CTA TRIPLET: DISENTANGLING CONTEXT, CONSTRAINT AND ALIGNMENT STAGES IN IN-CONTEXT LEARNING WITH SOURCE-SELECTIVE VALIDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning (ICL) enables language models to solve novel tasks from demonstrations without parameter updates. However, a fine-grained functional decomposition of the ICL procedure across network layers remains absent, which is potentially critical for advancing theoretical accounts of ICL. Via combining existing readout and causal intervention approaches, we introduce the CTA triplet, a layer-wise functional coordinate system that decomposes ICL procedure into three independently measurable stages: stage C for context preparation, stage T for task-constraint writing, and stage A for native output alignment, quantified by context-state convergence, query-distribution sensitivity to demonstration perturbations, and the native margin of a fixed candidate pair, respectively. Source-selective interventions and position-preserving permutations verify their distinct functional identities. On Llama3-8B, 60.3–71.4% of classification examples and 70.0% of evaluated token-position trajectories in generation exhibit the following stage procedure: $C \to T \to A$, as measured by peak order. Stage-A scores correlate across layers with recovery of the final output margin after restoring query states from the original prompt (Spearman $\rho = 0.797$–0.938 across six task–corruption conditions). It enables stage-localized ICL failure diagnosis and repair with a maximum 5.4 percentage-point accuracy gain in held-out evaluation, and predicts functional-stage asymmetry in hybrid attention design, validated by depth-dependent performance effects of equal-budget attention replacement. Our CTA triplet establishes a unified testable framework bridging ICL theory, layer-wise readout, causal analysis, and architecture design.

1 INTRODUCTION

In-Context Learning (ICL) enables new tasks through demonstrations without parameter updates (Brown et al., 2020; Dong et al., 2023). Yet a fundamental question remains unresolved: *how does the model's eventual decision form across layers?* Has context stabilization completed, has the demonstration-induced constraint reached the query state, and when does the final winner begin to separate from its eventual runner-up? These questions motivate three perspectives of layer dynamics.

Theoretically, existing ICL theories offer coarse accounts: implicit Bayesian inference (Xie et al., 2022), implicit gradient descent (von Oswald et al., 2023; Akyurek et al., 2023; Dai et al., 2023), contextualize-then-aggregate circuits (Bakalova et al., 2025), and task schema binding (Kim, 2025). These theories link contextual processing to task execution but leave stage locations and separate operational definitions unspecified. A shared set of layer-wise scores would enable empirical comparisons of these theories and their predictions about stages.

In practice, ICL failures are often diagnosed through trial-and-error prompt engineering. When accuracy drops, practitioners cannot determine whether the cause is unstable context, failure to infer the task from demonstrations, or weak preference for the eventual output label, each of which calls for a different intervention. Layer-wise scores may help prioritize candidate repairs. Comparing combined scores with single-score baselines tests whether combining them provides useful information.

Technically, two dominant mechanistic toolkits exist but remain disconnected. Readout-centric methods—logit lens (nostalgebraist, 2020), tuned lens (Belrose et al., 2023), and linear probes (Alain and Bengio, 2017; Hewitt and Manning, 2019)—track where a linear readout can predict labels from hidden states. Intervention-centric methods—activation patching (Meng et al., 2022; Zhang and Nanda, 2023), path patching (Goldowsky-Dill et al., 2023), causal tracing (Wang et al., 2023),

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.

and causal scrubbing (Chan et al., 2022; Conmy et al., 2023)—localize which layers matter causally. Relating layer-wise scores to the layers where replacing hidden states most strongly restores the output requires an explicit operational framework. Here, we present the CTA triplet to bridge this gap by inserting an explicit, measurable task-constraint stage between contextual preparation and output alignment.

The CTA triplet describes ICL as three stages: stage C for context preparation, stage T for the influence of demonstrations on query predictions (task-constraint writing), and stage A for preference for the final winning label (final-decision alignment), each represented by a layer-wise score. These coordinates track overlapping functional properties across layers. The proposed sequence concerns the order in which these properties emerge. Representation, perturbation, and intervention tests examine this interpretation. The framework also connects prior theoretical accounts: contextualize-then-aggregate spans the functions measured by stage C and stage T, while implicit gradient descent and task schema binding primarily explain the formation of stage T, and stage A separately captures the subsequent alignment with the model's native output space.

The CTA triplet also provides scores for selecting changes to prompts. Changes in stage C, stage T, and stage A motivate checking context representations, demonstration influence, and output preferences, respectively. These checks prioritize resampling demonstrations (context resampling), adding an instruction to infer the task rule (task-rule review), or adding an instruction to follow the required answer format (answer-format review), respectively. The layer-wise description also motivates testing whether replacing attention in early or late layers has different effects on accuracy.

