Is it the template or the tuning? Why neuron explanations fail inside a chat model's prompt format

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

Abstract

When the probes of a verified neuron explanation are read by an instruct model inside its chat template, fewer explanations pass than on plain text. An earlier analysis could not say whether the template or the tuning caused the fall, because it read only the instruct model inside the template. We read the probes of all 361 VERIFIED neuron poles of Qwen2.5-0.5B in four conditions: the base and the instruct model, each on plain text and inside the same chat template. 360 poles pass on the base model with plain text. The template alone costs the base model 76 of them, and the tuning alone costs 61 on plain text. Inside the template, the tuning costs only a net 19 (41 lost, 22 gained), and its effect on mean sensitivity is near zero (-0.003, against -0.024 on plain text). So instruction tuning does not move these explanations more on chat-formatted text. The template hurts the base model more than the instruct model, which suggests that the tuning adapted the model to its own format. Activations tell a slightly different story: the base-to-instruct correlation is lower on chat text for 343 of 360 poles, although by a median of only 0.011. The fall in the keep share inside the template is therefore mostly a format effect that the base model shows too.

Status: Preprint. Work left: #6285 (a second size, matched contexts and a threshold set inside the template).

1 Introduction

An explanation of a neuron is useful only where it holds. Most deployed language models are instruct models, tuned from a base model and run inside a chat template: a fixed frame of system and role tokens around every user turn. If the explanations of a base model's neurons are to serve its instruct model, they must hold for text inside that frame.

An earlier analysis in this registry tested this directly (`docs/analyses/base-instruct.md`). It took each VERIFIED neuron pole of Qwen2.5-0.5B, re-ran the pole's own adversarial probes on Qwen2.5-0.5B-Instruct, and recorded whether the explanation still passes. With plain probes, 252 of 311 passed (81%). With the probes inside the instruct model's chat template, 225 of 312 passed (72%). The analysis read the base model only on plain text, so the fall mixes two causes. The template text could change how the neurons respond, in any model. Or the tuning could have changed the neurons most on inputs like its training data, which are chat-formatted.

The second cause is what theory predicts. Kumar et al. (2022) prove that, in a two-layer linear model, fine-tuning leaves the feature directions orthogonal to the span of the tuning data unchanged. Instruction tuning presumably trains on chat-formatted text, so our extension of their result to a transformer's neurons predicts that the neurons move most inside the template. We call the first cause a *format effect*: a loss that comes from the form of the input, not from a loss of what the model computes. It is close to the split of Zheng et al. (2025) between a model's alignment to a task's format and its knowledge. This paper separates the two with the missing control: the base model read inside the same template.

The paper's contributions are these:

- A reading of every VERIFIED pole of Qwen2.5-0.5B on its own probes in four conditions, base and instruct, plain and inside one shared chat template, with paired tests of the three changes between them.
- Evidence that the template, not the tuning, causes most of the fall inside the template. The template costs the base model 76 explanations, and the tuning costs only a net 19 inside the template.
- Evidence that the tuning moves the probe readings further inside the template, but in both directions, so that it undoes about as much as it breaks.
- Each pole's base-to-instruct activation correlation on a chat-formatted sample beside a plain one, and its mean activation on the template's own tokens in both models.

The paper bears on research question 4, *Do explanations carry across models?* (research questions).

2 Related work

Where fine-tuning moves features. Kumar et al. (2022) compare fine-tuning with linear probing under a distribution shift. Their theory, for a two-layer linear network with a random or zero head, shows that the gradient to the features is zero in every direction orthogonal to the span of the training data. So the head and the features of other inputs fall out of step. Their review also records a case where the out-of-distribution features moved more

than the in-distribution ones. The registry's earlier analysis read correlations on plain web text, which is nearer to the theory's out-of-distribution inputs than chat text is. Their prediction for this paper is that the tuning moves the poles more inside the template. The theory concerns a linear model and a random head, so the step to the neurons of a pretrained transformer is ours.

Format and knowledge. Zheng et al. (2025) split a language model's performance 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 first while the second stays, and that retraining on part of the old task brings the performance back. Their alignment is lost through training, whereas the template effect here appears in a base model that was never trained on the template. So we borrow their split by analogy, at the grain of one neuron.

