IN-CONTEXT LEARNING VIA ATTENTION PATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning (ICL) enables language models to adapt to new tasks from demonstrations, but repeatedly including multiple examples increases context length and inference cost. Many existing implicit ICL methods encode demonstrations as fixed vectors, limiting adaptation of the intervention to individual queries and token positions. We propose Attention-Patching, an implicit ICL method that models contextual changes in self-attention outputs through task-specific updates to attention output projections. Using query-answer pairs processed with and without a fixed demonstration set, Attention-Patching fits layer-wise affine transformations from non-contextual attention representations to the demonstration-induced output changes. Across 12 classification benchmarks and four model families, Attention-Patching achieves higher average accuracy than the evaluated implicit ICL baselines while maintaining performance comparable to explicit ICL at near zero-shot inference cost. Further experiments on diverse generation tasks demonstrate its applicability to autoregressive multi-token generation.

1 INTRODUCTION

In-context learning (ICL) enables large language models (LLMs) to adapt to new tasks from a small set of input-output demonstrations without gradient-based parameter updates Brown et al. (2020); Wies et al. (2023); Schoch & Ji (2025); Min et al. (2022). In explicit ICL, however, the demonstrations must be included in the prompt for every query Hao et al. (2022); Li et al. (2025a). This increases sequence length and the associated prefill computation and memory requirements, particularly when the same task is repeatedly applied to many queries.

Recent studies address this cost through implicit ICL, which encodes demonstrations into task representations and applies them during inference without retaining the demonstration tokens. These representations are commonly derived from contextual hidden states, attention-head outputs, or activation differences Hendel et al. (2023); Todd et al. (2024); Liu et al. (2024); Liu & Deng (2025); Li et al. (2025b). Prior results show that such representations can recover a substantial portion of the task behavior induced by explicit demonstrations while using shorter inference prompts.

While vector-based implicit ICL avoids retaining demonstrations in the inference prompt, it typically represents their effect using a fixed vector. Such a representation cannot account for the fact that the influence of the same demonstrations may vary with the current query and token representation.

To address this limitation, we propose **Attention-Patching**, which learns task-specific transformations that map attention representations to the activation shifts observed under explicit ICL. By fitting layer-wise affine mappings over paired forward passes, our method enables adaptive steering, producing input-dependent corrections in both direction and magnitude rather than applying a fixed steering direction. These mappings are incorporated directly into the attention output projections, preserving the original inference architecture without introducing additional matrix multiplications.

We evaluate Attention-Patching on 12 classification benchmarks across four model families. The results demonstrate consistent improvements in average accuracy over the evaluated implicit ICL baselines, with performance approaching or exceeding explicit few-shot ICL depending on the model. Beyond classification, experiments on diverse generation tasks provide evidence that the learned patches extend to autoregressive multi-token generation. Because the learned patches are merged into the model weights, per-query inference cost of Attention-Patching remains close to zero-shot inference even when the task representation is derived from long demonstration contexts.

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.

Overall, our contributions are threefold:

- We introduce **Attention-Patching**, which represents the contextual effect of demonstrations as task-specific updates to the attention output projections.
- Across classification and multi-token generation tasks, Attention-Patching generally outperforms evaluated implicit ICL baselines while remaining competitive with explicit ICL and maintaining inference cost close to zero-shot inference.
- We analyze the adaptive steering behavior of the learned patches, showing that input-dependent weight transformations account for most of their performance and characterizing their structure across singular directions and model layers.

2 Related Work

2.1 Implicit In-Context Learning

Implicit ICL seeks to encode demonstrations into internal task representations that can be reused during query-only inference. Early approaches primarily differ in the activation source used to construct these representations. Hendel et al. (2023) introduce Task Vectors (TV), which are extracted from hidden states at selected layers; Todd et al. (2024) construct Function Vectors (FV) by aggregating the outputs of attention heads associated with ICL, Liu et al. (2024) derive In-Context Vectors (ICV) as steering directions from activation differences between paired demonstration inputs and outputs.

More recent work has focused on improving the robustness and expressiveness of these representations, as well as their injection strategies. Zhang et al. (2024) and Liu & Deng (2025) aggregate or iteratively refine activation changes across demonstrations, whereas Li et al. (2025b) integrate attention and MLP representations through calibrated layer-wise coefficients. LTV Saglam et al. (2025) and DyVec Cai et al. (2025) further replace heuristic choices with learned strategies for composing attention-derived representations and determining their injection locations.

Beyond activation-vector methods, Dherin et al. (2025) and Mazzawi et al. (2026) propose Thought Patch, showing that contextual effects as implicit updates to downstream MLP weights and study static approximations in controlled settings. Attention-Patching instead targets the attention output projection, where demonstration-induced changes can be modeled before being propagated through downstream nonlinear dynamics.

2.2 Attention-based task adaptation

Prior work has identified specialized attention heads that support different aspects of ICL, including heads that encode task functions, retrieve contextual information, and access parametric knowledge (Olsson et al., 2022; Yin & Steinhardt, 2025; Kahardipraja et al., 2025). Complementary theoretical studies in controlled regression and categorical prediction settings show that multi-head attention performs layer-dependent computations that can be interpreted as preprocessing and optimization-like inference Von Oswald et al. (2023); Chen et al. (2024); Wang et al. (2025).

Building on these findings, SITE (Park et al., 2026) learns task-specific head-wise coefficients to interpolate current head activations with fixed task embeddings extracted from few-shot prompts. In-Context Routing (ICR) (Li et al., 2026) trains a query-conditioned router on multiple tasks to combine cross-task query-key directions and modify attention logits.

These findings further motivate us to encode demonstration effects at the attention output projection, where information across attention heads is combined. Unlike SITE and ICR, Attention-Patching fits task-specific affine patches via ridge regression on activations collected from paired forward passes, without backpropagation through the language model.

3 Method

In this section, we introduce Attention-Patching by defining demonstration effect in attention (Section 3.1) and constructing affine attention patches to capture these effects (Section 3.2). We then provide an adaptive steering view of the learned patches in Section 3.3.

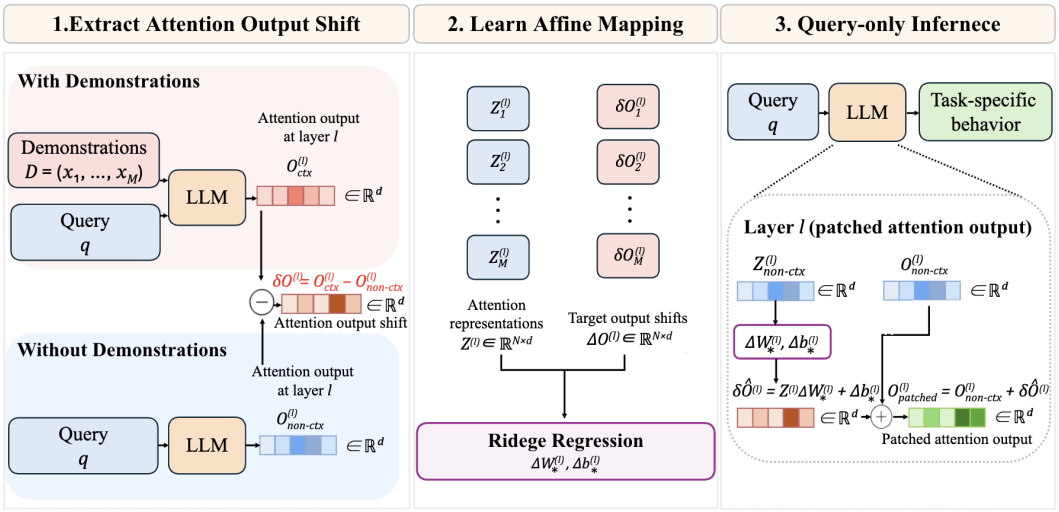


Figure 1: Framework of Attention-Patching. Attention output shifts induced by demonstrations are used to learn task-specific mappings, incorporated into the attention output projections for inference.

3.1 Demonstration Effect in Attention

Let $\mathcal{D}$ denote an ordered sequence of demonstrations, and let $\mathcal{Q} = \{(x_i, y_i)\}_{i=1}^{M}$ denote M query–answer pairs. For a query-answer pair $(x, y) \in \mathcal{Q}$, we consider two prompts that contain the same query-answer pair but differ in whether the demonstrations are included:

$$p_{\text{ctx}} = [\mathcal{D}; x; y], \qquad p_{\text{non-ctx}} = [x; y].$$

We compare the attention outputs at the token positions corresponding to the query-answer pair under these two prompts. At layer ℓ, let

$$O^{(\ell)}_{\text{ctx}}(x, y),\ O^{(\ell)}_{\text{non-ctx}}(x, y) \in \mathbb{R}^{n \times d_{\text{out}}}$$

denote the corresponding attention outputs. Here, n is the number of tokens in (x, y), and d_{out} is the representation dimension. We define the layer-wise demonstration effect as

$$\delta O^{(\ell)}(x, y) = O^{(\ell)}_{\text{ctx}}(x, y) - O^{(\ell)}_{\text{non-ctx}}(x, y).$$

3.2 Attention-Patching

Calibration At each layer ℓ, the attention output is obtained by applying the output projection $(W_O^{(\ell)}, b_O^{(\ell)})$ to the attention representation $Z^{(\ell)}$:

$$O^{(\ell)} = Z^{(\ell)} W_O^{(\ell)} + \mathbf{1}_n b_O^{(\ell)\top}, \qquad Z^{(\ell)} = \text{Concat}_{h=1}^{H}\left(z_h^{(\ell)}\right),$$

where $z_h^{(\ell)}$ denotes the output of attention head h. Accordingly, the demonstration effect defined in Section 3.1 can be expressed as

$$\delta O^{(\ell)}(x, y) = \left(Z^{(\ell)}_{\text{ctx}}(x, y) - Z^{(\ell)}_{\text{non-ctx}}(x, y)\right) W_O^{(\ell)},$$

where the shared output-projection bias cancels between the two forward passes.