In summary, our contributions are as follows:

1. We define three distinguishable layer-wise operational coordinates for stage C, stage T, and stage A, providing a testable framework for examining context preparation, task-constraint writing, and output alignment in ICL. Taking layer-wise hidden states as input, the method computes scores for stage C, stage T, and stage A from prompt representations, changes in query predictions after demonstration perturbations, and the probability difference between the final winning label and its runner-up, respectively.
2. We use this framework to organize candidate ICL failure modes and prioritize context resampling, task-rule review, and answer-format review. Evaluation on held-out examples tests repair outcomes and whether combining CTA triplet scores improves on using a single score. After applying repairs guided by the CTA triplet, all six model–task runs obtain non-negative accuracy changes. The largest gain is 5.4 percentage points, from 87.5% to 92.9%.
3. The coordinate framework motivates a testable proposition: layers associated with later output alignment may tolerate attention replacement better than earlier layers. The results support replacement sensitivity that varies across layers, while an advantage over selection based only on layer position remains unestablished. Early-layer replacement substantially degrades task performance, whereas late-layer replacement is less damaging under the tested replacement budget, although it does not uniformly preserve full-attention accuracy.

2 Related Work

Predicting labels from hidden states and testing their effects. Logit-lens and probe analyses study at which layers labels can be predicted from hidden states (nostalgebraist, 2020; Belrose et al., 2023; Alain and Bengio, 2017; Hewitt and Manning, 2019), whereas activation patching, path patching, causal scrubbing, and automated circuit discovery (ACDC) test which states or paths affect an output (Geiger et al., 2021; Meng et al., 2022; Zhang and Nanda, 2023; Chan et al., 2022; Goldowsky-Dill et al., 2023; Conmy et al., 2023; Heimersheim and Nanda, 2024). The CTA triplet combines these questions while preserving their distinct roles: stage A applies the model's final normalization and vocabulary output head to hidden states. Trained probes predict task labels from those states, and patching tests whether replacing them restores the output. Recent work extends lens-based interpretation to recurrent neural networks and state-space models (Paulo et al., 2024), motivating cross-architecture validation of the CTA triplet.

Theoretical accounts of ICL. Accounts based on implicit gradients, low-rank updates, schema binding, and contextualize-then-aggregate circuits explain how demonstrations can alter a frozen model's computation (von Oswald et al., 2023; Akyurek et al., 2023; Dai et al., 2023; Dherin et al., 2025; Kim, 2025; Bakalova et al., 2025). Most provide a coarse transition from contextual processing to task-relevant prediction. The CTA triplet refines this description by separately measuring similarity to final context representations, sensitivity to changes in demonstrations, and preference for the final winning label. Comparing the layers where trained classifiers predict task labels with those where

the native output margin peaks, and testing recovery of that margin after state replacement, provide complementary evidence for this distinction. Layerwise ICL analysis motivates this operational approach (Jiang et al., 2025).

Task vectors and function vectors. Recent work shows that ICL may produce transferable task vectors or function vectors in hidden space (Hendel et al., 2023; Todd et al., 2024). These results motivate stage T, which tracks how changes to demonstrations alter query predictions across layers.

ICL beyond Transformers and alternatives to full attention. ICL studies beyond pure Transformers motivate validation across different model architectures (Wang et al., 2026). Mamba and selective state-space models (Gu and Dao, 2023; Dao and Gu, 2024) offer linear-time alternatives to attention. Overthinking and false-induction-head analyses motivate testing how intermediate computations affect predictions under incorrect demonstrations (Halawi et al., 2023). Gated linear attention and hybrid sparse/linear attention methods (Yang et al., 2024; Du et al., 2025; MiniCPM Team et al., 2026; Meng et al., 2026) motivate testing attention replacement at different layers. The CTA triplet motivates comparing the accuracy, throughput, and memory use obtained after replacing attention at different layers.

3 Method: The CTA Triplet

We first define the layer-wise readout and then introduce the three functional coordinates of the CTA triplet. We subsequently locate the layer intervals in which each coordinate takes relatively high values.

Readout To compare predictions across layers, let $h_q^{(l)} \in \mathbb{R}^d$ be the query state after applying the model's final normalization once, where d is the hidden dimension. Let $\mathcal{Y} = \{y_1, \ldots, y_K\}$ contain K single-token candidate labels. The logits z_l and softmax distribution $p_l(y_k)$ on label set $\mathcal{Y}$ at layer l are calculated by Eq. (1),

$$z_l = W_o h_q^{(l)}, \qquad p_l(y_k) = \frac{\exp(z_{l,k})}{\sum_{i=1}^{K} \exp(z_{l,i})}, \quad k = 1, \ldots, K, \tag{1}$$

where $W_o \in \mathbb{R}^{K \times d}$ contains the candidate-label rows of the model's vocabulary output matrix, and $z_{l,k}$ is the logit for label y_k.

The CTA triplet For each layer l, we define three functional coordinates (c_l, t_l, a_l) through the calculations below. We test how these scores vary together across layers.