Chat-specific features. Minder et al. (2025) compare Gemma 2 2B with its chat model through a crosscoder, a sparse dictionary over both models' activations. They find that 40% of the latents specific to the chat model in their BatchTopK crosscoder fire mostly on the template's tokens. Patching in the chat model's activations on the template tokens alone brings the base model's outputs as close to the chat model's as patching in the top half of the chat-specific latents. Their units are dictionary latents, chosen because they differ between the models. Ours are base-model neurons with tested explanations, so we read the template tokens as a third measure, not as the paper's question.

Which inputs to measure on. Shenfeld et al. (2026) find, in a toy setting, that the forgetting of a fine-tuned model is predicted best by the KL divergence between the old and new models on the new task's own inputs. The fit is weaker on language models. For instruction tuning, those inputs are chat-formatted. A companion paper found that a neuron's Fisher importance on plain text does not predict whether its explanation keeps (Fisher keep). This paper asks the prior question: whether the tuning's effect on the explanations differs between plain and chat inputs at all.

3 Method

Models and units. The base model is Qwen2.5-0.5B and the tuned model is Qwen2.5-0.5B-Instruct (Yang et al., 2024). Both have 24 layers (Yang et al., 2024) of 4,864 MLP neurons, the intermediate size in the released model configurations, 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 counts as firing when the pole's activation reaches 0.25 of its maximum on the base model. 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 2026-11-22, Qwen2.5-0.5B had 361 VERIFIED poles.

Four conditions. Each pole's own probes are read in four conditions:

- *base, plain*: the base model reads each probe as plain text, as in the review.
- *base, chat*: the base model reads each probe as one user turn of the chat template.
- *instruct, plain*: the instruct model reads each probe as plain text.
- *instruct, chat*: the instruct model reads each probe inside the template.

Both models read one template, the instruct model's own, rendered with no generation prompt. The two tokenizers' released templates differ in their default system prompt, so a shared template keeps the text the same. Inside the template, the pole is read only on the tokens that overlap the probe's own characters, so a template token cannot fire it. The instruct model's activation scale is the base scale times the ratio of the pole's maxima in the two models on a plain sample, as in the earlier analysis. A pole *keeps* in a condition when it passes both gates there.

Three changes. For each pole and each measure (keep, specificity, sensitivity), three paired changes follow:

$$\Delta_{\text{template}} = m_{\text{base, chat}} - m_{\text{base, plain}}, \quad \Delta_{\text{tuning, plain}} = m_{\text{instruct, plain}} - m_{\text{base, plain}}, \quad \Delta_{\text{tuning, chat}} = m_{\text{instruct, chat}} - m_{\text{base, chat}},$$

where m is the measure in a condition. $\Delta_{\text{tuning, chat}}$ is the tuning's effect with the template's own effect removed. The issue's question is whether $\Delta_{\text{tuning, chat}}$ is larger than $\Delta_{\text{tuning, plain}}$. Their difference is the interaction of the template and the tuning in the two-by-two design.

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. For the rates, the test is an exact two-sided sign test on the poles whose rate rises or falls, and we give the mean change beside it. The rates are shares of 10 to 30 probes, so many changes are ties or a few equal steps, and a sign test handles ties more simply than a signed-rank test. Only the 360 poles that pass on the base model with plain text count, because a pole that fails there has no explanation to lose.

Activations on text. Each pole's activations are also read on two corpus samples, about 20,000 tokens each, from the start of the registry's scan corpus. The *plain* sample is the corpus as one stream of tokens in windows of 512. The *chat* sample is each document, cut at 2,000 characters, as one user turn of the template. For each pole, Pearson's r between the two models' activations is read over the plain sample, and over the document tokens of the chat sample. On the chat sample, each pole's mean strength on the template's own tokens is read in both models: the activation clamped at zero from below, with the sign flipped for a negative pole.

The animated figure. The order of the four conditions is the argument, so Figure 2 steps through them.