Motivated by the affine structure of the output projection, we model the demonstration effect from the non-contextual attention representation as

$$\widehat{\delta O}^{(\ell)}(x, y) = Z^{(\ell)}_{\text{non-ctx}}(x, y) \Delta W^{(\ell)} + \mathbf{1}_n \Delta b^{(\ell)\top},$$

where $(\Delta W^{(\ell)}, \Delta b^{(\ell)})$ parameterize the layer-wise attention patch.

Combining the demonstration effect with the non-contextual attention output gives

$$\begin{aligned} O^{(\ell)}_{\text{ctx}} &= O^{(\ell)}_{\text{non-ctx}} + \delta O^{(\ell)} \\ &= Z^{(\ell)}_{\text{non-ctx}} W^{(\ell)}_O + \mathbf{1}_n b^{(\ell)\top}_O + \left(Z^{(\ell)}_{\text{ctx}} - Z^{(\ell)}_{\text{non-ctx}}\right) W^{(\ell)}_O \\ &\approx Z^{(\ell)}_{\text{non-ctx}} \left(W^{(\ell)}_O + \Delta W^{(\ell)}\right) + \mathbf{1}_n \left(b^{(\ell)}_O + \Delta b^{(\ell)}\right)^\top. \end{aligned}$$

Thus, approximating the demonstration effect with the affine patch is equivalent to approximating the contextual attention output through an update to the original output projection.

To calibrate the patch parameters, we collect the non-contextual attention representations and their corresponding demonstration effects over the calibration set $\mathcal{Q}$. Let $\mathbf{Z}^{(\ell)} \in \mathbb{R}^{N \times d_{\text{in}}}$ and $\mathbf{\Delta O}^{(\ell)} \in \mathbb{R}^{N \times d_{\text{out}}}$ denote the row-wise concatenations of $Z^{(\ell)}_{\text{non-ctx}}(x, y)$ and $\delta O^{(\ell)}(x, y)$ over all $(x, y) \in \mathcal{Q}$, respectively, where N is the total number of query-answer tokens in the calibration set.

The layer-wise attention patch is then fitted by ridge regression:

$$\left(\Delta W^{(\ell)}_*, \Delta b^{(\ell)}_*\right) = \arg\min_{\Delta W, \Delta b} \frac{1}{N} \left\| \mathbf{Z}^{(\ell)} \Delta W + \mathbf{1}_N \Delta b^\top - \mathbf{\Delta O}^{(\ell)} \right\|^2_F + \rho \|\Delta W\|^2_F,$$

where $\rho > 0$ controls the regularization strength. This objective admits the following closed-form solution:

$$\Delta W^{(\ell)}_* = \left(\bar{\mathbf{Z}}^{(\ell)\top} \bar{\mathbf{Z}}^{(\ell)} + N\rho I\right)^{-1} \bar{\mathbf{Z}}^{(\ell)\top} \overline{\mathbf{\Delta O}}^{(\ell)}, \qquad \Delta b^{(\ell)\top}_* = \mu^{(\ell)}_{\Delta O} - \mu^{(\ell)}_Z \Delta W^{(\ell)}_*,$$

where $\bar{\mathbf{Z}}^{(\ell)}$ and $\overline{\mathbf{\Delta O}}^{(\ell)}$ denote the centered versions of $Z^{(\ell)}$ and $\Delta O^{(\ell)}$, respectively. The full derivation and definitions are provided in Appendix A.1, with a functional view of the learned patches presented in Appendix A.2.

Inference To reduce sensitivity to the calibration set $\mathcal{Q}$, we introduce a scalar patch strength α that controls the magnitude of the calibrated patch. After calibration, the scaled layer-wise patches are merged into the original attention output projections:

$$W^{(\ell)}_O \leftarrow W^{(\ell)}_O + \alpha \Delta W^{(\ell)}_\star, \qquad b^{(\ell)}_O \leftarrow b^{(\ell)}_O + \alpha \Delta b^{(\ell)}_\star.$$

The coefficient α therefore controls the strength of the learned contextual correction while preserving the same parameterization as the original attention output projection. Once merged, the patch is applied through the existing projection and requires no additional matrix multiplication or intervention module at inference time.

3.3 Adaptive Steering with Attention-Patching

At token position t, applying the calibrated attention patch with strength α yields

$$\widehat{\delta O^t} = \alpha \left(Z^t_{\text{non-ctx}} \Delta W_\star + \Delta b^\top_\star\right),$$

where we omit the layer index for clarity. Using the singular value decomposition $\Delta W_\star = U \Sigma V^\top = \sum_i \sigma_i u_i v_i^\top$, the correction becomes

$$\widehat{\delta O^t} = \alpha \left(\sum_i \sigma_i \big(Z^t_{\text{non-ctx}} u_i\big) v_i^\top + \Delta b^\top_\star \right).$$

This decomposition provides an adaptive steering view of Attention-Patching. The right singular vectors v_i define task-specific steering directions, with coefficients $\sigma_i(Z^t_{\text{non-ctx}} u_i)$ that vary with the current attention representation across inputs and token positions. The bias term $\Delta b^\top_\star$ provides a shared additive correction, and the patch strength α controls the overall correction magnitude.

4 Experiments

4.1 Experimental Setup

Models We evaluate Attention-Patching (AP) using four open-weight language model families: Llama-3.2-3B-Instruct (Grattafiori et al., 2024), Qwen3.5-0.8B (Qwen Team, 2026), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), Gemma-2-27b-it and Gemma-2-2b-it (Team et al., 2024).

Table 1: Baseline comparison across classification benchmarks. All results are averaged over five independent runs. The best and second-best results in each column within each model are indicated in bold and underline, respectively. Method abbreviations are defined in Section 4.1.

Method	AG	SST-2	TREC	CSQA	PIQA	SST-5	MR	MRPC	CB	COPA	CREAK	AI2SciE	Avg.
Llama-3.2-3B-Instruct													
0-shot	58.8	90.8	62.0	**76.6**	75.6	33.8	89.0	66.9	67.9	94.0	79.2	87.0	73.5
ICL	86.4	**93.5**	79.9	73.4	74.9	**47.8**	**91.2**	50.7	**83.6**	88.8	**86.0**	86.9	78.6
LoRA	**87.4**	91.1	**86.3**	76.3	76.0	43.8	88.8	56.3	64.6	**94.4**	73.0	87.2	77.1
TV	68.8	90.7	69.9	76.2	76.1	31.0	90.2	40.9	54.3	90.8	79.4	87.4	71.3
I2CL	83.3	86.3	79.5	68.8	74.2	41.9	88.4	59.4	65.0	90.2	72.8	85.1	74.6
FV	61.1	91.4	81.4	71.9	74.2	31.2	88.5	53.0	82.0	82.5	77.4	**87.8**	73.5
IV	84.8	92.6	70.0	67.6	73.2	45.9	88.2	31.6	70.5	74.7	85.9	86.6	72.6
ICV	64.4	88.5	71.0	75.8	**76.4**	35.7	88.9	66.9	67.9	94.0	79.0	85.9	74.5
TP	74.6	88.0	71.4	62.4	74.6	38.4	88.4	64.4	78.6	88.2	75.2	45.7	70.8
AP (Ours)	85.8	93.0	83.9	76.2	75.8	46.7	91.1	**68.7**	79.6	93.2	**86.0**	87.7	**80.7**
Gemma-2-27b-it													
0-shot	78.2	95.2	77.6	80.4	88.8	39.6	**93.8**	63.7	85.7	98.0	87.6	**97.8**	82.2
ICL	88.6	94.7	91.6	**85.7**	**90.4**	**55.3**	93.7	75.0	90.0	**100.**	**92.7**	97.0	**87.9**
LoRA	89.4	94.6	**91.8**	82.8	87.5	52.5	91.7	**77.0**	87.9	99.0	91.4	97.6	86.9
TV	83.1	95.0	79.0	80.8	88.1	39.2	93.6	56.5	80.7	98.0	86.0	97.5	81.5
I2CL	84.4	92.6	82.4	80.2	87.7	42.1	92.3	64.7	83.6	99.2	85.9	97.2	82.7
FV	88.0	**96.0**	85.5	82.6	86.4	47.4	92.9	72.4	90.4	98.8	82.0	97.4	85.0
IV	**89.5**	94.8	87.9	83.4	85.9	49.1	92.5	75.4	92.4	99.1	89.0	97.2	86.3
ICV	78.1	94.4	78.4	81.0	88.7	39.4	**93.8**	63.5	84.6	98.0	89.3	97.6	82.2
TP	83.8	94.8	84.1	83.4	88.9	39.6	93.7	68.3	**93.9**	99.0	90.0	96.5	84.7
AP (Ours)	89.2	95.1	91.3	81.5	88.9	53.8	93.6	74.6	91.8	99.8	90.3	97.5	87.3
Qwen3.5-0.8B													
0-shot	64.2	**88.4**	62.8	49.0	59.0	28.6	**86.4**	**68.4**	32.1	56.0	57.6	79.6	61.0
ICL	81.2	83.4	76.8	56.6	62.4	**43.8**	85.4	42.7	69.6	62.2	72.6	**85.0**	68.5
LoRA	79.4	85.9	69.5	57.1	60.5	35.0	84.4	31.9	32.9	73.4	60.0	83.5	62.8
TV	73.3	86.4	64.4	53.4	**65.2**	27.4	84.8	65.3	52.1	**78.0**	58.4	83.3	66.0
I2CL	73.6	85.6	69.3	49.7	56.0	34.2	81.9	56.1	56.4	74.8	63.7	77.4	64.9
FV	75.3	81.9	77.5	26.6	53.0	33.0	76.2	33.1	52.5	45.7	62.7	64.7	56.9
IV	80.2	83.3	**80.9**	57.5	62.2	37.3	82.6	68.1	**80.0**	54.1	**72.9**	82.3	70.1
ICV	63.6	87.0	56.6	52.0	57.7	31.0	84.4	59.9	37.9	57.2	57.2	80.0	60.4
TP	54.9	78.1	49.0	37.4	61.1	27.1	81.5	68.3	58.2	57.8	59.6	41.8	56.2
AP (Ours)	**81.5**	88.3	73.2	**59.0**	64.2	37.9	85.3	67.9	63.9	73.8	66.8	83.2	**70.4**