Context-preparation score c_l. Stage C tracks how context representations approach their final-layer states. Let h_i^l be the block-output state at position i for $l < L$, and h_i^L the final-layer state after normalization, where L is the final layer. Let $\mathcal{S}$ contain up to eight uniformly spaced prompt positions before the final query readout token, potentially including query input. We measure directional agreement by the mean cosine distance in Eq. (2):

$$c_l = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left(1 - \frac{\langle h_i^l, h_i^L \rangle}{\|h_i^l\|_2 \|h_i^L\|_2 + \epsilon} \right), \tag{2}$$

where $|\mathcal{S}|$ counts sampled positions, $\langle \cdot, \cdot \rangle$ is the inner product, $\| \cdot \|_2$ is the Euclidean norm, and $\epsilon > 0$ prevents division by zero. We set $c_L = 0$. Smaller c_l indicates closer agreement with the final reference.

Task-constraint score t_l. We quantify how demonstrations constrain query predictions by measuring sensitivity to changes in demonstration order, label assignment, and input content. Averaging the Jensen–Shannon (JS) divergences across these three perturbations gives t_l, with larger values indicating stronger sensitivity. Let $\mathcal{D} = ((x_i, v_i))_{i=1}^n$ contain n ordered demonstrations with inputs x_i and labels $v_i \in \mathcal{Y}$, and let q be the fixed query. For input x_i, let $(u_{i1}, \ldots, u_{im_i})$ denote the m_i tokenizer IDs wholly inside its span and eligible for replacement. Let $\mathcal{V}_{\text{reg}}$ contain non-special vocabulary IDs supported by the embedding table. We construct the three perturbed demonstration sequences in Eq. (3):

$$\begin{aligned} \mathcal{D}^1 &= ((x_{n+1-i}, v_{n+1-i}))_{i=1}^n, \\ \mathcal{D}^2 &= ((x_i, \widetilde{v}_i))_{i=1}^n && \text{(replace labels)}, \\ \mathcal{D}^3 &= ((x_i[u_{ij} \leftarrow r_{ij}]_{j \in B_i}, v_i))_{i=1}^n && \text{(replace input tokens)}, \end{aligned} \tag{3}$$

where $\mathcal{D}^1$ reverses demonstration order. The replacement label is $\widetilde{v}_i = y_2$ for $v_i = y_1$, and y_1 otherwise. For token replacement with rate $p \in (0, 1]$, B_i contains $\lceil pm_i \rceil$ uniformly sampled

positions without replacement, and each r_{ij} is drawn independently and uniformly from $\mathcal{V}_{\text{reg}} \setminus \{u_{ij}\}$. We set $p = 0.15$ as a fixed design choice to perturb a limited fraction of each input while retaining most tokens. Token replacement preserves prompt length and leaves labels and the query unchanged. Using Eq. (1) with the fixed query q, let p_l^0 denote the clean distribution and p_l^1, p_l^2, p_l^3 the distributions under $\mathcal{D}^1, \mathcal{D}^2, \mathcal{D}^3$, respectively. We calculate t_l in Eq. (4):

$$D_l^j = \frac{1}{2} \sum_{y \in \mathcal{Y}} \left[p_l^0(y) \log \frac{p_l^0(y)}{m_l^j(y)} + p_l^j(y) \log \frac{p_l^j(y)}{m_l^j(y)} \right], \qquad t_l = \frac{1}{3} \sum_{j=1}^{3} D_l^j, \tag{4}$$

where $m_l^j = (p_l^0 + p_l^j)/2$ is the mixture distribution, D_l^j is the JS divergence for perturbation $j \in \{1, 2, 3\}$, and log is the natural logarithm.

Output-alignment score a_l. We quantify alignment with the eventual decision by the probability margin between the final winning label and its runner-up. A positive a_l favors the final winner, and a larger positive value indicates stronger separation. The pair is selected from the model's own final prediction, independently of the correct label. Let p_L be the candidate-label distribution from Eq. (1) at the final layer L. We select the candidate pair in Eq. (5):

$$y_1^* = \arg\max_{y \in \mathcal{Y}} p_L(y), \qquad y_2^* = \arg\max_{y \in \mathcal{Y} \setminus \{y_1^*\}} p_L(y), \tag{5}$$

where y_1^* is the final winning label and y_2^* is its runner-up. This pair is fixed across layers. Using the layer-l probabilities $p_l(y)$ and logits $z_l(y)$ from Eq. (1), we calculate their margins in Eq. (6):

$$M_l^p = p_l(y_1^*) - p_l(y_2^*), \qquad M_l^z = z_l(y_1^*) - z_l(y_2^*), \tag{6}$$

where M_l^p is the native probability margin, which defines $a_l = M_l^p$, and M_l^z is the native logit margin used as an alternative readout.

Locating stages with activity windows We observe that most CTA triplet curves are monotonic or unimodal. To summarize the region where each curve takes relatively high values, we define an *activity window*. For each coordinate, we first scale individual profiles to $[0, 1]$ and average them across examples. We then select the largest contiguous interval containing the earliest maximum and remaining at or above 80% of that maximum. These windows summarize the locations and overlap of the profiles for stage C, stage T, and stage A. We assess their ordering separately from the ordering of individual peak layers.

