When Parts are Greater Than Sums: Individual LLM Components Can Outperform Full Models

Anonymous ACL submission

Abstract

This paper studies in-context learning (ICL) by decomposing the output of large language models into the individual contributions of attention heads and MLPs (*components*). We observe curious components: good-performing ones that individually do well on a classification task, even when the model performs poorly; bad-performing ones that do much worse than chance; and label-biased components that always predict the same label. We find that component accuracies are well-correlated across different demonstration sets and perturbations of prompt templates, even when the full-model accuracy varies greatly. Based on our findings, we propose component reweighting, which learns to linearly re-scale the component activations from a few labeled examples. Given 24 labeled examples, our method improves by an average of 6.0% accuracy points over 24-shot ICL across 8 tasks on Llama-2-7B. Overall, this paper both enriches our understanding of ICL and provides a practical method for improvement by examining model internals.

1 Introduction

Rapid changes in large language models (LLMs) have popularized prompting, which guides LLMs to perform tasks with instructions or examples. Notably, in-context learning (ICL; Brown et al., 2020) adapts LLMs to a new task using only a few labeled examples without parameter updates. However, how LLMs react to the in-context examples is sometimes unintuitive (Min et al., 2022). Recently, Sclar et al. (2024) find that even for LLMs with instruction tuning (Ouyang et al., 2022) and large size, adding a space or newline in prompts can greatly affect accuracy.

We look into the LLM internals to understand what causes the surprising behavior across various ICL settings. Our work stands in contrast to prior studies on ICL, which treat LLMs as black boxes and alter either the input (Liu et al., 2022; Bertsch et al., 2024) or output (Zhao et al., 2021; Holtzman et al., 2021). We introduce a new view of ICL by decomposing the output of an LLM into the sum of individual contributions of MLPs and attention heads, denoted "components." Figure 1 reveals three types of curious components: good-performing (blue) that individually perform well or even outperform the full model, bad-performing (red) that perform below chance, and label-biased (green) that predict the same label on the entire test set. We observe these three types on Llama-2-7B, Llama-2-13B (Touvron et al., 2023), and Mistral-Instruct-7B (Jiang et al., 2023) across 8 tasks.

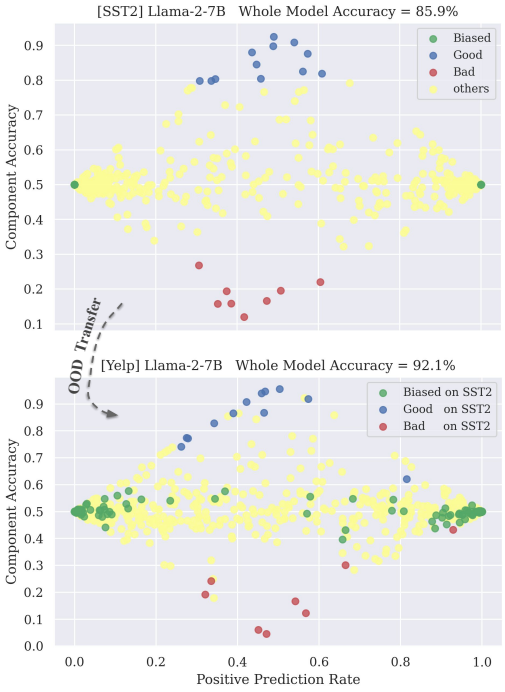

Figure 1: Each dot represents a component (attention head or MLP) under 4-shot ICL on Llama-2-7B. The x-axis shows how often a component predicts "positive" on the test set. **Up:** We discover good-performing (blue), bad-performing (red), and label-biased (green) components. **Down:** Most components identified on SST2 show similar characteristics on Yelp-polarity.