Baselines Across 12 classification benchmarks, we compare Attention-Patching with 0-shot, explicit few-shot ICL and LoRA (Hu et al., 2022). We additionally consider five vector-based implicit ICL baselines: Task Vector (TV) (Hendel et al., 2023), Function Vector (FV) (Todd et al., 2024), Iterative Vectors (IV) (Liu & Deng, 2025), Implicit In-context Learning (I2CL) (Li et al., 2025b), and ICV (Liu et al., 2024). We also compare against Thought Patching (TP) (Mazzawi et al., 2026). For all baselines, we follow the experimental details reported in their original papers.

Evaluation Following Li et al. (2026), we evaluate each method on up to 500 instances from the evaluation split of each of the 12 benchmarks, using all available instances for smaller datasets. We evaluate the classification tasks: AGNews (Zhang et al., 2015), SST-2 (Socher et al., 2013), TREC (Li & Roth, 2002), CSQA (Talmor et al., 2019), PIQA (Bisk et al., 2020), SST-5 (Socher et al., 2013), MR (Pang & Lee, 2005), MRPC (Dolan & Brockett, 2005), CB (de Marneffe et al., 2021), COPA (Roemmele et al., 2011), CREAK (Onoe et al., 2021), and AI2SciE (Clark et al., 2018), using five random seeds. For the multi-token prediction task, we evaluate on GSM8K (Cobbe et al., 2021), FLORES (Korean and Spanish subsets) (Goyal et al., 2022), DailyMail (See et al., 2017) and CommonGen (Lin et al., 2020). Other details are in Appendix B.1.

4.2 MAIN RESULTS

Table 1 presents the results on Llama-3.2-3B-Instruct, Gemma-2-27b-it, and Qwen3.5-0.8B. Additional results on Mistral-7B-Instruct-v0.3 and Gemma-2-2b-it are provided in Appendix C.1.

Attention-Patching outperforms implicit ICL baselines. As shown in Table 1, Attention-Patching achieves the highest average accuracy among the evaluated implicit ICL methods across all three models. With 25 calibration examples per class, Attention-Patching obtains average accuracies of 80.7, 87.3, and 70.4 on Llama-3.2-3B-Instruct, Gemma-2-27b-it, and Qwen3.5-0.8B, respectively, outperforming the strongest implicit ICL baseline on each model by 6.1, 1.0, and 0.3 percentage points.

Attention-Patching remains competitive with explicit ICL. Despite processing only query tokens at inference time, Attention-Patching achieves performance comparable to explicit ICL across the three models. Specifically, Attention-Patching exceeds explicit ICL by 2.1 and 1.9 percentage points on Llama and Qwen, respectively, while remaining within 0.6 points on Gemma-2-27b-it. Compared with LoRA, Attention-Patching also achieves higher average accuracy across all three models, with improvements of 3.6, 0.4, and 7.6 percentage points, respectively. These findings suggest that the learned weight-space transformations can match or exceed the predictive performance of explicit prompting and LoRA on the evaluated classification tasks.

Table 2: Multi-token generation results under different context lengths. We report flexible-match accuracy on GSM8K and ROUGE-L F1 scores on FLORES, DailyMail and CommonGen.

Method	GSM8K	FLORES(Ko)	FLORES(Sp)	DailyMail	CommonGen
Llama-3.2-3B-Instruct					
Zero-shot	50.6	5.5	50.6	17.2	46.1
LoRA	58.4	12.2	51.5	18.0	48.4
Many-shot (1K)	72.6	20.6	51.8	22.1	58.3
ICV	52.2	4.5	51.9	17.6	48.6
AP	71.6	20.5	51.7	19.8	56.7
Many-shot (2K)	73.8	20.6	51.6	24.8	58.1
ICV	52.0	4.7	51.8	17.7	48.6
AP	70.4	19.5	51.8	22.2	57.5
Many-shot (4K)	71.8	21.4	51.7	25.8	57.2
ICV	52.0	4.5	51.7	17.7	48.3
AP	70.6	20.8	51.8	24.3	56.3
Many-shot (8K)	69.4	21.5	51.3	25.6	56.4
ICV	52.2	4.6	51.8	17.6	48.6
AP	66.6	20.8	51.9	23.6	55.0
Many-shot (16K)	70.8	21.1	51.4	26.1	57.7
ICV	52.2	4.7	51.8	17.6	48.4
AP	64.8	20.9	51.7	24.7	54.1
Gemma-2-27b-it					
Zero-shot	79.8	23.4	55.6	21.4	55.7
LoRA	64.0	22.9	52.6	11.0	59.8
Many-shot (1K)	89.2	23.0	56.1	25.8	62.4
ICV	79.6	23.5	55.8	21.7	55.5
AP	84.6	23.9	55.9	25.4	59.5
Many-shot (2K)	88.8	22.8	55.9	26.8	62.1
ICV	79.6	23.6	55.7	21.7	55.3
AP	85.2	23.7	55.7	26.2	56.8
Many-shot (4K)	89.2	23.1	55.9	26.8	62.9
ICV	79.6	23.6	55.6	22.1	55.8
AP	82.8	23.9	55.6	26.9	60.6

4.3 Multi-token Generation Task

To examine whether the learned patches remain effective beyond single-token classification, we evaluate Attention-Patching on four autoregressive generation tasks using 32 calibration samples. GSM8K is evaluated using flexible-match accuracy, while the remaining tasks are evaluated using ROUGE-L F1. We compare Attention-Patching against ICV, whose original formulation supports autoregressive generation. Other implicit ICL baselines primarily target single-token prediction and lack a standardized intervention procedure for multi-token decoding.

As shown in Table 2, Attention-Patching generally outperforms ICV across the evaluated generation tasks and demonstration-context budgets. Averaged across the five context budgets on Llama, Attention-Patching achieves 68.8 accuracy on GSM8K, compared with 52.1 for ICV and 71.7 for explicit many-shot ICL. On FLORES Korean and Spanish, Attention-Patching obtains average ROUGE-L F1 scores of 20.5 and 51.8, respectively, compared with 4.6 and 51.8 for ICV and 21.0 and 51.6 for many-shot ICL. On DailyMail and CommonGen, Attention-Patching achieves average ROUGE-L F1 scores of 22.9 and 55.9 on Llama, respectively, compared with 17.6 and 48.5 for ICV. On Gemma-2-27b-it, Attention-Patching achieves 26.2 and 59.0, respectively, approaching the performance of explicit many-shot ICL on both tasks. These results suggest that the learned patches retain their effectiveness during autoregressive multi-token generation.

5 Analysis

5.1 Inference Efficiency

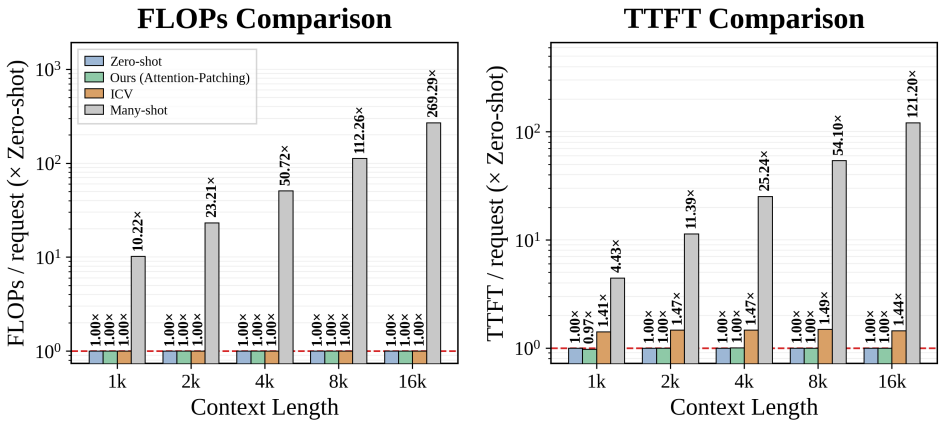


Figure 2: Average inference FLOPs and TTFT on GSM8K using Llama-3.2-3B-Instruct across different context lengths, averaged over 500 samples.

To investigate whether Attention-Patching preserves the inference efficiency, we measure its computational cost as the demonstration context used for calibration increases in length.