4 Experiments

We evaluate the CTA triplet through five research questions (RQs). Reported layer indices are zero-based Transformer blocks. **RQ1: Distinguishing the roles of the three CTA triplet scores.** Do changes to demonstration inputs, labels, and query states affect the three CTA triplet scores differently, in ways consistent with their proposed roles? **RQ2: Stage order and relation to existing ICL theories.** Do the three CTA triplet scores follow the proposed stage procedure $C \to T \to A$ across layers, and how does this pattern relate to existing ICL theories? **RQ3: Effects of restoring hidden states and changing demonstrations.** How does restoring hidden states affect the final output, how does query-state recovery relate to stage-A scores, and how do demonstration changes affect t_l and accuracy? **RQ4: CTA triplet across models, tasks, and architectures.** Do the patterns extend across models, tasks, and architectures, including free generation and non-Transformer backbones? **RQ5: Prompt repair and attention replacement.** Can CTA triplet scores help detect errors and select prompt repairs, do combined scores outperform single-score baselines, and how does the location of attention replacement affect accuracy?

4.1 RQ1: Distinguishing the roles of the three CTA triplet scores

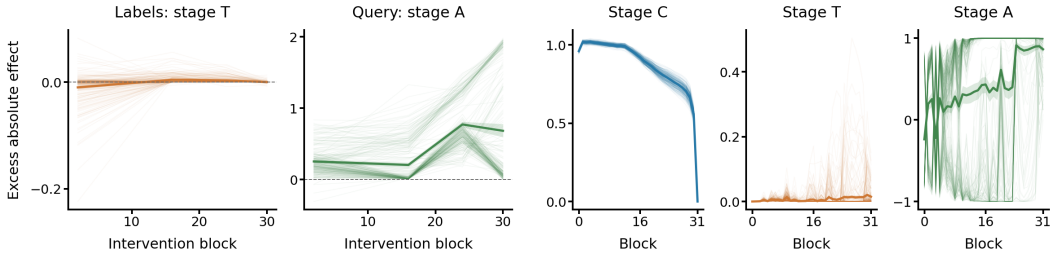
To test whether the three scores respond differently to information in different prompt regions, *Zeroing states in selected prompt regions* compares selected regions with equally many random token states. *Effects of token selection, demonstration changes, and margin choice* tests how the scores depend on these choices. *Choice of stage T perturbations* compares alternative ways to change demonstrations, checking how the selected changes affect t_l.

Zeroing states in selected prompt regions On 866 Qwen3-4B/Llama3-8B model–example pairs, we zero demonstration-input, label, or query states. Controls zero equally many random token states. We compare absolute score changes averaged over later blocks, using 613,128 repeated layer readouts. Query, label, and input interventions exceed controls for stage A, stage T, and stage C in 18/18, 14/18, and 7/18 task-by-layer comparisons across the two models. Benjamini–Hochberg correction

Table 1: Checks of token selection, demonstration changes, and output margins. Cosine-distance comparisons use Qwen3-4B across three tasks.

Coordinate	Check	Result
c_l	Input and label tokens versus equal-size random token sets	Excess mean cosine distance: inputs 0.0136–0.0251, labels 0.0264–0.0325
t_l	Order, label, and input changes	Selected combination ranks 3rd–14th among 27 combinations by mean JS
a_l	Probability and logit margins	Both margins increase mainly in later layers

across 54 comparisons gives adjusted p-values $q \leq 0.05$ for 18, 12, and 4 positive differences. Figure 1a shows paired effects. Matching token count controls intervention size. Token positions, state magnitudes, and the later states they can affect remain possible influences. Label and query interventions affect stage T and stage A more consistently than input interventions affect stage C.



(a) Prompt-region interventions, Llama3-8B RTE (b) CTA triplet trajectories, Llama3-8B RTE

Figure 1: Effects of interventions in different prompt regions and CTA triplet curves across layers. Thin transparent lines denote examples; bold curves summarize the population.

Effects of token selection, demonstration changes, and margin choice Table 1 compares token selection for stage C, demonstration changes for stage T, and probability versus logit margins for stage A. The results show that c_l varies with token selection, t_l varies with the perturbation combination, and both margins increase mainly in later layers.

Choice of stage T perturbations On 120 model–example evaluations, we compare reversal, random permutation, and rotation of demonstration order. Label changes use the first different candidate, the next candidate cyclically, or shuffled labels. Input changes replace, permute, or delete tokens at matched positions. Each of 27 combinations uses one method per category and two seeds. The selected combination ranks 3rd–14th by layer-averaged JS, describing its response magnitude relative to these alternatives.

Appendix A.2 reports shuffling within prompt regions, word versus token replacement, and added-space controls, supporting stage-C content sensitivity while showing that stage T also responds to formatting and perturbation choices.