We study the sensitivity of LLM components to multiple prompts, formed by different demonstrations and templates. We also construct contrast sets of templates—pairs of similar templates that yield large differences in ICL accuracy. Despite large variance in full-model accuracy, we find that component accuracies correlate well across different demonstrations ($r = 0.79$ on average) and contrast set templates ($r = 0.57$). The top-performing components in contrast set pairs overlap and achieve decent accuracy even when the full model performs near random (Figure 2). In contrast, the component accuracies of two sampled templates are less correlated ($r = 0.34$). Further, good-performing components generalize well to out-of-distribution test sets. For instance, the top-1 component for MNLI outperforms the full Llama-2-13B model by 9.1% on MedNLI; Figure 1 also shows that components are transferrable from SST2 to Yelp. We conclude that components are relatively consistent in their behavior across prompts and datasets.

Inspired by our findings, we propose component reweighting. Compared to prior work that selects prompts to improve ICL accuracy (Rubin et al., 2022), component reweighting softly selects components by learning weights from the same few-shot examples to scale component activations. Training these weights only involves learning a linear layer—less than a minute on one CPU. Overall, component reweighting better utilizes the same labeled examples, improving over 24-shot ICL by $6.0\%, 2.2\%, 5.1\%$ on Llama-2-7B, Llama-2-13B, and Mistral-Instruct-7B, respectively, while enjoying similar inference speed as 4-shot ICL.

Finally, we study the training dynamics of components using the Pythia pretraining checkpoints (Biderman et al., 2023). During pretraining, good-performing components emerge well before the full model performs well. These findings suggest that LLMs acquire the internal ability to perform ICL early in training, but this ability only surfaces in the full model's behavior later on. Our work conducts extensive analysis of LLM internals, which motivates a practical method to improve ICL; we hope to inspire future work that further sheds light on LLM internals in order to improve performance.

2 Decomposing the Transformer in ICL

We introduce a new view of in-context learning by decomposing the Transformer architecture (Vaswani et al., 2017). Our decomposition is exact—a mathematically equivalent formula for the model's outputs—and enables us to analyze model internals without training additional parameters (unlike, e.g., probing). We first discuss what our new view offers over the standard view of ICL, and then walk through the mathematical details.

2.1 A New View of In-Context Learning

Standard view. An LLM performs in-context learning (ICL) on a task based on a few demonstrations without training, where each demonstration is a templated example (x, y) consisting of an input x and a label word y. We refer to a sequence of K demonstrations $[x_1, y_1, \ldots, x_K, y_K]$ as a *prompt*. The LLM makes predictions on a test input x_{test} conditioned on the prompt, denoted by $\arg\max_{y \in \mathcal{Y}} P(y|\textit{prompt}, x_{\text{test}})$, where $\mathcal{Y}$ is the set of possible label words in a classification task.

Our view. The residual stream of an LLM directly carries the information of the initial hidden state, every attention head, and every MLP, collectively named "components," towards the output layer. We view this information as the direct contributions[1] of components to the output logits, and derive a formula for logits, $\sum_j \mathbf{g}_j$, where $\mathbf{g}_j$ is the direct contribution of the component indexed by j. We can obtain the predictions of component j with $\arg\max_{y \in \mathcal{Y}} \mathbf{g}_j$, and then calculate its individual ICL accuracy. Specifically, we derive $\mathbf{g}_j = U \cdot C_j$ in Eq. 8 below, where U is the output embedding matrix and C_j is the post-layernorm activations of component j. We name the operation $(C_j \mapsto U \cdot C_j)$ as early decode, sharing the same spirit as nostalgebraist (2020); Geva et al. (2022), which interpret hidden representations by decoding through U. Compared to the standard view, we can directly study the behavior of individual components (Figure 2), characterizing them and scaling their contributions to the model output.

2.2 A Walkthrough of the Decomposition

A Transformer of L layers consists of a multi-headed attention (MHA) and MLP in every layer. Let $a^{(l)} \in \mathbb{R}^d$ and $m^{(l)} \in \mathbb{R}^d$ be the output of the MHA and MLP at layer l, respectively. Due to

[1] In comparison, a component has indirect contributions to the output by affecting other components in later layers.

