<t}$ the tokens before t, v runs over the vocabulary, and T is the set of response tokens of 50 held-out pairs of the stage's own task. The responses are the dataset's references, not samples from either model.

Tests. Every comparison is paired by pole. For keep, the test is an exact McNemar test: a two-sided binomial test at one half on the poles that change outcome. The overlap of two sets of failing poles is tested with the upper tail of the hypergeometric distribution.

The run is one seed and one task order. It took about 4 hours, most of it the keep tests.

4 Does a sequence of narrow fine-tunes cost explanations?

The question is how many of the 362 explanations still pass their own probes after each stage. Figure 1 gives the share at each stage for both models, and Table 1 gives the paired changes.

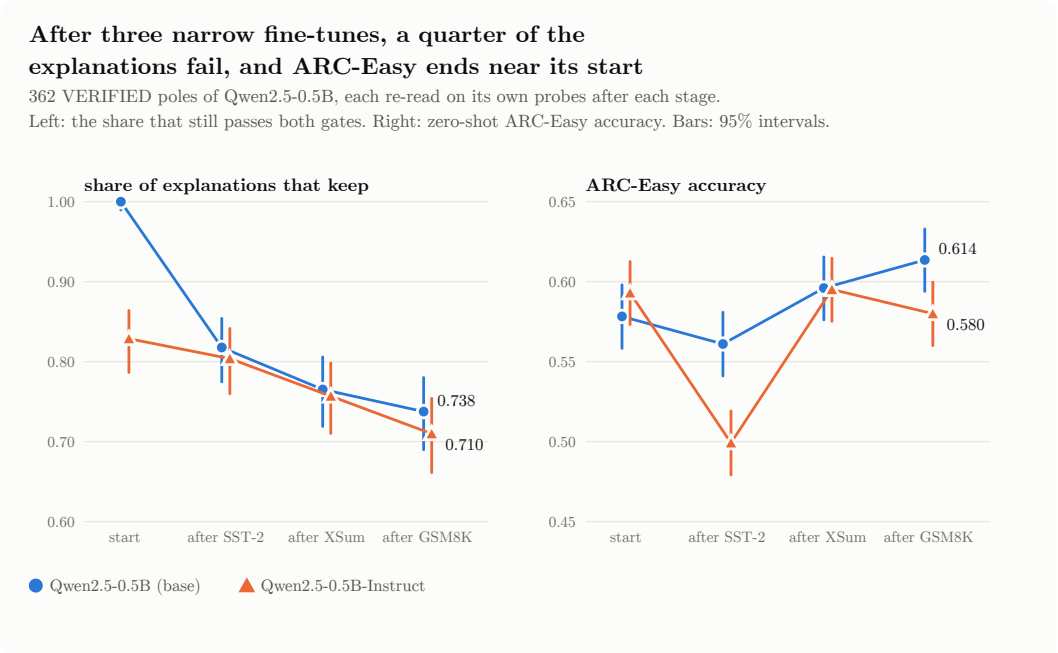

Figure 1. The share of the 362 reproduced explanations that keep (left), and zero-shot ARC-Easy accuracy (right), at the start and after each of three narrow fine-tunes. Circles are the base model and triangles the instruct model. Bars are 95% Wilson intervals.

Table 1. The explanations that stop keeping (lost) and start keeping (gained) at each stage, with an exact McNemar p, and the mean change of each gate over the 362 poles.

Model	Stage	Keep after	Lost	Gained	p	Mean Δ specificity	Mean Δ sensitivity
base	SST-2	296	66	0	2.7×10^{-20}	−0.014	−0.034
base	XSum	277	48	29	0.040	+0.002	−0.029
base	GSM8K	267	31	21	0.21	−0.008	−0.003
instruct	SST-2	291	36	27	0.31	−0.007	−0.012
instruct	XSum	274	43	26	0.053	+0.001	−0.026
instruct	GSM8K	257	35	18	0.027	−0.007	−0.013

The sequence costs explanations at every stage, and the cost falls with each stage. On the base model, the keep share falls from 100% to 81.8% after SST-2, to 76.5% after XSum and to 73.8% after GSM8K (95% interval 69.0–78.0%). The instruct model starts at 82.9%, the cost of its instruction tune, and falls to 80.4%, 75.7% and 71.0% (66.1–75.4%). After three stages, 95 explanations fail on the base model and 105 on the instruct model.