FLOPs. Figure 2 compares the per-query inference cost of Attention-Patching with zero-shot, ICV, and explicit many-shot ICL on GSM8K by varying the context length from 1K to 16K tokens. We estimate FLOPs using an analytical FLOP counter based on runtime tensor shapes. All results are normalized by the corresponding zero-shot result. As context length increases, the computational cost of explicit ICL rises from $10.22\times$ to $269.29\times$ that of zero-shot inference. In contrast, Attention-Patching maintains approximately the same per-query FLOPs as zero-shot inference, as the learned updates are incorporated directly into the existing attention output projections.

TTFT. A similar trend holds for time-to-first-token (TTFT). Many-shot ICL increases from $4.43\times$ to $121.20\times$, while Attention-Patching remains within 0.97–$1.00\times$ of zero-shot inference. ICV also avoids processing demonstration tokens and has approximately constant FLOPs, but incurs a TTFT overhead of 1.41–$1.49\times$, since each layer and token position requires adding a steering vector and

renormalizing the activation. Together, the FLOPs and TTFT measurements show that Attention-Patching introduces negligible per-query inference overhead.

5.2 Role of Linear and Bias Components

To understand whether the gains of Attention-Patching stem from input-dependent transformations or fixed additive corrections, we compare linear, bias, and full affine patch formulations. We evaluate two variants: Bias-only, parameterized by $\Delta b^{(\ell)}$, and Weight-only, parameterized by $Z^{(\ell)}\Delta W^{(\ell)}$.

Table 3: Component ablation of Attention-Patching on Llama-3.2-3B-Instruct. We report average accuracy across the 12 classification benchmarks.

Method	AG	SST-2	TREC	CSQA	PIQA	SST-5	MR	MRPC	CB	COPA	CREAK	AI2SciE	Avg.
Bias	58.8	87.1	68.7	76.1	75.8	34.2	89.0	68.2	58.2	93.8	77.2	87.0	72.8
Weight	85.7	93.1	84.6	75.7	76.0	46.6	90.8	68.3	80.0	92.6	86.0	87.6	80.6
Full	85.8	93.0	83.9	76.2	75.8	46.7	91.1	68.7	79.6	93.2	86.0	87.7	80.7

As shown in Table 3, the weight component accounts for most of the performance of the full patch. On the classification benchmarks, Bias-only achieves an average accuracy of 72.8, whereas Weight-only reaches 80.6, nearly matching the 80.7 accuracy of the full Bias + Weight parameterization.

From the adaptive steering perspective, $\Delta W^{(\ell)}$ combines multiple steering directions with input-dependent coefficients, whereas $\Delta b^{(\ell)}$ provides only a single fixed steering vector. This explains the smaller contribution of the bias component and suggests that the effectiveness of Attention-Patching primarily arises from input-dependent transformations rather than fixed additive corrections.

5.3 Singular Structure of Adaptive Steering

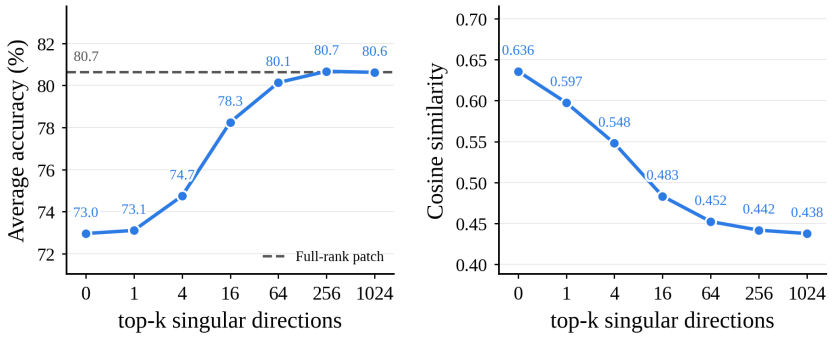


Figure 3: Singular structure of adaptive steering on Llama-3.2-3B-Instruct. Average accuracy across 12 classification tasks (left) and cosine similarity to the bias-only steering vector averaged across tasks and layers (right) using rank-k truncated SVD of the weight patch.

To empirically examine the adaptive steering interpretation in Section 3.3, we compare the learned weight patch with a bias-only variant. For the bias-only variant, the closed-form solution is $\Delta b_{\text{bias}}^{\top} = \mu_{\Delta O} = \frac{1}{N}\sum_{j=1}^{N}\delta O_j$, which corresponds to a fixed steering vector and matches the averaging form used in Contrastive Activation Addition (Rimsky et al., 2024).

We then replace $\Delta W_\star$ with its rank-k truncated SVD, $\Delta W_{\star,k} = \sum_{i=1}^{k}\sigma_i u_i v_i^{\top}$, while retaining the bias $\Delta b_\star$ learned jointly with the full affine patch. The resulting correction is

$$\widehat{\delta O_k^t} = \alpha\left(\sum_{i=1}^{k}\sigma_i\left(Z_{\text{non-ctx}}^{t}u_i\right)v_i^{\top} + \Delta b_\star^{\top}\right).$$

Unlike a fixed steering vector, the contribution of each singular direction is modulated by the current attention representation. Note that $k = 0$ retains only the bias of the full affine patch and is therefore distinct from the separately fitted bias-only variant.

As shown in Figure 3, the $k = 0$ intervention shows positive alignment with the bias-only steering vector, with a cosine similarity of 0.636. As the leading singular components are incorporated, the similarity steadily decreases to 0.438, while average accuracy increases from 73.0 to 80.7. This trend supports the adaptive steering view, where singular directions provide input-dependent refinements beyond a fixed steering vector.

5.4 Layer-wise Structure of Learned Patches

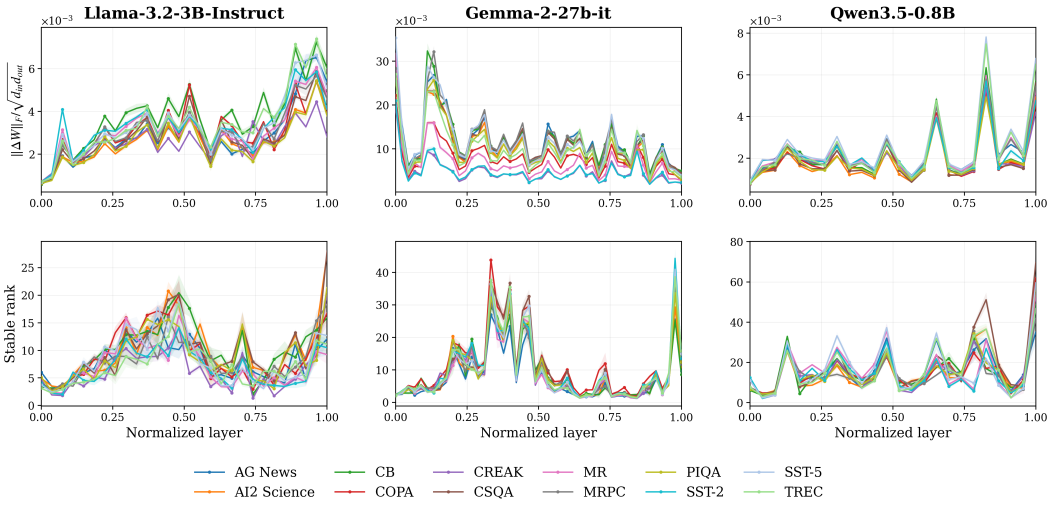


Figure 4: Layer-wise structure of the learned attention patches across classification benchmarks. The top row shows the normalized Frobenius norm of the patch weights, while the bottom row shows their stable rank. Each column corresponds to a model architecture, and each curve represents one classification task.

Figure 4 illustrates the layer-wise structure of the learned patches across different model architectures, characterizing their update magnitude and effective dimensionality using the normalized Frobenius norm and stable rank, respectively.

For each layer ℓ, the top row of Figure 4 reports the update magnitude using the normalized Frobenius norm: $m^{(\ell)} = \frac{\|\Delta W^{(\ell)}\|_F}{\sqrt{d_{\text{in}} d_{\text{out}}}}$, and the bottom row reports the stable rank of each matrix update: $\operatorname{srank}\big(\Delta W^{(\ell)}\big) = \frac{\|\Delta W^{(\ell)}\|_F^2}{\|\Delta W^{(\ell)}\|_2^2} = \frac{\sum_i \big(\sigma_i^{(\ell)}\big)^2}{\big(\max_i \sigma_i^{(\ell)}\big)^2}$, where $\sigma_i^{(\ell)}$ denotes the i-th singular value of $\Delta W^{(\ell)}$.

The two measures exhibit distinct layer-wise patterns across model architectures, indicating that larger updates do not necessarily correspond to higher-dimensional transformations. This suggests that Attention-Patching distributes task-specific corrections across different layers and singular directions rather than relying on a uniform intervention throughout the model.

6 Conclusion

We introduce Attention-Patching, a method that compresses the effects of in-context demonstrations into attention patches, enabling query-only inference without repeatedly processing the demonstrations. Across 12 classification benchmarks, Attention-Patching achieved the strongest average performance among the evaluated implicit ICL methods, while remaining competitive with explicit ICL. Attention-Patching also generally outperformed ICV on multi-token generation tasks across context budgets from 1K to 16K tokens, demonstrating its applicability beyond single-token classification. These results suggest Attention-Patching as an effective approach for retaining demonstration-induced behavior while reducing the inference-time overhead of explicit ICL.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation, improve the readability and clarity of the manuscript, assist in the writing of proofs, and assist with code implementation. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final content of this work. We have not used generative AI tools for developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning and reformatting datasets, or interpreting results. Generating synthetic datasets and supporting qualitative and thematic data analysis were not applicable to this work.