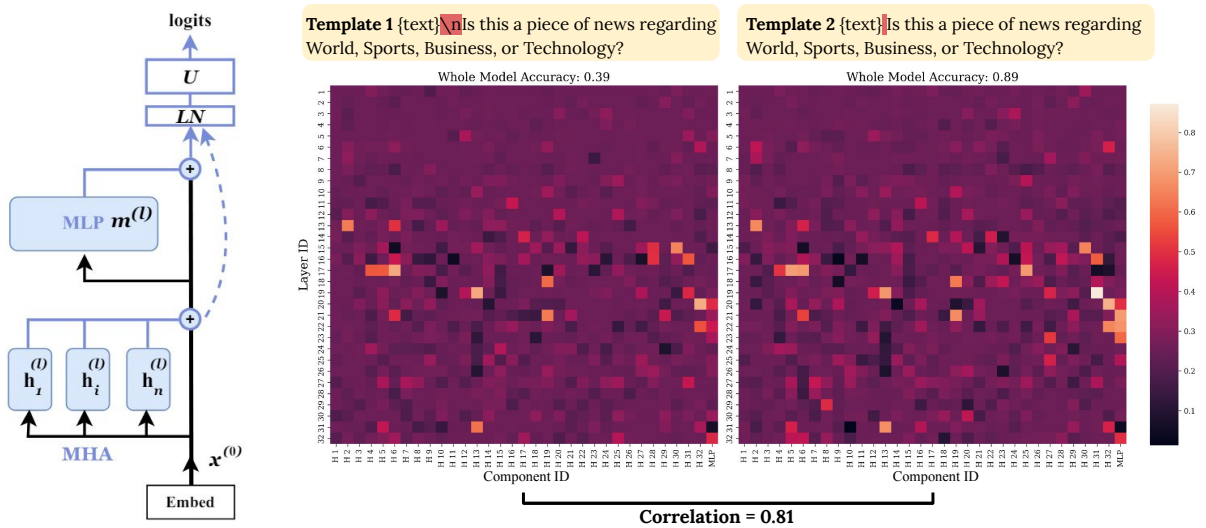

Figure 2: **Left:** Transformer decomposition. The components, MLPs and attention heads, are filled with blue and the blue lines show the flow of early decoding. **Right:** We can calculate the individual accuracy of every component after decomposition. Although a pair of templates that only differ slightly yield contrasting accuracies (0.39 vs. 0.89 on AGNews with Llama-2-7B), the accuracies of their internal components are highly correlated. The top components for Template 1 overlap with the ones for Template 2 and achieve > 0.7 accuracy despite the poor full-model accuracy.

residual connections, the hidden state $x^{(l)} \in \mathbb{R}^d$ is:

$$x^{(l)} = x^{(l-1)} + a^{(l)} + m^{(l)}, \quad (1)$$

$$x^{(L)} = x^{(0)} + \sum_{l=1}^{L} \left(a^{(l)} + m^{(l)}\right). \quad (2)$$

Note that GPT2-like LLMs apply layernorm before MHA and MLP (Radford et al., 2019); thus, layernorm is already taken into account as part of the formula for computing $a^{(l)}$ and $m^{(l)}$ (see A.3).

An MHA $a^{(l)}$ is composed of n attention heads:

$$a^{(l)} = W_o^{(l)} \cdot \text{Concat}([h_1^{(l)}, \ldots, h_n^{(l)}]) \quad (3)$$

for $h_i^{(l)} \in \mathbb{R}^{d_{\text{head}}}$ a head and $W_o^{(l)} \in \mathbb{R}^{d \times n d_{\text{head}}}$ the output projection in MHA aggregating all heads. Elhage et al. (2021) rewrite Eq. 3 by segmenting $W_o^{(l)}$ into n matrices $W_{o_i}^{(l)} \in \mathbb{R}^{d \times d_{\text{head}}}$:

$$a^{(l)} = \sum_{i=1}^{n} \left(W_{o_i}^{(l)} \cdot h_i^{(l)}\right) = \sum_{i=1}^{n} \tilde{h}_i^{(l)}, \quad (4)$$

$$\text{where } [W_{o_1}^{(l)}, \ldots, W_{o_n}^{(l)}] = W_o^{(l)} \quad (5)$$

Thus, we can treat each head as a single component adding $\tilde{h}_i^{(l)} = W_{o_i}^{(l)} \cdot h_i^{(l)}$ to the residual stream.