The gates show how the explanations fail. Specificity moves by at most 0.014 at any stage, whereas sensitivity falls by up to 0.034. So a failing explanation mostly stops firing on its holdouts, and rarely starts firing on its hard negatives. The base-instruct analysis and the template paper found the same pattern.

Each stage did learn its task (Figure 3). The held-out loss of each task falls at its own stage: SST-2 from 2.31 to 0.20 nats on the base model, XSum from 3.12 to 2.54 and GSM8K from 1.15 to 0.57. The training loss of each stage falls too (Appendix C). So the explanations fail while the model learns, not because a stage failed to train.

5 Does instruction tuning protect explanations, or spend them first?

Luo et al. (2023) found that a general instruction tune before the sequence reduced forgetting on benchmarks. The question here is whether it does the same for explanations. Table 2 compares the two models stage by stage, on the poles that keep on one model only.

Table 2. At each stage, the explanations that keep on the base model only and on the instruct model only, with an exact McNemar p.

Stage	Base only	Instruct only	p
start	62	0	4.3×10^{-19}
after SST-2	37	32	0.63
after XSum	35	32	0.81
after GSM8K	44	34	0.31

The instruct model starts 62 explanations behind, which is the cost of its instruction tune. After the first narrow stage, the gap is gone. At every later stage, the poles that keep on one model only split almost evenly, and no test comes near significance. So the instruction tune neither protects explanations through the sequence nor makes them more fragile. It spends its share before the sequence starts, and the base model spends about the same share at its first stage.

The two first costs fall on the same poles more often than chance. The first stage costs the base model 66 explanations, and the instruction tune costs 62. 28 poles are in both sets, against 11.3 expected if the sets were independent (hypergeometric $p = 3.8 \times 10^{-8}$). This suggests a set of fragile explanations that the first update of any kind breaks, whether it is a general instruction tune or 500 steps of sentiment labels. The instruct model's own first narrow stage costs only a net 9 (36 lost, 27 gained, p = 0.31), which agrees with that reading: its fragile explanations had already failed.

6 Do failed explanations stay failed?

A keep test at one stage gives a snapshot. The question here is whether an explanation that fails stays failed. Figure 2 steps through each model's stages, one pole per mark.