REPRODUCIBILITY STATEMENT

We provide the source code and configurations for all key experiments, including model training and evaluation. Detailed descriptions of the proposed method, experimental settings, and implementation details are provided in Section 4.1 and Appendix B.1 to facilitate reproducibility.

REFERENCES

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Wang Cai, Hsiu-Yuan Huang, Zhixiang Wang, and Yunfang Wu. Beyond demonstrations: Dynamic vector construction from latent representations. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5842–5857, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.297. URL https://aclanthology.org/2025.emnlp-main.297/.

Xingwu Chen, Lei Zhao, and Difan Zou. How transformers utilize multi-head attention in in-context learning? a case study on sparse linear regression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=fG8TukiXa5.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. Universal Dependencies. *Computational Linguistics*, 47(2):255–308, June 2021. doi: 10.1162/coli_a_00402. URL https://aclanthology.org/2021.cl-2.11/.

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.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. URL https://aclanthology.org/I05-5002/.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538, 2022.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yaru Hao, Yutao Sun, Li Dong, Zhixiong Han, Yuxian Gu, and Furu Wei. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713*, 2022.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.624. URL https://aclanthology.org/2023.findings-emnlp.624/.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Patrick Kahardipraja, Reduan Achtibat, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. The atlas of in-context learning: How attention heads shape in-context retrieval augmentation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=5h9mS87Pyt.

Jiaqian Li, Yanshu Li, Ligong Han, Ruixiang Tang, and Wenya Wang. Train once, reuse everywhere: Generalizable implicit ICL by routing attention. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Gb2mqvRRa0.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context LLMs struggle with long in-context learning. *Transactions on Machine Learning Research*, 2025a. ISSN 2835-8856. URL https://openreview.net/forum?id=Cw2xlg0e46.

Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. URL https://aclanthology.org/C02-1150/.

Zhuowei Li, Zihao Xu, Ligong Han, Yunhe Gao, Song Wen, Di Liu, Hao Wang, and Dimitris Metaxas. Implicit in-context learning. In *International Conference on Learning Representations*, volume 2025, pp. 101644–101666, 2025b.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1823–1840, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.165. URL https://aclanthology.org/2020.findings-emnlp.165/.

Sheng Liu, Haotian Ye, Lei Xing, and James Y. Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=dJTChKgv3a.

Yiting Liu and Zhi-Hong Deng. Iterative vectors: In-context gradient steering without backpropagation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=1v3XEcRMyP.

Hanna Mazzawi, Benoit Dherin, Michael Munn, Adrian Goldwaser, Michael Wunder, and Javier Gonzalvo. Transmuting prompts into weights, 2026. URL https://arxiv.org/abs/2510.08734.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11048–11064, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.759. URL https://aclanthology.org/2022.emnlp-main.759/.

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.

Yasumasa Onoe, Michael JQ Zhang, Eunsol Choi, and Greg Durrett. CREAK: A dataset for commonsense reasoning over entity knowledge. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=mbW_GT3ZN-.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Kevin Knight, Hwee Tou Ng, and Kemal Oflazer (eds.), *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 115–124, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219855. URL https://aclanthology.org/P05-1015/.

Jungwon Park, Jimyeong Kim, Changin Choi, and Wonjong Rhee. Soft head selection for injecting ICL-derived task embeddings. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 27161–27214, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1355. URL https://aclanthology.org/2026.findings-acl.1355/.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, Stanford University, March 2011. URL http://ict.usc.edu/pubs/Choice%20of%20Plausible%20Alternatives-%20An%20Evaluation%20of%20Commonsense%20Causal%20Reasoning.pdf.

Baturay Saglam, Xinyang Hu, Zhuoran Yang, Dionysis Kalogerias, and Amin Karbasi. Learning task representations from in-context learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6634–6663, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.345. URL https://aclanthology.org/2025.findings-acl.345/.

Stephanie Schoch and Yangfeng Ji. The good, the bad, and the debatable: A survey on the impacts of data for in-context learning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn

Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 29798–29812, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1514. URL https://aclanthology.org/2025.emnlp-main.1514/.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://www.aclweb.org/anthology/P17-1099.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron Wallace, and David Bau. Function vectors in large language models. In *International conference on learning representations*, volume 2024, pp. 17282–17333, 2024.

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*, pp. 35151–35174. PMLR, 2023.

Aaron T Wang, William Convertino, Xiang Cheng, Ricardo Henao, and Lawrence Carin. On understanding attention-based in-context learning for categorical data. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=7Daf4TMtX9.

Noam Wies, Yoav Levine, and Amnon Shashua. The learnability of in-context learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=f3JNQd7CHM.

Kayo Yin and Jacob Steinhardt. Which attention heads matter for in-context learning? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=C7XmEByCFv.

Kaiyi Zhang, Ang Lv, Yuhan Chen, Hansen Ha, Tao Xu, and Rui Yan. Batch-icl: Effective, efficient, and order-agnostic in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10728–10739, 2024.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

A Theoretical Analysis of Attention-Patching

A.1 Derivation of the Closed-Form Ridge Solution

In this section, we derive the closed-form solution used to compute the layer-wise attention patch in Attention-Patching. For conciseness, we omit the layer index ℓ throughout the derivation.

We collect the non-contextual attention representations and their target demonstration effects over all query-answer token positions in the calibration set as

$$\mathbf{Z} \in \mathbb{R}^{N \times d_{\text{in}}}, \qquad \mathbf{\Delta O} \in \mathbb{R}^{N \times d_{\text{out}}},$$

where N is the total number of query-answer token positions. The rows of $\mathbf{Z}$ are taken from $Z_{\text{non-ctx}}$, and the corresponding rows of $\mathbf{\Delta O}$ are the demonstration effects δO defined in Section 3.1.

The patch parameters

$$\Delta W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}, \qquad \Delta b \in \mathbb{R}^{d_{\text{out}}}$$

are fitted by

$$\min_{\Delta W, \Delta b} \mathcal{L}(\Delta W, \Delta b) = \frac{1}{N} \left\| \mathbf{Z}\Delta W + \mathbf{1}_N \Delta b^\top - \mathbf{\Delta O} \right\|_F^2 + \rho \left\| \Delta W \right\|_F^2,$$

where $\mathbf{1}_N$ is the all-ones vector and $\rho > 0$ is the ridge coefficient.

We first solve for the bias term. Differentiating the objective with respect to Δb gives

$$\nabla_{\Delta b} \mathcal{L} = \frac{2}{N} \left(\mathbf{Z}\Delta W + \mathbf{1}_N \Delta b^\top - \mathbf{\Delta O} \right)^\top \mathbf{1}_N.$$

Setting the gradient to zero yields

$$\mathbf{1}_N^\top \mathbf{Z} \Delta W + N \Delta b^\top - \mathbf{1}_N^\top \mathbf{\Delta O} = 0.$$

We define the row means as

$$\mu_Z = \frac{1}{N} \mathbf{1}_N^\top \mathbf{Z}, \qquad \mu_{\Delta O} = \frac{1}{N} \mathbf{1}_N^\top \mathbf{\Delta O}.$$

For any fixed ΔW, the optimal bias is therefore

$$\Delta b^\top = \mu_{\Delta O} - \mu_Z \Delta W.$$

We then center the two matrices as

$$\overline{\mathbf{Z}} = \mathbf{Z} - \mathbf{1}_N \mu_Z, \qquad \overline{\mathbf{\Delta O}} = \mathbf{\Delta O} - \mathbf{1}_N \mu_{\Delta O}.$$

Substituting the expression for Δb into the residual gives

$$\begin{aligned} \mathbf{Z}\Delta W + \mathbf{1}_N \Delta b^\top - \mathbf{\Delta O} &= \mathbf{Z}\Delta W + \mathbf{1}_N \left(\mu_{\Delta O} - \mu_Z \Delta W\right) - \mathbf{\Delta O} \\ &= \left(\mathbf{Z} - \mathbf{1}_N \mu_Z\right) \Delta W - \left(\mathbf{\Delta O} - \mathbf{1}_N \mu_{\Delta O}\right) \\ &= \overline{\mathbf{Z}}\Delta W - \overline{\mathbf{\Delta O}}. \end{aligned}$$

The objective therefore reduces to

$$\mathcal{L}(\Delta W) = \frac{1}{N} \left\| \overline{\mathbf{Z}}\Delta W - \overline{\mathbf{\Delta O}} \right\|_F^2 + \rho \left\| \Delta W \right\|_F^2.$$

Differentiating with respect to ΔW gives

$$\nabla_{\Delta W} \mathcal{L} = \frac{2}{N} \overline{\mathbf{Z}}^\top \left(\overline{\mathbf{Z}}\Delta W - \overline{\mathbf{\Delta O}} \right) + 2\rho \Delta W.$$

Setting the gradient to zero and rearranging gives

$$\left(\overline{\mathbf{Z}}^\top \overline{\mathbf{Z}} + N\rho I_{d_{\text{in}}} \right) \Delta W = \overline{\mathbf{Z}}^\top \overline{\mathbf{\Delta O}}.$$