4 Does the template alone cost explanations?

Yes, and it costs more than the tuning does. On the base model, putting the probes inside the template takes the keeps from 360 to 284 (Figure 1). No pole that fails on plain text can gain, so all 76 changes are losses ($p < 10^{-6}$). The fall is in sensitivity, not specificity (Table 1). Mean sensitivity drops by 0.060, and 117 poles lose sensitivity against 34 that gain it. Specificity barely moves, by −0.009, with 68 poles down and 74 up (p = 0.67). Of the 76 poles that fail, 55 fail on sensitivity alone, 20 on specificity alone and 1 on both.

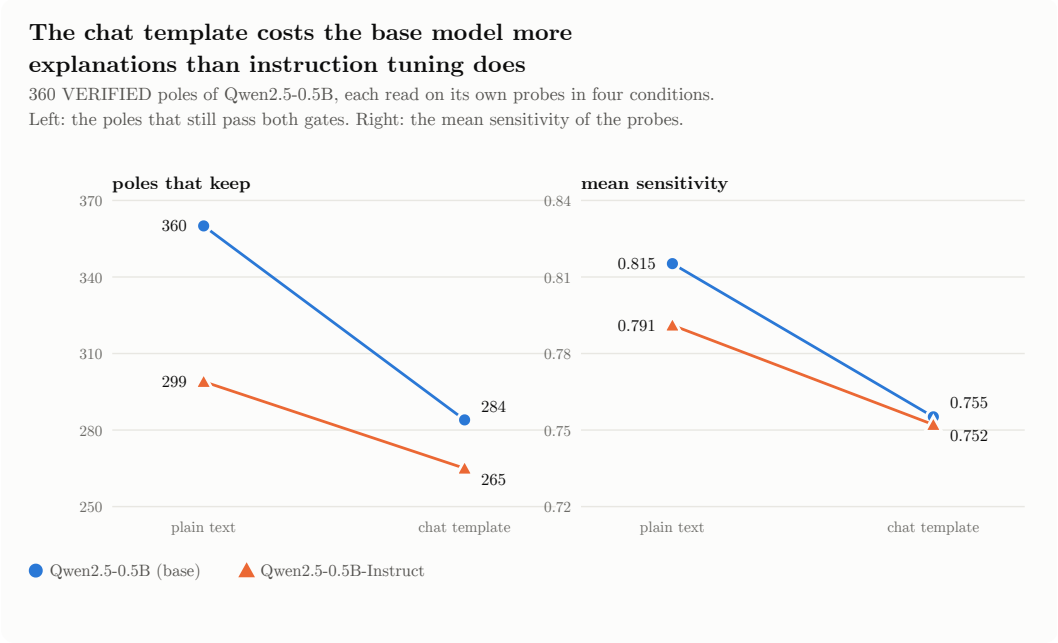

Figure 1. The 360 poles that pass on the base model with plain text, in the four conditions. Circles are the base model and triangles the instruct model. Left: the poles that pass both gates. Right: the mean sensitivity of their probes. The base model's line falls more steeply than the instruct model's.

So the template makes the neurons fire less on the text that their explanations describe, in a model that was never tuned on it. The explanations still say when the poles stay quiet, but the poles fire less often where they should. This is the pattern that the earlier analysis found for the instruct model inside the template, and it is already present in the base model.

Table 1. The three paired changes, over the 360 poles that pass on the base model with plain text. Keep: the poles that lose and gain, with an exact McNemar p. Rates: the poles whose rate falls and rises, with a sign-test p, and the mean change.

Change	Keeps	Lost	Gained	p	Sensitivity: down, up	p	Mean	Specificity: down, up	p	Mean
Template, base model	360 → 284	76	0	$< 10^{-6}$	117, 34	$< 10^{-6}$	−0.060	68, 74	0.67	−0.009
Tuning, plain text	360 → 299	61	0	$< 10^{-6}$	73, 33	0.0001	−0.024	60, 44	0.14	−0.012
Tuning, chat text	284 → 265	41	22	0.023	76, 76	1.0	−0.003	64, 53	0.36	−0.003

5 Does the tuning move explanations more inside the template?

No. On plain text, the tuning costs 61 of the 360 explanations, and gains none. Inside the template, it costs 41 of 284 and gains 22, a net loss of 19 (Table 1). Its effect on mean sensitivity is −0.024 on plain text and −0.003 inside the template, where as many poles rise as fall (76 each). The interaction is in the opposite direction to the prediction. For 59 poles, the tuning does better inside the template than on plain text, and for 19 it does worse ($p = 6 \times 10^{-6}$). Mean sensitivity gives the same sign: the tuning's change is 0.021 higher inside the template (98 poles up, 67 down, p = 0.019).