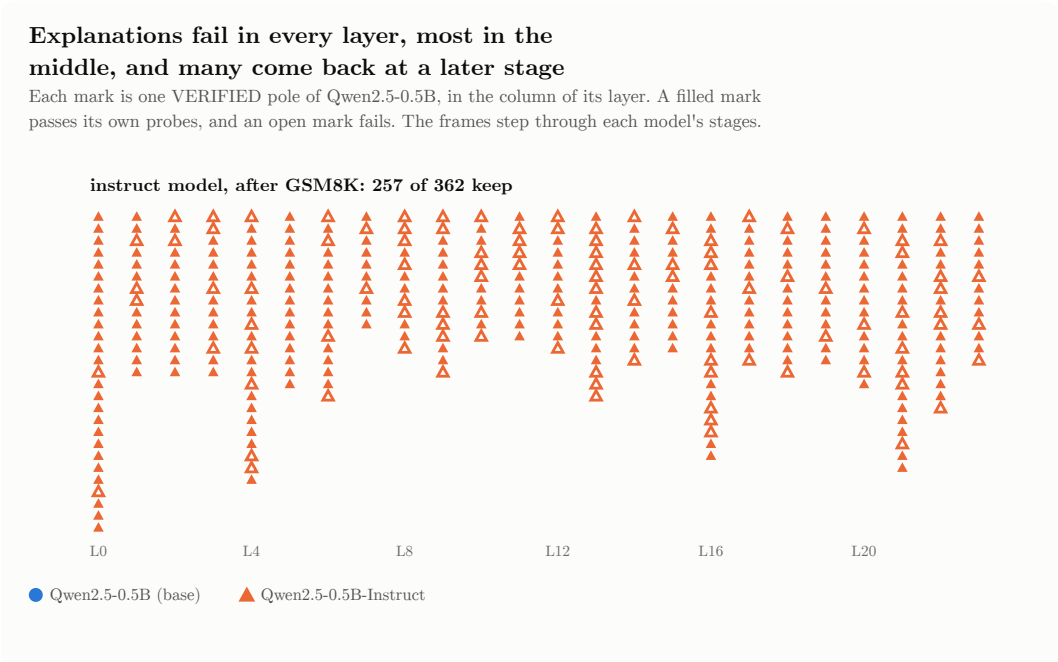

Figure 2. Every reproduced pole, in the column of its layer, at each stage of each model in turn. A filled mark passes its own probes, and an open mark fails. Circles are the base model and triangles the instruct model. The animation rests on the instruct model after the last stage. Each frame is a measured stage.

The losses churn. On the base model, 136 explanations fail at some stage, but only 95 fail at the end, because 50 of those that fail come back at a later stage. Some fail again after that. On the instruct model, 151 fail at some stage, 105 at the end, and 70 come back. Table 1 shows the churn within each stage. Apart from the base model's first stage, every stage both loses and gains explanations, from 18 to 29 gains a stage.

The first failures concentrate early. On the base model, 66 of the 136 poles that ever fail do so first at SST-2, 48 at XSum and 22 at GSM8K. On the instruct model, 62 fail before any narrow stage, then 36, 33 and 20. So the first update finds the most fragile explanations, and each later update finds fewer new ones.

A pole that comes back was, at most, near a gate. The gates are thresholds on shares of 10 to 30 probes, so one probe that crosses the activation line can flip a pole. The churn

therefore says that many explanations sit near their gates after the first update, not that the later stages restore them. Without a second seed, this paper cannot say how much of the churn is noise of this kind (#6367).

7 Does the benchmark score track the explanations?

A general benchmark is the usual measure of forgetting. The question is whether it moves with the explanations. Figure 1 shows ARC-Easy beside the keep share.

It does not. On the base model, ARC-Easy accuracy falls from 0.578 to 0.561 at SST-2, then rises to 0.596 and 0.614, so it ends 3.5 points above its start. Over the same stages, the base model loses a quarter of its explanations. On the instruct model, ARC-Easy falls from 0.593 to 0.499 at SST-2, 9.4 points, while the explanations fall by a net 9 of 362. Then accuracy comes back to 0.595 at XSum and ends at 0.580. So the largest benchmark fall coincides with the smallest loss of explanations, and the largest loss of explanations coincides with a benchmark that hardly moves.

The instruct model's fall and recovery fit the account of Zheng et al. (2025). Its SST-2 stage starts from a held-out SST-2 loss of 8.78 nats, far above the base model's 2.31, because the instruct model expects a different answer form. A stage that teaches a new answer form can disturb the zero-shot answers of ARC, and the next stage, with a longer free-text response, can undo it. We did not test this account. The point that this paper's data support is narrower: a benchmark score and the keep share of explanations can move in opposite directions, so neither stands in for the other.

8 Does the stage's KL predict the explanations it costs?

Shenfeld et al. (2026) find that the KL divergence on the new task's inputs predicts forgetting. The question is whether it predicts the explanations that a stage costs. Figure 4 plots each stage's KL against its net loss of explanations, and Figure 3 gives the task losses that bear on it.