Since $\rho > 0$, the matrix on the left is positive definite and therefore invertible. The optimal weight patch is

$$\Delta W_\star = \left(\overline{\mathbf{Z}}^\top \overline{\mathbf{Z}} + N\rho I_{d_{\text{in}}} \right)^{-1} \overline{\mathbf{Z}}^\top \overline{\mathbf{\Delta O}},$$

with the corresponding bias

$$\Delta b_\star^\top = \mu_{\Delta O} - \mu_Z \Delta W_\star.$$

Restoring the layer index, the calibrated patch parameters are

$$\Delta W_\star^{(\ell)} = \left(\overline{\mathbf{Z}}^{(\ell)\top} \overline{\mathbf{Z}}^{(\ell)} + N\rho I_{d_{\text{in}}} \right)^{-1} \overline{\mathbf{Z}}^{(\ell)\top} \overline{\mathbf{\Delta O}}^{(\ell)},$$

and

$$\Delta b_\star^{(\ell)\top} = \mu_{\Delta O}^{(\ell)} - \mu_Z^{(\ell)} \Delta W_\star^{(\ell)}.$$

A.2 Functional View

Consider a single attention head and a target position t in a query-answer pair, and omit the layer and head indices for clarity. Let $\mathcal{X}_t$ denote the source positions from the query-answer pair visible to t, and let $\mathcal{D}_t$ denote the demonstration-token positions visible to t in the contextual prompt. The non-contextual attention representation is

$$z_t^{\text{non-ctx}} = \frac{\sum_{s \in \mathcal{X}_t} \exp(a_{ts}^{\text{non-ctx}}) v_s^{\text{non-ctx}}}{\sum_{s \in \mathcal{X}_t} \exp(a_{ts}^{\text{non-ctx}})}.$$

When demonstrations are added, the attention weights and value representations of the original context may change. We define the contextual attention mass over $\mathcal{X}_t$ and the corresponding normalized representation as

$$A_t^{\text{ctx}} = \sum_{s \in \mathcal{X}_t} \exp(a_{ts}^{\text{ctx}}), \qquad z_{t,\mathcal{X}}^{\text{ctx}} = \frac{1}{A_t^{\text{ctx}}} \sum_{s \in \mathcal{X}_t} \exp(a_{ts}^{\text{ctx}}) v_s^{\text{ctx}}.$$

For the demonstration-token positions, we similarly define

$$B_t = \sum_{m \in \mathcal{D}_t} \exp(a_{tm}^{\text{ctx}}), \qquad r_{D,t} = \frac{1}{B_t} \sum_{m \in \mathcal{D}_t} \exp(a_{tm}^{\text{ctx}}) v_m^{\text{ctx}}.$$

The contextual attention representation can then be written as

$$z_t^{\text{ctx}} = (1-\lambda_t) z_{t,\mathcal{X}}^{\text{ctx}} + \lambda_t r_{D,t}, \qquad \lambda_t = \frac{B_t}{A_t^{\text{ctx}} + B_t}.$$

Subtracting the non-contextual output gives

$$z_t^{\text{ctx}} - z_t^{\text{non-ctx}} = \underbrace{\lambda_t r_{D,t}}_{\text{demonstration retrieval}} - \underbrace{\lambda_t z_t^{\text{non-ctx}}}_{\text{attention redistribution}} + \underbrace{(1-\lambda_t)\left(z_{t,\mathcal{X}}^{\text{ctx}} - z_t^{\text{non-ctx}}\right)}_{\text{context change}}.$$

This decomposition shows that the effect of demonstrations is not a single fixed shift. At each token position, the head-level changes are combined across attention heads to form Z_{ctx}^t and $Z_{\text{non-ctx}}^t$. Over all token positions, these representations give the layer-wise demonstration effect

$$\delta O = (Z_{\text{ctx}} - Z_{\text{non-ctx}})\, W_O.$$

From this perspective, Attention-Patching can be viewed as compactly representing these demonstration-induced changes through an affine correction, which provides the basis for the adaptive steering view.

B Experimental Setup

B.1 Reproducibility Details

All classification experiments are repeated using five random seeds, $\{42, 43, 44, 45, 46\}$, and the results are averaged across the five runs. All experiments are conducted on NVIDIA RTX 6000 Ada Generation GPUs.

For each classification task, we use a fixed demonstration set containing five examples per class and construct a disjoint calibration set from the training split. Attention-Patching uses 25 calibration examples per class. We select the patch strength α by sweeping $\alpha \in \{0.001, 0.1, 0.2, \ldots, 1.0\}$ over 100 held-out validation instances and choosing the value that achieves the best validation performance. For evaluation, we use up to 500 instances from the evaluation split of each classification benchmark, using all available instances for smaller datasets.

For multi-token generation, we use 32 calibration examples and follow the same procedure for selecting the patch strength α. For evaluation, we use 500 instances for each task, with demonstration-context budgets ranging from 1K to 16K tokens; for models with shorter context windows, we use budgets from 1K to 4K tokens.

Figure 5 reports the selected patch strengths across tasks and models.

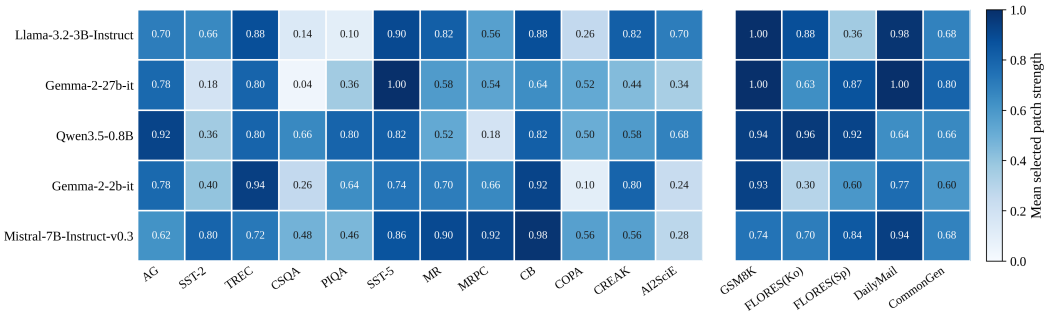


Figure 5: Selected patch strength α for each task and model. The left panel corresponds to classification tasks, where values are averaged across five independent runs, and the right panel corresponds to multi-token generation tasks, where values are averaged across the evaluated context lengths for each model. In each experiment, α is selected on the held-out validation set following the procedure described above.

Table 4: Prompt templates used in the classification experiments.

Dataset	Task Type	Template
AGNews	Topic classification	News: {text}; Type: [World, Sports, Business, Technology]
SST-2	Sentiment	Review: {text}; Sentiment: [negative, positive]
TREC	Question type classification	Question: {text}; Answer Type: [Abbreviation, Entity, Description, Person, Location, Number]
CSQA	Commonsense	Question: {question}; A. {optA}; B. {optB}; C. {optC}; D. {optD}; E. {optE}; Answer (A/B/C/D/E); Options: [A, B, C, D, E]
PIQA	Physical commonsense	Goal: {goal}; A. {optA}; B. {optB}; Answer (A/B); Options: [A, B]
SST-5	Sentiment	Sentence: {text}; Sentiment: [terrible, negative, neutral, positive, great]
MR	Movie review	Review: {text}; Sentiment: [negative, positive]
MRPC	Paraphrase	{pair}; A. Paraphrase; B. Not paraphrase; Answer (A/B); Options: [A, B]
CB	NLI	{pair}; A. Entailment; B. Contradiction; C. Neutral; Answer (A/B/C); Options: [A, B, C]
CREAK	Claim verification	Claim: {claim}; Label: yes / no; Options: [yes, no]
COPA	Causal reasoning	{context}; A. {optA}; B. {optB}; Answer (A/B); Options: [A, B]
AI2SciE	Science MCQ	Question: {question}; A. {optA}; B. {optB}; C. {optC}; D. {optD}; E. {optE}; F. {optF}; G. {optG}; H. {optH}; Answer (A/B/C/...); Options: [A, B, C, D, E, F, G, H]

B.2 Classification Prompt Templates

Table 4 summarizes the dataset-specific prompt templates used in our classification experiments. Each template specifies the input format and the corresponding set of candidate labels.

B.3 LoRA Setup

For comparison with Attention-Patching, we apply LoRA to the attention output projection (`o_proj`), corresponding to the projection modified by our method. We use a rank of 8, a scaling factor of 16, and a dropout rate of 0.05. The model is trained for one epoch using AdamW with a learning rate of 2×10^{-4} and a batch size of 4. We use no weight decay or learning-rate warmup and set the maximum gradient norm to 1.0.

For classification, LoRA is trained on 5, 15 or 25 examples per class, corresponding to the data budgets reported in the experiments. For multi-token generation, we use 32 training examples, matching the calibration-set size used for Attention-Patching.

C Additional Results

C.1 Classification Results on Additional Models

Table 5: Baseline comparison across classification benchmarks. All results are averaged over five independent runs. The best and second-best results in each column within each model are indicated in bold and underline, respectively.