Part of that interaction is a floor. A pole that the template already cost on the base model cannot lose again under the tuning. But the 22 gains are not a floor effect. They are poles that fail inside the template on the base model and pass inside it on the instruct model. Of the 76 poles that the template cost, 22 come back on the instruct model inside the template.

The same pattern shows when the template is added to the instruct model. There, it costs 45 explanations and gains 11, a net 34, against 76 on the base model. Mean sensitivity falls by 0.039 on the instruct model and by 0.060 on the base model. The template harms the model that was tuned on it less than the model that was not.

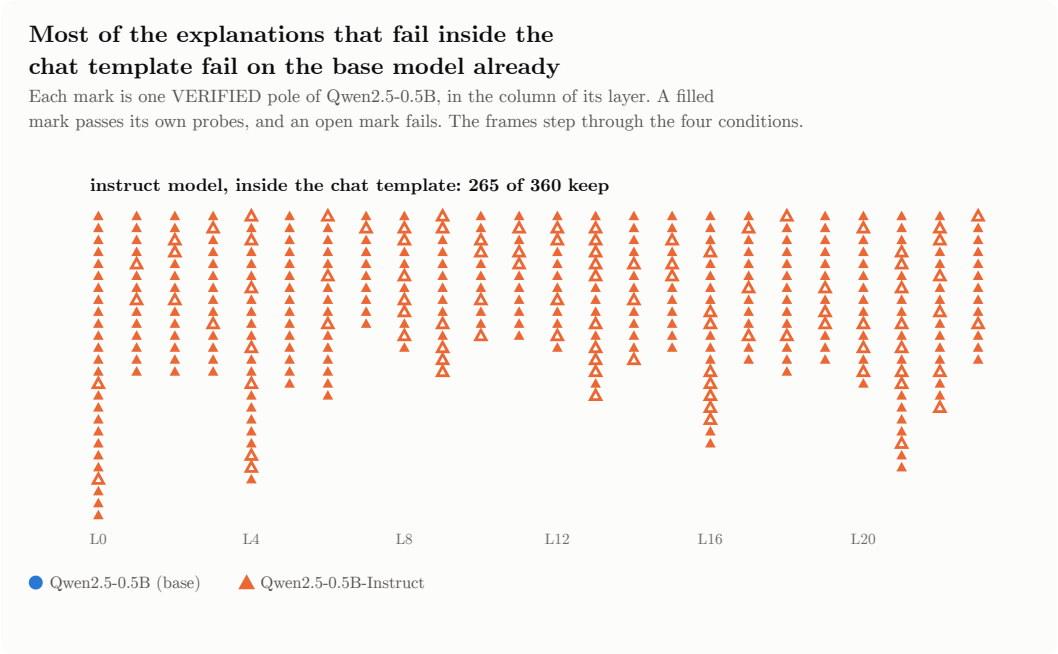

Figure 2. Each mark is one pole, in the column of its layer. A filled mark passes its own probes and an open mark fails. The frames step through the four conditions, and the last frame rests on the instruct model inside the template. Circles are the base model and triangles the instruct model. Most of the open marks of the last frame are already open in the second frame.

The figure shows where the failures come from. Of the 95 poles that fail on the instruct model inside the template, 54 already fail on the base model inside the template. The template's losses spread across all depths. In the first third of the layers it costs 19 of 132, in the middle third 30 of 101, and in the last third 27 of 127. It costs positive and negative poles alike: 35 of 177 positive and 41 of 183 negative.

6 Does the tuning move the readings further inside the template?

It does, but in both directions. A pole's change can be large and still leave it on the same side of a gate, so we compare the size of each pole's tuning change on chat and on plain text. The tuning changes sensitivity by a mean absolute 0.086 inside the template and 0.058 on plain text. For 107 poles the change is larger inside the template, and for 52 it is smaller ($p = 2 \times 10^{-5}$). Specificity shows no clear difference (68 against 52, p = 0.17).