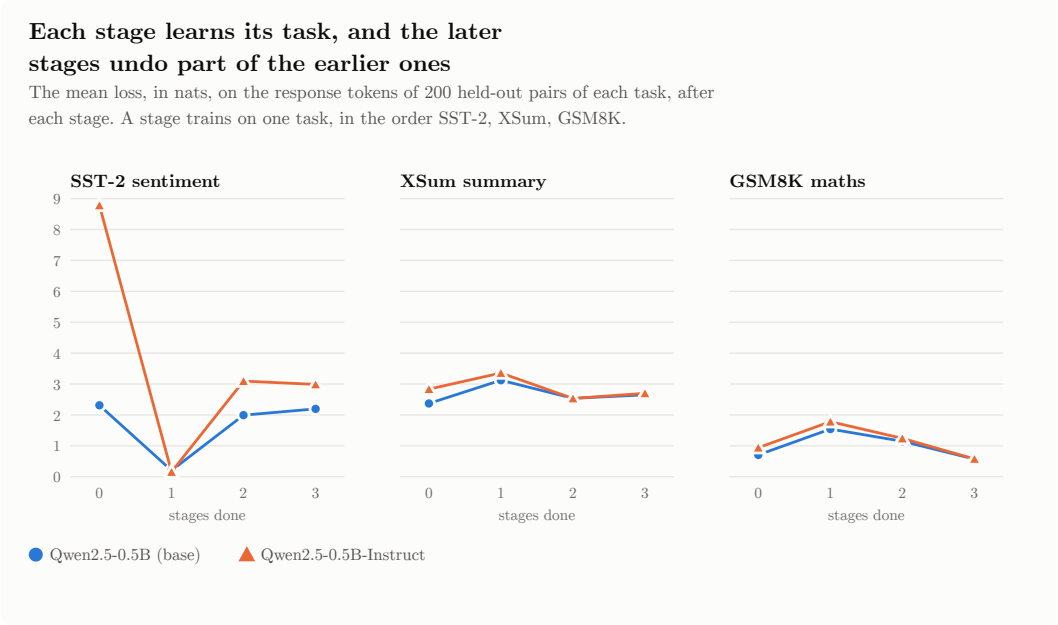

Figure 3. The mean loss on the response tokens of 200 held-out pairs of each task, after each stage. Each task's loss falls at its own stage, and the SST-2 loss rises again at the later stages.