4.2 RQ2: Stage order and relation to existing ICL theories

To test whether the CTA triplet scores follow the proposed stage procedure $C \to T \to A$, *Peak locations and order* compares the layers where each score peaks. *Activity-window locations and overlap* then examines whether the intervals around these peaks have the same order and how much they overlap. *Relation to existing ICL theories* discusses how the observed patterns relate to earlier accounts of ICL. To test whether task labels can be predicted from hidden states before the model's final winning label is most strongly favored, *Comparing classifier accuracy and output margins* compares the first layer where a trained classifier reaches 80% accuracy with the layer where the model's output margin peaks.

Peak locations and order Table 2 reports three Llama3-8B runs. In the sample-mean curves, stage C peaks at layers 1–3, and stage T follows stage A on SST-2 and RTE. Within each example, let l_C^*, l_T^*, l_A^* denote the three peak layers. Strict order, $l_C^* < l_T^* < l_A^*$, holds for 60.3–71.4% of examples. Qwen3-14B has probability-margin peaks at layers 31/36/38 on SST-2/RTE/CB, showing late output alignment in this larger model. Together with Figure 1b, these results show recurring individual peak order alongside differences in the order of sample-mean peaks.

Activity-window locations and overlap Across the six full runs, windows fitted on training examples have higher mean scores inside than outside on held-out examples for all 18 combinations

Table 2: Llama3-8B accuracy (Acc., percent), mean-curve peak layers, and fraction of examples with strict stage order.

Model	Task	Acc.	c_l	t_l	a_l	Order
Llama3-8B	SST-2	91.0	2	30	26	62.0%
Llama3-8B	RTE	81.2	3	30	24	60.3%
Llama3-8B	CB	91.1	1	23	26	71.4%

Table 3: Activity windows at the 80% threshold, fitted on all examples for description. Bounds are zero-based blocks. Comparisons of scores inside and outside windows on held-out examples use windows fitted on separate training folds.

Model	Task	c_l window	t_l window	a_l window
Llama3-8B	CB	0–23	20	24–31
	RTE	0–21	9	24–31
	SST-2	0–24	30–31	15–31
Qwen3-4B	CB	0–23	34–35	31–35
	RTE	0–23	4–5	31–35
	SST-2	0–26	35	19–35

of score and model–task setting (95% bootstrap intervals for inside-minus-outside differences exclude zero, with fitted windows fixed). However, window centers (midpoints) follow strict stage order in only 23.8–58.9% of examples at the primary threshold. Table 3 shows overlaps and order reversals. Qwen3-4B/RTE's scaled and raw mean t_l profiles peak at blocks 4 and 30, respectively: scaling changes the relative weight of low-amplitude profiles. Stage-T window boundaries also vary by up to 11 blocks across folds. These windows locate elevated scores on held-out examples, with overlap and ordering that vary across settings.

Relation to existing ICL theories Contextualize-then-aggregate describes how contextual information is processed and combined. Implicit gradient descent and task schema binding describe how demonstrations change the model's computation for the query. Stage C and stage T provide scores for comparing representation changes and sensitivity to demonstrations with these accounts. Stage A tracks the model's preference for its final winning label. These comparisons concern the functional properties emphasized by each account.

Comparing classifier accuracy and output margins We train classifiers based on ridge regression to predict task labels from each layer's query states. With five-fold cross-validation, they first reach 80% accuracy at layers 7/13/8 for Llama3-8B on CB/RTE/SST-2. On RTE and SST-2, the stage-A probability margin peaks at layers 24 and 26, respectively, 11 and 18 layers after the classifier reaches the threshold. Thus, the classifiers can predict task labels before the model's preference for its final winner is strongest.

Peak order is common across examples, while windows overlap and can reverse order. Appendix A.3 tests dependence on window thresholds, training splits, and raw versus normalized margins.

4.3 RQ3: Effects of restoring hidden states and changing demonstrations

To test how hidden states at different layers influence the final output, we replace states in disturbed prompts with those from the original prompts. *Restoring query states* tests whether replacing query states at layers with larger stage-A scores produces greater recovery of the final probability margin. *Restoring demonstration-input and label states* tests the effects of replacing demonstration-input and label states. To connect score responses with prediction quality, *Effects of demonstration quality on stage T and accuracy* compares t_l and accuracy after changing demonstration content and count.

Restoring query states We replace query states in label-flipped and no-demonstration Llama3-8B runs with states from the original prompt, measuring recovery of the probability margin for its final winner and runner-up. Across six task–corruption conditions, correlations between stage-A scores and recovery are 0.797–0.938 (Spearman). Shuffling one curve across layers gives $p < 10^{-4}$, and shifting it cyclically gives $p = 0.03125$. Positive intervals from resampling examples support the association between larger stage-A scores and greater margin recovery after query-state restoration.

Restoring demonstration-input and label states To test the effects of restoring demonstration states, we evaluate Llama3-8B and Qwen3-4B on the same 20 stratified examples per task. After demonstration-input zeroing or label corruption, we restore input or label states from the original prompt at a fixed set of layers. Table 4 summarizes differences in the final output margin between targeted restoration and control interventions, averaged over the layers selected in advance. Label restoration more consistently exceeds random-position controls. Restoring input states at early layers can recover accuracy. Their final-margin advantage over random controls is not significantly positive when averaged over the tested layers.