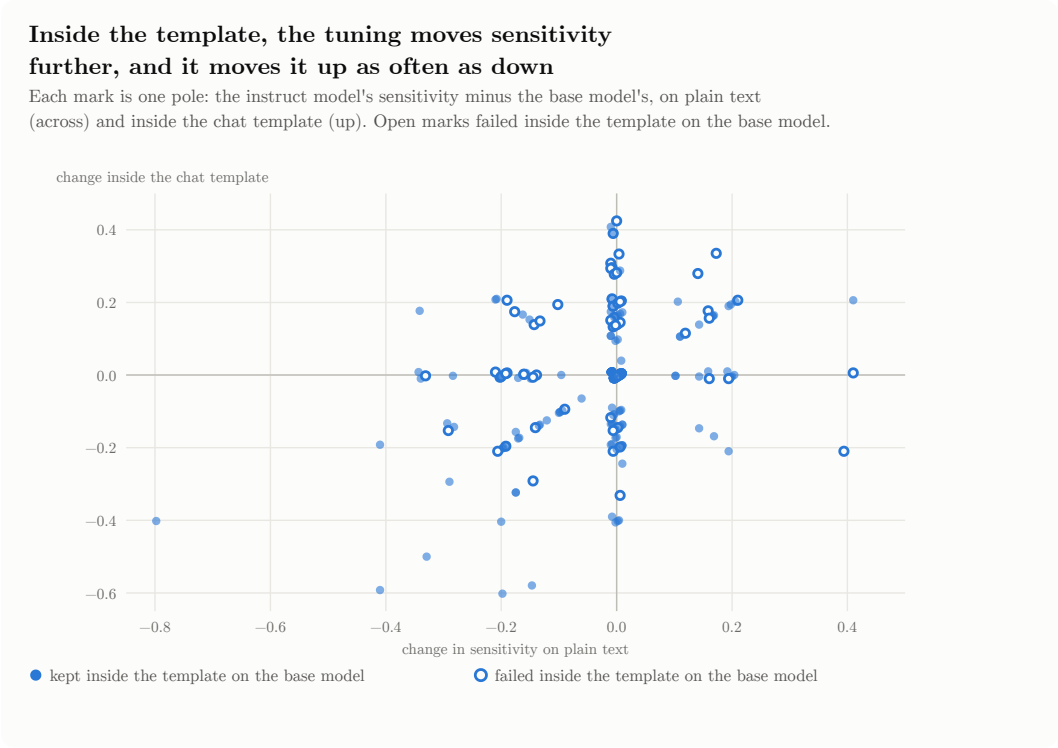

Figure 3. Each mark is one pole. Across: the instruct model's sensitivity minus the base model's, on plain text. Up: the same inside the chat template. Open marks are the poles that failed inside the template on the base model. The marks spread further up and down than across, and many open marks sit above zero.

The changes inside the template go up about as often as down (Figure 3). The poles that the template hurt most on the base model are the ones whose sensitivity the tuning raises most inside it. Across poles, the template's change in sensitivity and the tuning's change inside the template have Spearman $\rho = -0.42$. Part of that relation is arithmetic. The base model's reading inside the template appears in both changes with opposite signs, so any noise in it gives a negative correlation. The 22 poles that come back show that the relation is not only noise.

So the tuning moves the probe readings more inside the template, as our extension of Kumar et al. (2022) predicts, but the net effect is not a loss. It partly undoes what the template does to the base model.

7 Do the activations move more on chat text?

They do, slightly. Across the 360 poles, the base-to-instruct correlation has a median of 0.974 on the plain sample and 0.963 on the chat sample (Figure 4). It is lower on the chat sample for 343 poles and higher for 15, with 2 ties. The median drop is 0.011. The lowest correlation is 0.927 on plain text and 0.872 on chat text. A pole that moves more on one sample moves more on the other: the two values of $1 - r$ have Spearman $\rho = 0.95$.