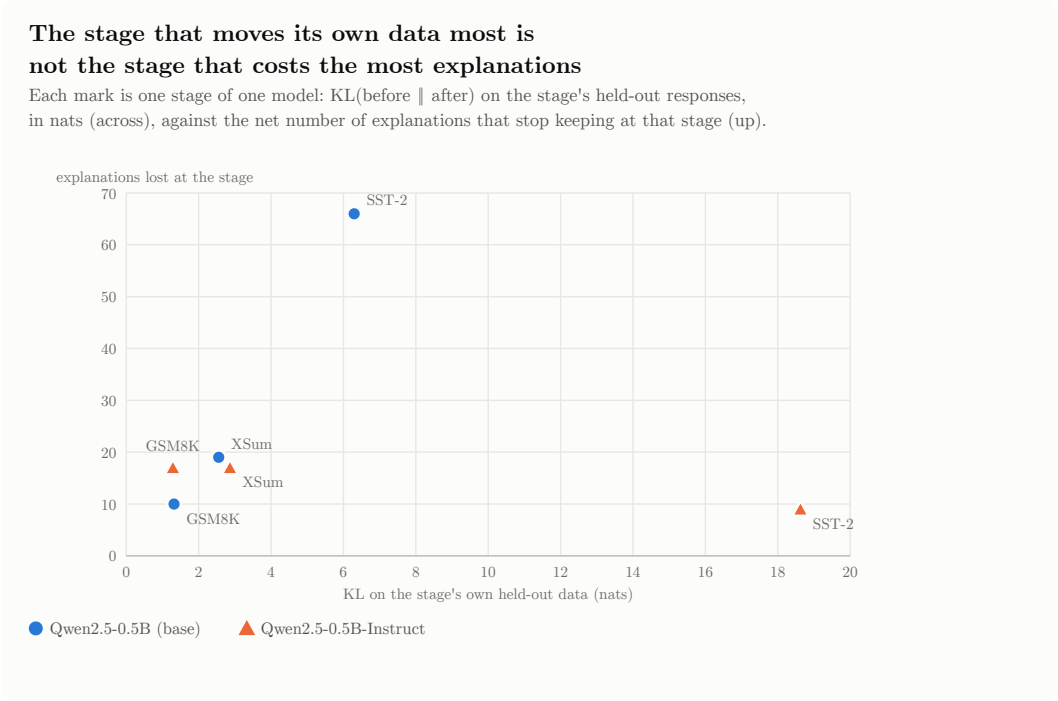

Figure 4. Each stage of each model: the KL divergence from the model before the stage to the model after it, on the stage's own held-out responses, against the net number of explanations lost at that stage (lost minus gained).

The KL does not order the stages by their cost. The instruct model's SST-2 stage has by far the largest KL, 18.6 nats, and costs a net 9 explanations. The base model's SST-2 stage has a KL of 6.3 and costs 66. The other four stages have KLs from 1.3 to 2.9 and cost a net 10 to 19. With six stages, two models and one seed, this is not a test of the law of Shenfeld et al. (2026), which concerns forgetting of earlier tasks, not internal units. It does show that the KL on a stage's own data measures how far the outputs move on that data. The instruct model's large KL comes from a new answer form. It does not measure how far the explained units move.

9 Where in the model do explanations fail?

The question is whether some depths lose more explanations than others. Figure 2 shows the poles by layer, and Table 3 gives the keeps after the last stage, by thirds of the layers.

Table 3. The explanations that keep after the last stage, by thirds of the 24 layers.

Layers	Poles	Base keeps	Instruct keeps
0–7	133	115 (86%)	109 (82%)
8–15	101	57 (56%)	56 (55%)
16–23	128	95 (74%)	92 (72%)

The middle layers lose the most. After three stages, about 55% of the explanations in layers 8 to 15 keep, against 82–86% in the first third and 72–74% in the last. Both models show the same order, at almost the same shares. One reading is that early-layer explanations are more often about single tokens, whose inputs a short fine-tune barely changes. Another is that the updates are larger in the middle layers, as Laitinen-Fredriksson Lundstrom-Imanov (2026) reports for continual fine-tuning. The first reading also differs from Ramasesh et al. (2021), who found that the layers nearest the output change most. This run measured neither the inputs nor the updates, so it cannot choose between them.

10 Discussion

Three findings bear on how an explanation registry should treat fine-tuned models. First, an explanation of a base model carries to its fine-tuned descendants at a rate that falls with each stage, to about 70–75% after three narrow fine-tunes here. That is high enough for a registry to be useful on such a model, and low enough that every explanation needs a re-test. Second, the cost is front-loaded. The first update costs the most, whether it is a broad instruction tune or a narrow one, and the explanations that it breaks overlap. A registry could therefore re-test after one update to find most of its fragile explanations. Third, a pass or a fail at one stage is a noisy reading for many poles near their gates. A registry that marks an explanation as broken after one failed re-test will mark some that pass again at the next stage.

The results also bear on how forgetting is measured. Luo et al. (2023) measured forgetting with benchmarks, and found that it grows along a sequence. Our explanations agree that losses grow along the sequence, but our benchmark does not: ARC-Easy ends higher on the base model. The disagreement fits the split of Zheng et al. (2025) between format and knowledge, and the wrapper account of Jain et al. (2024). A benchmark reads the model's answers through its current answer format, whereas a keep test reads a unit's activations directly. The two can move apart in either direction.

Other readings remain. The keep test reads the probes on plain text, so it does not measure how the explanations hold inside the instruction template that the stages train on. The template paper found that the template alone costs the base model explanations, so a keep test inside the template would likely show more failures. The scale ratio is fixed from one corpus sample per stage. A pole whose maximum moves on rare tokens could pass or fail on the ratio alone. Finally, the stages are short: 2,000 pairs each, a fiftieth of the 100,000 samples that Luo et al. (2023) used, and about a seventy-fifth of the samples seen in training.