Finally, through the output embedding matrix $U \in \mathbb{R}^{|\text{Vocab}| \times d}$, the output logits are:

$$\begin{aligned} \text{logits} &= U \cdot \text{LN}(x^{(L)}) \\ &= U \cdot \text{LN}\left(x^{(0)} + \sum_{l=1}^{L}\sum_{i=1}^{n} \tilde{h}_i^{(l)} + \sum_{l=1}^{L} m^{(l)}\right) \\ &= U \cdot \text{LN}\left(\sum_{j=1}^{1+L\times n+L} z_j\right), \end{aligned} \quad (6)$$

where $z = [x^{(0)}, \mathbf{h}_1^{(1)}, \ldots, \mathbf{h}_n^{(L)}, m^{(1)}, \ldots, m^{(L)}]$ in Eq. 6 and we index every term in the summation with j. $\text{LN}(\cdot)$ denotes the final layernorm, specifically, RMSNorm (Zhang and Sennrich, 2019) for LLMs in our paper (see A.3). In Eq. 6, $\text{LN}(\sum_j z_j) = \frac{\sum_j z_j}{\text{RMS}(\sum_j z_j)} \odot \gamma$, where RMS denotes root mean square, $\odot$ denotes element-wise multiplication, and $\gamma \in \mathbb{R}^d$ is the affine parameters. By pre-computing $\hat{\gamma} = \frac{\gamma}{\text{RMS}(\sum_j z_j)}$, we have:

$$\text{logits} = U \cdot \left(\sum_j z_j \odot \hat{\gamma}\right) \quad (7)$$

$$= \sum_j U \cdot C_j, \text{ where } C_j = z_j \odot \hat{\gamma} \quad (8)$$

We refer to all $C_j \in \mathbb{R}^d$ as the component activations, which include the activations of attention heads and MLPs after the final layernorm.[2] Now

[2]Empirically, we find that $x^{(0)}$ has near-random ICL accuracy on all the tasks, so we omit it in the rest of the paper.