Method	AG	SST-2	TREC	CSQA	PIQA	SST-5	MR	MRPC	CB	COPA	CREAK	AI2SciE	Avg.
Gemma-2-2b-it													
0-shot	75.6	92.6	72.8	69.8	71.8	36.8	91.6	45.3	62.5	94.0	72.4	87.4	72.7
ICL	83.3	**93.6**	83.3	**72.2**	**75.5**	**48.0**	**92.2**	69.2	**76.1**	92.0	**87.0**	**88.8**	80.1
LoRA	**87.0**	91.6	81.7	68.0	75.2	44.6	91.0	45.2	71.4	94.4	64.8	87.9	75.2
TV	85.4	92.5	76.4	70.3	71.8	37.7	90.0	43.5	57.1	93.4	79.3	86.6	73.7
I2CL	82.8	86.8	75.8	66.2	73.4	36.4	89.4	64.0	61.1	93.0	62.5	85.1	73.0
FV	75.2	79.9	78.7	65.8	69.8	29.8	75.5	69.2	72.1	90.6	55.6	84.6	70.6
IV	78.2	90.4	75.9	66.5	59.8	45.5	87.2	65.7	59.3	85.8	77.6	84.6	73.0
ICV	75.6	92.1	72.8	69.6	72.9	38.5	91.8	47.3	59.3	94.8	70.7	86.7	72.7
TP	51.8	89.0	47.5	54.8	69.6	34.8	87.7	68.3	55.7	92.0	81.0	65.8	66.5
AP (Ours)	86.6	93.2	**88.9**	70.5	74.9	44.3	91.5	**70.6**	74.3	**95.0**	85.9	87.8	**80.3**
Mistral-7B-Instruct-v0.3													
0-shot	76.6	90.4	58.8	69.6	77.0	36.2	89.2	32.6	19.6	92.0	51.6	86.8	65.0
ICL	83.5	94.0	80.9	**76.2**	**80.0**	51.1	93.1	69.6	88.2	**96.6**	**90.7**	**90.0**	**82.8**
LoRA	**86.9**	90.4	**86.7**	69.8	78.2	46.2	89.0	51.5	71.4	92.8	77.2	87.3	77.3
TV	81.6	88.2	61.7	71.4	77.4	36.5	92.1	35.8	30.4	93.2	57.0	87.1	67.7
I2CL	85.1	90.4	76.7	64.8	77.0	44.6	88.2	63.3	73.2	93.6	81.4	87.6	77.2
FV	85.7	93.9	62.3	74.4	79.9	43.4	92.4	73.7	**95.5**	96.1	90.0	89.4	81.4
IV	81.4	92.1	84.4	73.4	79.5	40.3	91.0	**73.9**	93.5	95.2	89.8	88.6	81.9
ICV	76.6	90.1	64.4	71.0	76.8	36.0	90.4	32.6	27.1	92.6	83.3	86.9	69.0
TP	69.5	90.6	68.9	71.0	78.8	39.4	91.1	68.1	78.2	96.2	89.0	85.3	77.2
AP (Ours)	85.8	**94.3**	79.8	74.2	79.6	**52.0**	**93.7**	71.9	87.5	95.8	90.4	87.9	82.7

Table 5 presents additional classification results on Gemma-2-2b-it and Mistral-7B-Instruct-v0.3, complementing the main results reported in Section 4.2.

C.2 Multi-token Generation Results on Additional Models

Table 6 presents additional multi-token generation results on Mistral-7B-Instruct-v0.3, Gemma-2-2b-it, and Qwen3.5-0.8B across mathematical reasoning, translation, summarization, and constrained text generation.

On Mistral-7B-Instruct-v0.3, Attention-Patching consistently outperforms ICV across all tasks and context lengths, achieving 58.4 accuracy on GSM8K at 1K tokens, compared with 55.0 for explicit ICL and 26.8 for ICV. On Gemma-2-2b-it, where explicit ICL underperforms zero-shot inference on GSM8K, Attention-Patching shows a similar trend while remaining close to explicit ICL on the translation tasks. On Qwen3.5-0.8B, Attention-Patching consistently outperforms ICV, although its performance relative to explicit ICL varies across tasks and context lengths.

These findings indicate that the effectiveness of Attention-Patching depends in part on the contextual behavior of the underlying model.

D Additional Ablation Studies

D.1 Effect of Calibration Set Size

To examine the sensitivity of Attention-Patching to calibration set size, we compare its performance with LoRA and Thought Patching under different data budgets. We consider 5, 15 and 25 examples per class.

Table 6: Multi-token generation results under different context lengths. We report flexible-match accuracy on GSM8K and ROUGE-L F1 scores on FLORES, DailyMail and CommonGen.

Method	GSM8K	FLORES(Ko)	FLORES(Sp)	DailyMail	CommonGen
Gemma-2-2b-it					
Zero-shot	56.0	21.0	52.9	19.2	48.2
LoRA	57.6	20.8	53.1	16.4	51.0
Many-shot (1K)	48.0	22.5	53.6	23.9	55.3
ICV	56.0	21.1	53.1	19.5	48.1
AP	44.4	21.6	53.5	24.1	52.0
Many-shot (2K)	48.8	22.6	53.4	24.4	55.2
ICV	56.4	21.1	53.0	19.4	48.2
AP	48.2	21.8	53.1	23.0	54.2
Many-shot (4K)	47.0	22.6	53.5	24.6	53.1
ICV	56.8	20.7	53.0	19.2	48.4
AP	48.4	21.4	53.3	23.9	53.2
Mistral-7B-Instruct-v0.3					
Zero-shot	26.4	18.4	50.9	20.6	46.2
LoRA	35.4	22.1	50.8	24.6	51.1
Many-shot (1K)	55.0	22.0	52.9	26.9	59.3
ICV	26.8	17.6	51.8	21.8	49.2
AP	58.4	22.3	52.7	25.6	57.7
Many-shot (2K)	54.2	22.2	53.0	27.6	60.3
ICV	26.8	17.5	51.9	22.0	48.8
AP	51.8	22.5	52.8	26.0	56.8
Many-shot (4K)	49.4	22.7	53.1	28.1	61.3
ICV	27.0	17.6	51.8	22.0	48.9
AP	52.6	23.1	53.0	26.4	59.6
Many-shot (8K)	53.6	23.0	53.3	28.0	59.6
ICV	26.6	17.6	51.8	22.0	49.2
AP	54.4	23.0	52.5	25.6	57.0
Many-shot (16K)	48.6	23.9	53.1	28.3	61.4
ICV	27.2	17.5	51.8	22.1	49.1
AP	49.0	23.4	52.4	25.9	58.8
Qwen3.5-0.8B					
Zero-shot	8.00	12.8	42.4	17.4	28.7
LoRA	9.20	13.7	43.4	19.4	29.5
Many-shot (1K)	34.8	17.4	47.1	13.7	51.1
ICV	11.4	13.7	43.2	17.7	28.7
AP	30.6	17.6	46.8	18.6	47.7
Many-shot (2K)	35.4	17.4	47.6	15.9	49.9
ICV	15.8	14.1	43.3	17.5	28.2
AP	29.4	17.1	46.5	18.9	43.3
Many-shot (4K)	35.4	17.5	47.9	17.8	50.6
ICV	14.6	13.4	43.4	17.9	28.8
AP	27.8	18.6	46.3	18.7	47.3
Many-shot (8K)	34.0	18.4	47.8	14.5	48.7
ICV	16.2	13.5	43.3	17.8	28.1
AP	27.6	17.4	46.6	18.3	44.4
Many-shot (16K)	30.8	19.0	48.2	17.5	48.8
ICV	16.0	13.5	43.5	17.7	28.8
AP	23.0	17.5	46.0	17.6	43.3

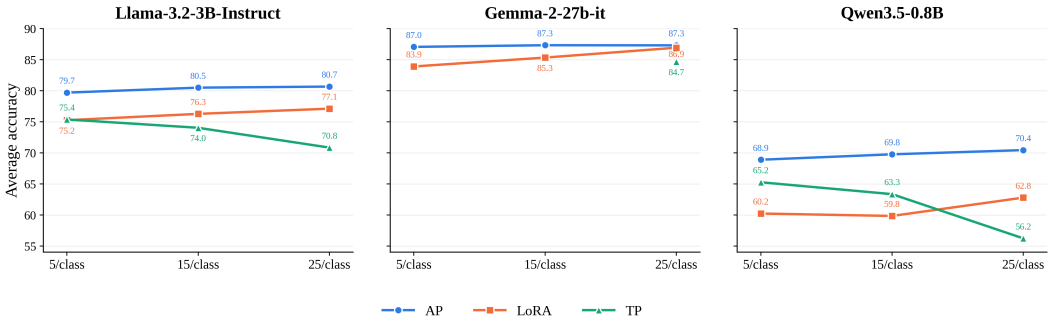


Figure 6: Effect of calibration set size on Llama-3.2-3B-Instruct, Gemma-2-27b-it, and Qwen3.5-0.8B. We report average accuracy across 12 classification benchmarks for LoRA, Thought Patching (TP), and Attention-Patching with 5, 15, and 25 examples per class.

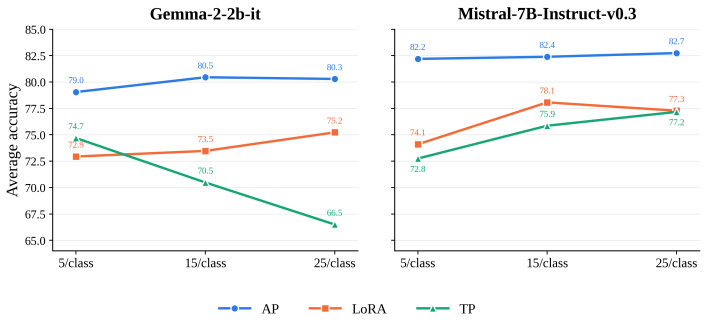


Figure 7: Effect of calibration set size on Gemma-2-2b-it and Mistral-7B-Instruct-v0.3. We report average accuracy across 12 classification benchmarks for LoRA, Thought Patching (TP), and Attention-Patching with 5, 15, and 25 examples per class.

As shown in Figures 6 and 7, Attention-Patching exhibits a clear advantage over LoRA and Thought Patching (TP) in the low-data regime. With only 5 examples per class, Attention-Patching achieves higher average accuracy than both baselines across all five evaluated models, while remaining competitive with the implicit ICL baselines reported in Tables 1 and 5. This advantage over LoRA and TP is maintained as the calibration set size increases to 15 and 25 examples per class. These results suggest that Attention-Patching effectively captures task-specific behaviors induced by demonstrations using a small calibration set, without requiring backpropagation through the language model.