11 Limitations

- **One seed and one order.** Every number comes from one run. The churn shows that many poles sit near a gate, so a second seed could move the per-stage counts. #6367 runs seeds and the reverse order.
- **No control stage.** A stage of plain corpus text with the same steps would show whether the first-stage loss comes from narrow instruction data or from any update (#6367).

- **One size and one family.** Qwen2.5-0.5B only. The base-instruct analysis found a higher keep share on Qwen2.5-1.5B.
- **Short stages.** 2,000 pairs for two epochs. Longer stages could cost more.
- **One benchmark.** ARC-Easy, zero-shot, with a per-token mean of log-probabilities. This differs from lm-evaluation-harness's normalised accuracy, which divides by characters, so our ARC numbers are not comparable with theirs.
- **Plain-text probes.** The keep test does not read the probes inside the instruction template.
- **KL on references.** The stage KL is read on the dataset's responses, not on samples from either model.

12 Conclusion

A sequence of narrow fine-tunes costs a growing share of neuron explanations. After three stages, about three quarters keep on Qwen2.5-0.5B and its instruct model. The first update costs the most, and a broad instruction tune and a narrow sentiment tune break an overlapping set of explanations. After the first stage, instruction tuning makes no difference to which explanations keep. Many failures are not permanent. Neither ARC-Easy accuracy nor the KL on a stage's own data tracks the explanations that a stage costs, so an explanation registry has to re-test its explanations on each fine-tuned model, not infer their state from its benchmarks.

References

Biderman, D., Portes, J., Gonzalez Ortiz, J. J., Paul, M., Greengard, P., Jennings, C., King, D., Havens, S., Chiley, V., Frankle, J., Blakeney, C., & Cunningham, J. P. (2024). *LoRA learns less and forgets less*. Transactions on Machine Learning Research. arXiv:2405.09673.

Chen, H., Zhu, J., & Wang, W. (2026). *Unifying and beyond the theory of pervasive capability degradation in LLM post-training SFT from feature interference*. Preprint. arXiv:2611.05003.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). *Think you have solved question answering? Try ARC, the AI2 reasoning challenge*. Preprint. arXiv:1803.05457.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., & Schulman, J. (2021). *Training verifiers to solve math word problems*. Preprint. arXiv:2110.14168.

Jain, S., Kirk, R., Lubana, E. S., Dick, R. P., Tanaka, H., Grefenstette, E., Rocktäschel, T., & Krueger, D. (2024). *Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks*. ICLR 2024. arXiv:2311.12786.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). *Overcoming catastrophic forgetting in neural networks*. Proceedings of the National Academy of Sciences, 114(13), 3521–3526. doi:10.1073/pnas.1611835114. arXiv:1612.00796.

Kotha, S., Springer, J. M., & Raghunathan, A. (2024). *Understanding catastrophic forgetting in language models via implicit inference*. ICLR 2024. arXiv:2309.10105.

Laitinen-Fredriksson Lundstrom-Imanov, G. O. Y. (2026). *Mechanistic analysis of catastrophic forgetting in large language models during continual fine-tuning*. Preprint. arXiv:2601.18699.

Loshchilov, I., & Hutter, F. (2019). *Decoupled weight decay regularization*. ICLR 2019. arXiv:1711.05101.

Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., & Zhang, Y. (2023). *An empirical study of catastrophic forgetting in large language models during continual fine-tuning*. Preprint. arXiv:2308.08747.

Narayan, S., Cohen, S. B., & Lapata, M. (2018). *Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization*. EMNLP 2018. arXiv:1808.08745.

Ramasesh, V. V., Dyer, E., & Raghu, M. (2021). *Anatomy of catastrophic forgetting: Hidden representations and task semantics*. ICLR 2021. arXiv:2007.07400.