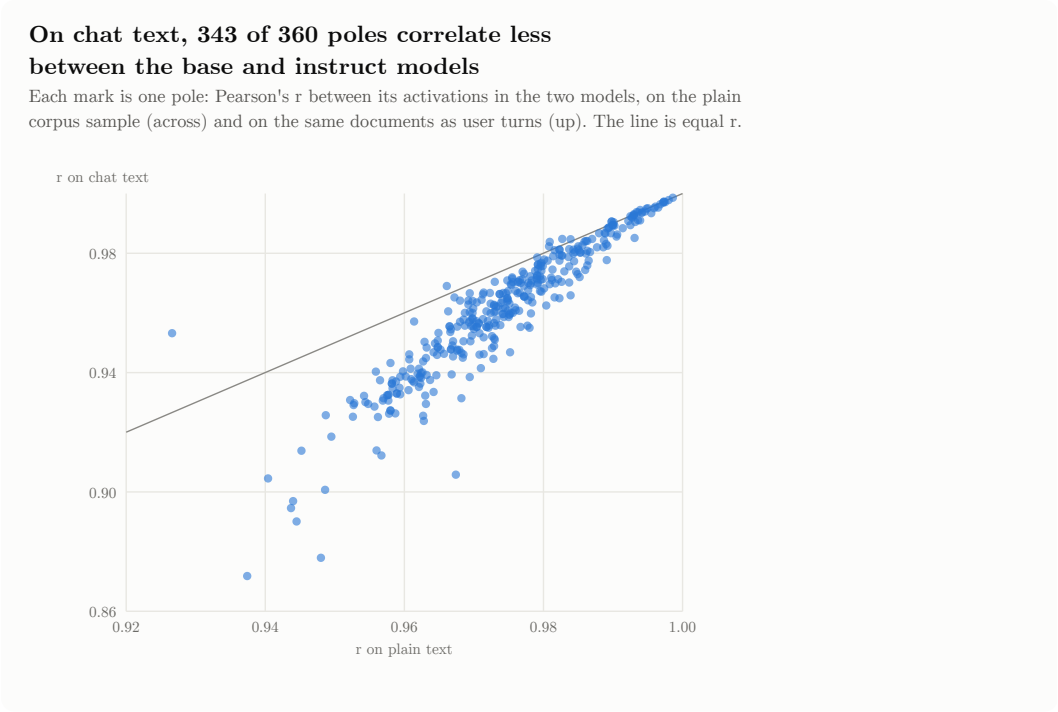

Figure 4. Each mark is one pole: Pearson's r between its activations in the base and instruct models, on the plain corpus sample (across) and on the document tokens of the chat sample (up). The line marks equal correlation. Nearly every mark is below it.

The two samples differ in more than the template. The plain sample reads windows of 512 tokens across document boundaries, and the chat sample reads each document as one turn of up to 2,000 characters. So the drop mixes the template with the length and the starts of the contexts. It is consistent with our extension of Kumar et al.'s result, but it does not test it alone.

On the template's own tokens, the poles fire weakly and about equally in both models. The mean strength there has a median of 0.036 on the base model and 0.040 on the instruct model. It is higher on the instruct model for 199 poles and lower for 161 ($p = 0.051$). A pole's strength on the template does not predict how much the template costs its sensitivity ($\rho = 0.005$). The template tokens that Minder et al. (2025) found to carry chat-specific latents do not engage these base-model poles in either model.

8 Discussion

The question was whether instruction tuning moves VERIFIED explanations more on chat-formatted text than on plain text. The answer for Qwen2.5-0.5B is no. The fall in the keep share inside the template is mostly a format effect. The base model shows it too, and it shows it more strongly than the instruct model. The tuning costs explanations on plain text (61 of 360), whereas inside the template it costs a net 19.

This is analogous to the split of Zheng et al. (2025) between format and knowledge, here a loss that comes from the format beside a smaller loss from what the tuning changed. Kotha et al. (2024) argue that fine-tuning changes which task a model infers from a prompt's form more than its capabilities, and that a prompt reshaped to look less like the tuning data brings old behaviour back. At the level of one neuron, the format effect is a fall in sensitivity. The template frame lowers the poles' activations on the probes' own tokens, so fewer holdouts reach the line at 0.25 of the maximum. The explanations' hard negatives still stay quiet, which suggests that the poles still compute the same thing, at a lower level. That fits the view of Jain et al. (2024) that fine-tuning at a small learning rate mostly adds a small transform on top of unchanged capabilities, although their capabilities are linear readouts, not single neurons. A threshold set inside the template, not on plain text, could recover many of these poles. We did not test that.