Effects of demonstration quality on stage T and accuracy On Llama3-8B, we compare clean 4-shot, half-label-flipped, irrelevant-input, and clean 8-shot prompts using identical examples and

Table 4: Effects of restoring demonstration states relative to control interventions across six model–task settings. Counts denote significant positive effects after global Benjamini–Hochberg (BH) correction over 24 comparisons. Parenthesized counts use Holm.

Restored states	Versus random positions	Versus shuffled states
Demonstration inputs	0/6 (0/6)	3/6 (1/6)
Demonstration labels	5/6 (3/6)	3/6 (1/6)

Table 5: Demonstration quality and stage-T scores. Each entry reports accuracy followed by the mean per-example t_l peak.

Condition	Llama3-8B/ SST-2	Llama3-8B/ RTE
Clean 4-shot	92.0%, 0.0908	82.3%, 0.0644
Label-flip-half	88.0%, 0.0783	81.2%, 0.0618
Irrelevant input	83.0%, 0.1123	83.0%, 0.0661
Clean 8-shot	90.0%, 0.1106	82.7%, 0.0717

Table 6: Final classification accuracy on the fixed evaluation sets used in this study: SST-2 ($n = 100$), RTE ($n = 277$), and CB ($n = 56$).

Model	SST-2	RTE	CB
Qwen3-14B	94.0%	86.6%	91.1%
Llama3-8B	91.0%	81.2%	91.1%
Gemma2-2B-IT	63.0%	72.2%	85.7%
GPT-2	53.0%	48.4%	51.8%

Table 7: Fraction of free-generation examples satisfying $l_C^* < l_T^* < l_A^*$ at each generated-token position. Each cell contains 20 examples.

Model and task	Token 1	Token 2	Token 4	Token 8
Llama3-8B, XSUM	45%	70%	80%	65%
Llama3-8B, WMT14	65%	70%	90%	75%

sampling seeds within this experiment. These prompts include task instructions, which the primary classification prompts omit. Table 5 reports all conditions. Moving from clean 4-shot to 8-shot prompts raises the mean t_l peak on both tasks, while accuracy does not improve consistently. On SST-2, irrelevant inputs increase t_l while reducing accuracy. Thus, stage T responds to changes in the demonstrations, but larger scores do not consistently indicate higher accuracy.

Appendix A.4 reports random-position and shuffled-state comparisons and controls that reinsert unchanged states or replace non-query states after the final block. These controls test replacement location and state placement, and verify that reinserting unchanged states preserves the output.

4.4 RQ4: CTA TRIPLET ACROSS MODELS, TASKS, AND ARCHITECTURES

To assess whether the observed layer patterns recur beyond the primary classification settings, *Cross-model and free-generation tests* compares output-margin peaks across models and CTA triplet profiles across generated-token positions. The Mamba comparison applies the same token-based perturbations to a state-space model.

Cross-model and free-generation tests Table 6 reports classification accuracy. Native-margin results for Qwen3-14B and Llama3-8B test late output alignment across models. Using task instructions and four training demonstrations, four-beam generation evaluates 20 WMT14 French–English translation and 20 XSum summarization examples with Llama3-8B. At output positions 1, 2, 4, and 8, preceding generated tokens are fixed across original and perturbed prompts, and the original prompt's c_l curve is reused. Here a_l compares the generated token with its strongest final-layer alternative. Of the resulting 160 position-level trajectories from 40 examples, 70.0% satisfy strict peak order. Positions from the same example are repeated measurements. Table 7 gives position-level results and Figure 2 shows WMT14 trajectories. With token perturbations, Mamba-2.8B attains 53.1% on RTE and predicts entailment for 258/277 examples. Mean-profile peaks for c_l, t_l, a_l are at blocks (0,42,46), respectively, while only 3.6% of examples follow strict peak order. Strict individual peak order is common in the tested Llama3-8B generation settings and rare in Mamba/RTE.

Appendix A.5 extends the comparison to multi-token candidate scoring and Qwen3-4B generation.

4.5 RQ5: PROMPT REPAIR AND ATTENTION REPLACEMENT

To test whether CTA triplet scores help select useful prompt changes, *Evaluating prompt repairs selected by CTA triplet thresholds* evaluates those changes on examples excluded from fitting. *Comparing combined CTA triplet scores with single-score baselines* tests their value for error detection and repair. To identify where predictions tolerate changes to attention, *Attention replacement at different layers* compares similarly sized early, middle, and late layer groups.

Evaluating prompt repairs selected by CTA triplet thresholds Unusually high stage-C or low stage-T or stage-A scores prioritize resampling demonstrations, adding task instructions, clarifying the answer format, or retaining the prompt. Thresholds are fitted on four folds. A fixed mapping from scores to these changes selects actions on the remaining fold without using its correct labels.