Shenfeld, I., Pari, J., & Agrawal, P. (2026). *RL's razor: Why online reinforcement learning forgets less*. ICLR 2026. arXiv:2509.04259.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). *Recursive deep models for semantic compositionality over a sentiment treebank*. EMNLP 2013. aclanthology.org/D13-1170.

von Oswald, J., Henning, C., Grewe, B. F., & Sacramento, J. (2020). *Continual learning with hypernetworks*. ICLR 2020. arXiv:1906.00695.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., . . . Qiu, Z. (2024). *Qwen2.5 technical report*. Preprint. arXiv:2412.15115.

Zheng, J., Cai, X., Qiu, S., & Ma, Q. (2025). *Spurious forgetting in continual learning of language models*. ICLR 2025. arXiv:2501.13453.

Appendices

A. Reproduce

The run needs a CUDA GPU with 12 GB of memory, and took about 4 hours on an RTX 4070 Ti shared with other jobs. The seed is 0 for the samples and the training order. Stage s shuffles its pairs with seed s.

```
uv run python scripts/sequential_tuning.py --device cuda \
    --out papers/sequential-tuning/results.jsonl
uv run python scripts/sequential_tuning.py --analyse \
     --out papers/sequential-tuning/results.jsonl   # the paired tests only, from the
pole rows
uv run python scripts/paper_figures.py
```

The datasets are read from the Hugging Face Hub at these commits:

Dataset	Repository	Commit	Licence on its card
SST-2	`stanfordnlp/sst2`	`8d51e7e4887a4caaa95b3fbebbf53c0490b58bbb`	unknown
XSum	`EdinburghNLP/xsum`	`7d4d486c2f8ef850b1a11aead99b894ff3dd7da9`	unknown
GSM8K	`openai/gsm8k` (main)	`740312add88f781978c0658806c59bc2815b9866`	MIT
ARC-Easy	`allenai/ai2_arc`	`210d026faf9955653af8916fad021475a3f00453`	CC BY-SA 4.0

The SST-2 and XSum cards name no licence. The run reads them for measurement only, and the repository stores no text from them.

B. Each stage

Table B1. Each stage of each model: the keep share with its 95% interval, the median Pearson r of the poles' activations with the base model's, ARC-Easy accuracy with its interval, and the stage KL.

Model	Stage	Keep	Keep 95% CI	Median r	ARC-Easy	ARC 95% CI	KL (nats)
base	start	1.000	0.990–1.000	1.000	0.578	0.558–0.598	–
base	SST-2	0.818	0.775–0.854	0.935	0.561	0.541–0.581	6.30
base	XSum	0.765	0.719–0.806	0.897	0.596	0.576–0.616	2.55
base	GSM8K	0.738	0.690–0.780	0.879	0.614	0.594–0.633	1.32
instruct	start	0.829	0.787–0.864	0.975	0.593	0.573–0.613	–
instruct	SST-2	0.804	0.760–0.842	0.920	0.499	0.479–0.519	18.63
instruct	XSum	0.757	0.710–0.798	0.885	0.595	0.575–0.615	2.87
instruct	GSM8K	0.710	0.661–0.754	0.863	0.580	0.560–0.600	1.29

C. Training

Table C1. The training loss of the first and the last step of each stage, in nats per response token, and the held-out loss of each task after the stage.

Model	Stage	First step	Last step	SST-2	XSum	GSM8K
base	start	–	–	2.31	2.37	0.71
base	SST-2	2.04	0.22	0.20	3.12	1.54
base	XSum	3.06	1.15	2.00	2.54	1.15
base	GSM8K	1.10	0.27	2.19	2.65	0.57
instruct	start	–	–	8.78	2.84	0.94
instruct	SST-2	8.72	0.07	0.16	3.36	1.78
instruct	XSum	3.29	1.17	3.10	2.54	1.24
instruct	GSM8K	1.18	0.26	2.99	2.70	0.58