The tuning moves the probe readings further inside the template than on plain text, as our extension of Kumar et al. (2022) predicts, and the activations correlate less across the tuning on chat text. But the larger movement is not a larger loss. On the instruct model, the template costs 34 explanations net, against 76 on the base model. One reading is that the tuning adapted the model to its own template, so that text inside the frame looks more like

plain text to the instruct model than to the base model. Another reading is regression: the poles that the template hurt most on the base model have the most room to rise. The 22 poles that come back fit both readings.

For an interpreter, the practical result is that a base-model explanation should be tested inside the template on the base model before the tuning is blamed for a failure. In this sample, more than half of the failures inside the template on the instruct model (54 of 95) are already failures inside the template on the base model. For the registry's earlier analysis, the keep share inside the template measures the format as much as the tuning.

9 Limitations

- **One model pair.** The test uses Qwen2.5-0.5B and its instruct model. The earlier analysis found a larger fall on Qwen2.5-1.5B, from 88% to 70%, on 43 poles. Another size or family could split differently.
- **One template, one turn.** Each probe is one user turn of one template, with the instruct model's default system prompt. Other system prompts, earlier turns and the assistant's side of the template were not tested.
- **The scale comes from plain text.** The activation line is 0.25 of the pole's maximum on plain text. A template that lowers a pole's whole range shows as lost sensitivity here, even if the pole's pattern inside the template is unchanged.
- **The two samples differ in context.** The plain and chat samples differ in window length and in where the contexts start, so the drop in correlation is not the template's effect alone.
- **Probes from the review.** The probes were written to test the explanation on plain text. Probes written for chat use could behave differently.
- **One run.** The readings are deterministic for a given model and device, but the run used a CUDA GPU in float32, whereas the stored records were measured on other devices. One pole of 361 did not pass on the base model with plain text in this run.

10 Conclusion

Instruction tuning does not move the VERIFIED explanations of Qwen2.5-0.5B more on chat-formatted text than on plain text. The chat template itself costs the base model 76 of 360 explanations, mostly through lost sensitivity. Inside the template, the tuning moves the readings further but in both directions, for a net loss of 19 against 61 on plain text. The fall in the keep share inside the template is mostly a format effect.

References

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.

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

Kumar, A., Raghunathan, A., Jones, R., Ma, T., & Liang, P. (2022). *Fine-tuning can distort pretrained features and underperform out-of-distribution.* ICLR 2022. arXiv:2202.10054.

Minder, J., Dumas, C., Juang, C., Chughtai, B., & Nanda, N. (2025). *Overcoming sparsity artifacts in crosscoders to interpret chat-tuning.* NeurIPS 2025. arXiv:2504.02922.

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

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.* 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 script reads the registry's VERIFIED records of Qwen2.5-0.5B from `wiki/`, loads both models in float32, and writes the rows. It uses no random numbers. On a CUDA GPU it took 84 minutes for the 361 poles, on a machine that other jobs also loaded. On a laptop CPU it was on course for about 35 hours. `--limit N` reads only the first N poles, as a timing test.

```
uv run python scripts/template_tuning.py --device cuda \
    --out papers/template-or-tuning/results.jsonl
uv run python scripts/paper_figures.py
```

The rows are in `results.jsonl`. The first row is the `summary`: the keeps in each condition with 95% Wilson intervals. The next three are the `change` rows of Table 1. Each `pole` row gives the four readings of one pole (specificity,

sensitivity, whether it passes and which gate fails), its scale ratio, its correlations on the two samples and its strength on the template's tokens in both models. The code is `scripts/template_tuning.py` and `openinterpretability/interference/template.py`.

B. Keeps by depth and sign

Table 2. The poles that keep in each condition, by third of the layers and by sign.

Group	Poles	Base, plain	Base, chat	Instruct, plain	Instruct, chat
Layers 0–7	132	132	113	124	112
Layers 8–15	101	101	71	70	62
Layers 16–23	127	127	100	105	91
Positive poles	177	177	142	145	136
Negative poles	183	183	142	154	129
All	360	360	284	299	265

The middle third loses the most to both causes, as in the earlier analysis. There the tuning on plain text costs about as much as the template (31 against 30).