		SST2	BoolQ	QQP	WiC	RTE	MNLI	AGNews	ARC-Easy	Avg.
Llama2 7B	FULL	$75.8_{18.1}$	$69.2_{12.0}$	$61.3_{9.9}$	$52.4_{3.0}$	$\mathbf{68.9}_{3.2}$	$34.4_{1.7}$	$70.0_{19.9}$	$\mathbf{57.5}_{14.4}$	61.2
	ORACLE-T1	$\mathbf{91.7}_{0.9}$	$\mathbf{69.7}_{7.7}$	$\mathbf{67.8}_{4.3}$	$\mathbf{57.8}_{1.1}$	$64.6_{2.7}$	$\mathbf{46.3}_{3.3}$	$\mathbf{80.8}_{5.2}$	$54.5_{10.1}$	**66.6**
	ORACLE-B1	$12.1_{2.7}$	$34.1_{7.3}$	$32.5_{3.9}$	$42.9_{1.2}$	$34.7_{2.8}$	$24.1_{2.4}$	$3.0_{1.1}$	$12.7_{4.2}$	24.5
Llama2 13B	FULL	$89.0_{5.3}$	$\mathbf{77.6}_{6.8}$	$71.0_{6.8}$	$55.0_{3.8}$	$75.1_{2.3}$	$45.7_{7.9}$	$70.8_{20.6}$	$\mathbf{73.2}_{13.7}$	69.7
	ORACLE-T1	$\mathbf{92.5}_{0.6}$	$77.5_{6.0}$	$\mathbf{73.5}_{2.9}$	$\mathbf{60.4}_{1.2}$	$\mathbf{75.7}_{2.3}$	$\mathbf{56.4}_{4.7}$	$\mathbf{84.6}_{3.6}$	$73.1_{7.9}$	**74.2**
	ORACLE-B1	$8.2_{1.0}$	$27.1_{9.7}$	$31.8_{3.4}$	$39.5_{1.6}$	$27.9_{2.8}$	$18.6_{2.6}$	$1.8_{0.9}$	$5.4_{3.5}$	20.0
Mistral 7B	FULL	$90.1_{2.9}$	$\mathbf{81.3}_{2.1}$	$70.9_{7.2}$	$58.5_{4.2}$	$80.5_{1.7}$	$56.1_{5.0}$	$83.0_{5.7}$	$\mathbf{79.8}_{1.4}$	75.0
	ORACLE-T1	$\mathbf{91.9}_{0.7}$	$80.8_{2.0}$	$\mathbf{75.6}_{2.6}$	$\mathbf{60.6}_{2.2}$	$\mathbf{81.3}_{0.8}$	$\mathbf{61.5}_{3.3}$	$\mathbf{83.7}_{4.3}$	$78.5_{2.2}$	**76.7**
	ORACLE-B1	$8.1_{0.9}$	$19.5_{2.5}$	$25.8_{4.1}$	$39.3_{2.8}$	$20.0_{1.7}$	$14.6_{2.9}$	$1.8_{0.7}$	$4.6_{1.3}$	16.7
	RANDOM	50.0	50.0	50.0	50.0	50.0	33.3	25.0	25.0	41.7

Table 1: $\{3, 4\}$-shot ICL accuracy of 8 tasks and the average accuracy (**Avg.**). We run 15 prompts for each task (see §3) and report the mean accuracy and standard deviation. We show the existence of good components (ORACLE-T1) inside LLMs that individually perform on par with the full model (FULL) on diverse tasks. Similarly, there exist bad components (ORACLE-B1) that perform substantially below chance (RANDOM).

that we have broken down the Transformer output into simple additions in Eq. 8, we can easily analyze the direct contribution of each component to the logits through the residual stream, $\mathbf{g}_j = U \cdot C_j$.

In ICL, we only need to do the decomposition when LLMs start to generate, i.e, when processing the last token of the input. The computations on the other tokens are the same as the standard ICL. In all our experiments, we use single-token label words. We use multiple templates from Bach et al. (2022) that cover diverse label words for each task.

3 Characterizing Components for ICL

We conduct in-context learning across 8 classification tasks on 3 LLMs: Llama-2-7B, Llama-2-13B, and Mistral-Instruct-7B. ICL is sensitive to prompts, so we randomly sample 5 disjoint sets of demonstrations formatted with 3 templates and report the standard deviation across the 15 runs. To avoid majority and recency biases (Zhao et al., 2021), each prompt consists of the same number of demonstrations from every class in shuffled order. We use $K = 3$ demonstrations for 3-way classification tasks and $K = 4$ for the other tasks. Except for §5.1, we refer to $K = \{3, 4\}$ without further notice. We sample 2000 examples with balanced labels as the test set for every task. Please see A.1 for details about the tasks and templates.

3.1 Good and Bad-Performing Components

Across all the tasks and LLMs, we observe good-performing components that perform well or even outperform the full model and bad-performing components, which individually perform much worse than chance (blue and red dots in Figure 1, respectively). Table 1 compares the full model (FULL) with the top-1 (ORACLE-T1) and bottom-1 (ORACLE-B1) components selected on the test set. On average, ORACLE-T1 outperforms FULL by $5.4\%, 4.5\%, 1.7\%$ on Llama-2-7B, Llama-2-13B, and Mistral-Instruct-7B, respectively; ORACLE-B1 underperforms random guessing (RANDOM) by $17.2\%, 20.7\%, 25.0\%$, respectively.