D.2 Effect of Layer Selection

To examine the contribution of different layer groups, we apply the learned patches to one quarter of the decoder layers at a time and compare their performance with patching all layers. We use the task-specific patch strength selected in the main experiments.

Table 7 evaluates the contribution of different layer groups by applying the learned patches to one quarter of the decoder layers at a time. Patching all layers achieves the highest accuracy of 80.7, outperforming every individual layer group. Based on this result, we apply the learned patches to all decoder layers by default.

D.3 Sensitivity to Ridge Regularization

Table 8 examines the sensitivity of Attention-Patching to the ridge coefficient ρ on classification benchmarks. We vary ρ from $1e-3$ to $1e-1$. Based on these results, we adopt $\rho = 1e-3$ as the default setting.

Table 7: Layer-group ablation of Attention-Patching on Llama-3.2-3B-Instruct across 12 classification benchmarks. We divide the decoder layers into four contiguous groups of equal size, where Q1 and Q4 denote the earliest and latest layer groups, respectively. "All" applies patches to every decoder layer.

Group	AG	SST-2	TREC	CSQA	PIQA	SST-5	MR	MRPC	CB	COPA	CREAK	AI2SciE	Avg.
None	58.8	90.8	62.0	76.6	75.6	33.8	89.0	66.9	67.9	94.0	79.2	87.0	73.5
Q1	66.8	91.9	67.2	76.1	75.8	31.3	89.6	66.5	61.8	92.8	69.8	86.9	73.0
Q2	72.4	90.5	74.7	76.5	75.3	36.9	89.0	67.3	65.4	94.0	83.7	86.3	76.0
Q3	63.3	89.8	61.5	76.5	75.6	34.5	89.2	67.6	62.5	93.6	78.6	87.3	73.3
Q4	56.7	90.6	57.2	76.3	75.6	34.7	89.4	67.9	64.6	94.0	75.9	86.8	72.5
All	85.8	93.0	83.9	76.2	75.8	46.7	91.1	68.7	79.6	93.2	86.0	87.7	80.7

Table 8: Effect of the ridge coefficient ρ on Llama-3.2-3B-Instruct. We report average accuracy across 12 classification benchmarks.

ρ	AG	SST-2	TREC	CSQA	PIQA	SST-5	MR	MRPC	CB	COPA	CREAK	AI2SciE	Avg.
1e-3	85.8	93.0	83.9	76.2	75.8	46.7	91.1	68.7	79.6	93.2	86.0	87.7	80.7
1e-2	79.0	92.8	83.2	76.3	75.9	43.4	91.3	66.6	79.3	94.8	85.0	88.0	79.6
1e-1	59.3	92.2	84.3	76.1	75.3	38.3	90.8	66.9	70.7	95.6	85.9	87.1	76.9

E Additional Representation Analysis

E.1 Layer-wise Structure on Additional Models

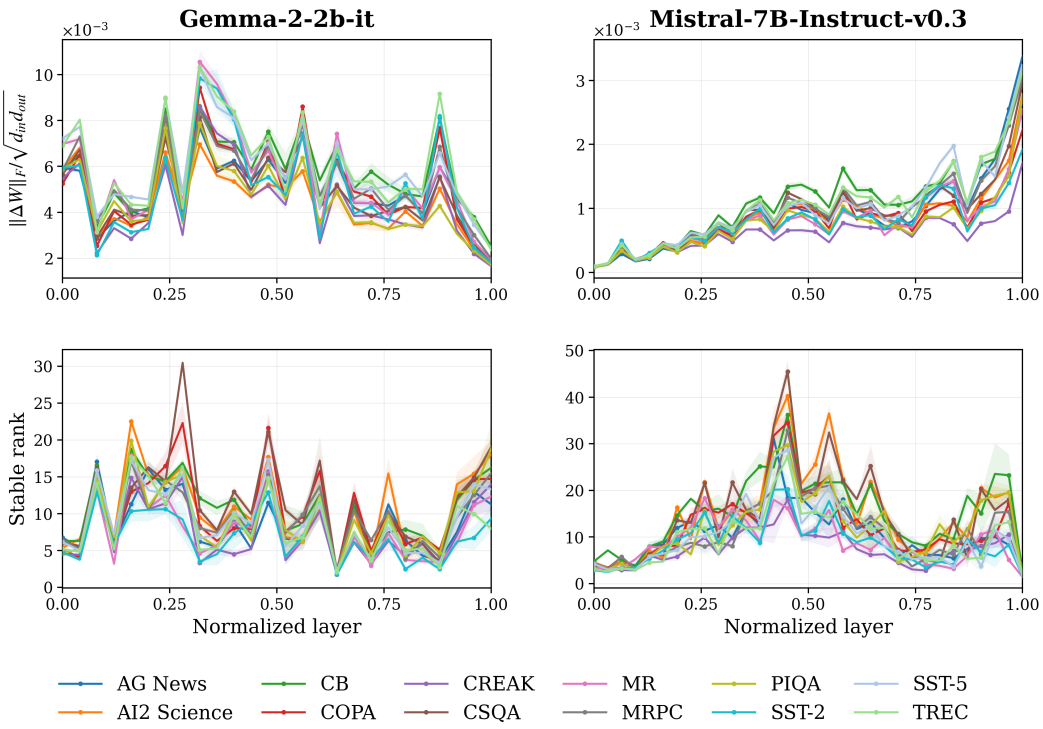


Figure 8: Layer-wise structure of the learned attention patches across classification benchmarks. The top row shows the normalized Frobenius norm of the patch weights, while the bottom row shows their stable rank. Each column corresponds to a model architecture, and each curve represents one classification task.

We provide additional representation analyses complementing the results presented in Section 5.4.

Figure 8 extends the layer-wise analysis of learned attention patches to additional model architectures, examining their magnitude and effective dimensionality through the normalized Frobenius norm and stable rank, respectively.

E.2 NMSE Comparison

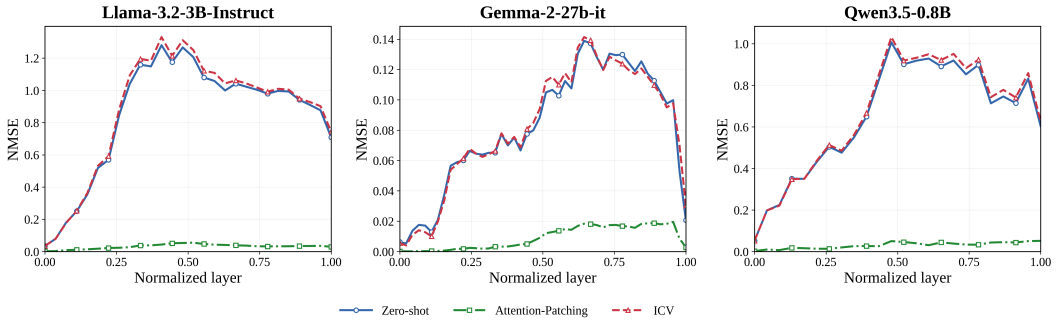


Figure 9: Layer-wise NMSE of hidden states relative to many-shot ICL, averaged over 500 GSM8K samples and context lengths. Lower values indicate greater representational similarity.

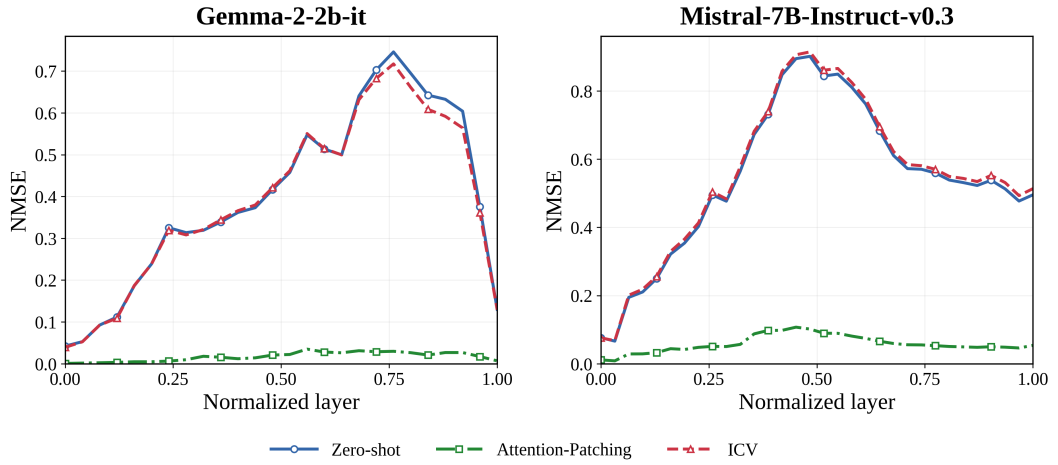


Figure 10: Layer-wise NMSE of hidden states relative to many-shot ICL, averaged over 500 GSM8K samples and context lengths. Lower values indicate greater representational similarity.

Figures 9 and 10 report the layer-wise normalized mean squared error (NMSE) between the layer-output hidden states produced by each method and those produced by explicit many-shot ICL. The results are evaluated on 500 randomly sampled GSM8K examples and averaged across the evaluated demonstration-context budgets for each model. Across layers, Attention-Patching produces hidden states that are closer to those of explicit ICL than either zero-shot inference or ICV. The reduced representational distance to explicit ICL suggests that the learned patches recover aspects of its internal activation patterns, providing further evidence that Attention-Patching captures contextual adaptation beyond the final output predictions.