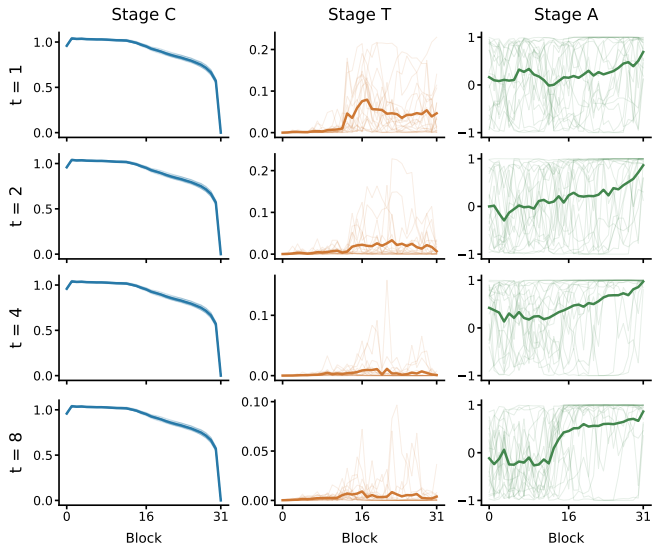


Figure 2: Free-generation CTA triplet trajectories on WMT14 for Llama3-8B. Thin transparent lines denote individual examples and thick lines denote means at generated-token positions 1, 2, 4, and 8.

The reported Llama3-8B repair policy changes SST-2/RTE/CB accuracy from 91.0/81.2/91.1% to 94.0/81.9/91.1%. The observed gains are descriptive (paired McNemar $p \geq 0.25$).

Comparing combined CTA triplet scores with single-score baselines In a separate five-fold comparison (Table 8), classifiers predict errors, and regressions predict each repair's increase in correct-label probability using training examples. Combining CTA triplet scores does not outperform using only the final stage-A score, equivalent to the final output margin.

Table 8: Comparison of combined CTA triplet scores, single scores, and random actions using five-fold evaluation across six full runs. AUROC: area under the receiver operating characteristic curve. Repair gain is in percentage points. Non-decrease rate is the fraction of modified examples whose correct-label probability does not decrease.

Method	Error AUROC	Repair gain (pp)	Non-decrease rate
Combined CTA triplet scores	0.794	-0.23	0.495
Final stage-A score	0.795	-0.10	0.589
Final output margin	0.795	-0.10	0.589
Random repair choice	0.500	$+0.55$	0.393

Attention replacement at different layers We report attention replacement in approximately equal numbers of early, middle, or late layers. Sliding-window attention uses a causal 48-token window; kernel-based causal linear attention uses positive exponential linear unit (ELU) features while retaining frozen projections and positional operations. In the attention-replacement runs, early sliding-window replacement on Llama3-8B reduces SST-2/RTE/CB accuracy from 90.0/80.1/91.1% to 58.0/51.6/44.6%, whereas late replacement gives 90.0/80.5/92.9%. Across six Qwen3-4B and Llama3-8B runs, early kernel-based replacement reaches 47.3–50.0%, while late replacement changes accuracy by -8.9 to $+3.0$ percentage points.

Appendix A.6 specifies both selection rules and reports throughput and memory comparisons against full attention.

5 Discussion

How should we interpret overlapping activity windows? Ordered peaks can coexist with overlapping activity windows. Scores remain higher inside fitted windows than outside on held-out examples (Appendix A.3). Overlap therefore describes layers where several CTA triplet scores are elevated, allowing the three properties to be examined together.

Does a higher stage T score imply better learning? A higher t_l indicates stronger responses to changed demonstrations. On RTE, added spaces produce responses comparable to the selected perturbations. On Llama3-8B/SST-2, irrelevant inputs increase t_l while reducing accuracy (Appendices A.2, A.4). This accuracy decrease shows that stronger sensitivity can reflect distracting information. Thus, t_l describes response strength, while accuracy assesses whether a prompt change helps prediction.

Why can classifiers predict labels before stage A peaks? Trained classifiers predict labels from hidden states earlier than the model's output margin peaks (Appendix A.3). These classifiers learn from task labels, while a_l uses the existing output head. Their earlier predictions reveal label information before the final decision is most strongly favored.

Why do hidden-state replacements have different effects across layers? Replacing demonstration-input states with those from the original prompt can recover accuracy at early layers (Appendix A.4). Later attention operations can pass this information to the query. No such operation remains after the final block, explaining the zero effect of non-query replacements there.

What does the CTA triplet describe during generation? During generation, the CTA triplet describes how sensitivity to demonstrations and preference for the generated token develop across layers, using the original prompt's c_l as a shared context reference. At each output position, t_l tracks sensitivity to demonstration changes, and a_l tracks preference for the generated token over its strongest final-layer alternative. With preceding tokens fixed, their peak locations vary across output positions, while the stage-C, stage-T, stage-A peak order recurs in 45–90% of examples across the tested settings (Appendix A.5). These results extend the layer-wise description to successive token predictions.

What do the results imply for prompt changes and attention replacement? CTA triplet thresholds select among resampling demonstrations, adding task instructions, and clarifying the answer format. These choices yield descriptive accuracy gains on Llama3-8B SST-2 and RTE (Appendix A.6). Later layers also preserve accuracy more consistently after attention replacement. Their greater tolerance motivates testing alternative attention mechanisms in these layers while monitoring accuracy.

6 Limitations

Choice and extent of changes to demonstrations. Stage T depends on how demonstration order, labels, and input tokens are changed. We sample 15% of eligible input tokens, but equal proportions can alter meaning differently across examples and tokenizers. Our 27-combination comparison tests variation across the selected methods at this fixed rate. Varying the rate would further characterize how t_l depends on the amount of input changed.

Window selection. Activity windows use per-example scaling and an 80% threshold. Scaling and threshold choice affect their endpoints, so the reported endpoints are approximate locations of elevated scores. Held-out and threshold checks assess the stability of these locations. Exact layer boundaries are unnecessary for describing the recurring order and overlap of the profiles.

Sample size and generation length. Experiments replacing disturbed hidden states with those from the original prompt use 20 examples per task and model. Comparisons of ways to change demonstration order, labels, and input tokens use the same examples. This sample size limits precision for small differences between conditions. Generation tests positions 1, 2, 4, and 8 with preceding tokens fixed. Longer outputs with freely generated prefixes would additionally capture how earlier output changes affect later predictions.

7 Conclusion

We introduced the CTA triplet to describe context preparation, sensitivity to demonstrations, and alignment with the final decision across layers. Changing demonstrations, zeroing selected or equally many random prompt states, and restoring states from the original prompt connect these profiles to changes in model predictions. The profiles exhibit recurring peak order and overlapping activity windows, while trained classifiers reveal label information before the model's output preference peaks. Generation experiments extend this description to successive token predictions. Prompt-repair and attention-replacement experiments further demonstrate how score-based action choices and intervention layers affect prediction outcomes. Together, these findings provide an operational framework for comparing layer-wise computation and designing targeted interventions in in-context learning.

REFERENCES

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2023.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit Bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning (ICML)*, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models. In *International Conference on Learning Representations (ICLR)*, 2023.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can GPT learn in-context? Language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2023.

Aleksandra Bakalova, Yana Veitsman, Xinting Huang, and Michael Hahn. Contextualize-then-Aggregate: Circuits for In-Context Learning in Gemma-2 2B. *arXiv preprint arXiv:2504.00132*, 2025.

Chaeha Kim. Task Schema and Binding: A Double Dissociation Study of In-Context Learning. *arXiv preprint arXiv:2512.17325*, 2025.

Benoit Dherin, Michael Munn, Hanna Mazzawi, Michael Wunder, and Javier Gonzalvo. Learning without training: The implicit dynamics of in-context learning. *arXiv preprint arXiv:2507.16003*, 2025.

Jiachen Jiang, Yuxin Dong, Jinxin Zhou, and Zhihui Zhu. From Compression to Expression: A Layerwise Analysis of In-Context Learning. *arXiv preprint arXiv:2505.17322*, 2025.

nostalgebraist. Interpreting GPT: The logit lens. *LessWrong*, 2020.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting Latent Predictions from Transformers with the Tuned Lens. *arXiv preprint arXiv:2303.08112*, 2023.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations (ICLR) Workshop*, 2017.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

Gonçalo Paulo, Thomas Marshall, and Nora Belrose. Does Transformer Interpretability Transfer to RNNs? *arXiv preprint arXiv:2404.05971*, 2024.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing: A method for rigorously testing interpretability hypotheses. *AI Alignment Forum*, 2022.

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

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*, 2024.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations (ICLR)*, 2023.

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

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.

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

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*, 2024.

Shenran Wang, Timothy Tin-Long Tse, and Jian Zhu. Understanding in-context learning beyond transformers: An investigation of state space and hybrid architectures. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2026.

Weikang Meng, Liangyu Huo, Yadan Luo, Jiawen Guan, Jingyi Zhang, Yingjian Li, and Zheng Zhang. STILL: Selecting Tokens for Intra-Layer Hybrid Attention to Linearize LLMs. *arXiv preprint arXiv:2602.02180*, 2026.

Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. Overthinking the Truth: Understanding how Language Models Process False Demonstrations. *arXiv preprint arXiv:2307.09476*, 2023.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated Linear Attention Transformers with Hardware-Efficient Training. *arXiv preprint arXiv:2312.06635*, 2023.

Jusen Du, Jiaxi Hu, Tao Zhang, Weigao Sun, and Yu Cheng. Native Hybrid Attention for Efficient Sequence Modeling. *arXiv preprint arXiv:2510.07019*, 2025.

MiniCPM Team, et al. MiniCPM-SALA: Hybridizing Sparse and Linear Attention for Efficient Long-Context Modeling. *arXiv preprint arXiv:2602.11761*, 2026.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing and language polishing, including improving grammar, sentence structure, terminology consistency, and clarity of presentation. The authors take responsibility for the final content of the manuscript, including the accuracy and integrity of its text, claims, and results.