3.2 Label-Biased Components

Besides good and bad-performing components, we also observe label-biased components, which predict a certain label on the entire test set (the green dots in Figure 1). These components exist in all the tasks and LLMs we study, accounting for $29.1\%, 26.4\%, 22.8\%$ of components on average in Llama-2-7B, Llama-2-13B, and Mistral-Instruct-7B, respectively (see Table 6). When we prompt the model with all demonstrations of positive labels, a biased component still insists on predicting "negative" on the entire test set, and vice versa (A.2).

4 Transferability of Components

We observe moderate to high component transferability across demonstrations, minimally contrastive templates, and data distributions, whereas there is little transferability across randomly sampled templates. Our decomposition uncovers hidden abilities of individual components when the full model performs poorly.

4.1 Transfer across Prompt Variants

We first measure the agreement in component accuracies between (1) two disjoint sets of demonstrations with a fixed template, (2) two randomly sampled templates with fixed demonstrations, and

		SST2	BoolQ	QQP	AGNews	ARC
Corr	(1) Demo	0.81	0.84	0.60	0.89	0.88
	(2) Temp	0.40	0.16	0.03	0.68	0.44
	(3) Cst T	0.72	0.63	0.23	0.82	0.46
IoU	(1) Demo	0.36	0.74	0.27	0.63	0.70
	(2) Temp	0.12	0.01	0.01	0.20	0.20
	(3) Cst T	0.40	0.23	0.02	0.36	0.45

Table 2: The average correlation and IoU between (1) two random sets of demonstrations, (2) two random templates, and (3) two minimally contrastive templates.

(3) two minimally-contrastive templates with fixed demonstrations. Recall that we have 5 sets of demonstrations and 3 templates in total (§3); here, we calculate the average agreement between every pair. For (3), we construct contrast sets (Gardner et al., 2020) by minimally editing the worst-performing template out of the 3 templates into a good template, which yields at least 10% improvement in average accuracy. Our edits include adding a space, removing a newline, or changing label words (see Table 8). We use two metrics to measure the agreement between each pair: Pearson correlation of the accuracies of all components and the intersection over union (IoU) on the sets of top-5 components, which measures whether the top-performing components of the pair overlap.

Table 2 summarizes the results on Llama-2-7B. A.6 shows similar findings on other models. (1) The accuracies of the internal components are highly consistent across different choices of demonstrations, having strong correlations and an average of 0.54 IoU. (2) The components have much weaker agreement across randomly sampled templates, having a near 0 IoU on BoolQ and QQP. (3) Nevertheless, there is agreement between minimally contrastive templates (Cst T), with an average correlation of 0.572 across tasks, despite contrasting full-model accuracy. For example, Figure 2 demonstrates that full-model accuracy changes dramatically (39% vs 89%) in a minimal pair of templates, but internal components have a high correlation of 0.81 and the pair shares top-performing components. Combining (2) and (3) suggests components behave similarly on similar templates, but this similarity decreases as the templates diverge.

4.2 Transfer to Out-of-Distribution Test Sets

We further study whether the best component selected on the test set can still perform well on an out-of-distribution (OOD) test set. We name this method, which uses a single component to make predictions, as TRANSFER-1. Specifically, we study component transferability from SST2 to Yelp-polarity, MNLI to MedNLI, and BoolQ to BoolQ Contrast Set. We compare TRANSFER-1 with using the full model (FULL) on the OOD test sets. To understand the best possible TRANSFER-1 accuracy, we also report the best component accuracy directly selected on the OOD set, ORACLE-1.

		Yelp-polarity	MedNLI	BoolQ Cst
Llama2 7B	FULL	$84.7_{15.4}$	$34.3_{1.7}$	$\mathbf{64.9}_{9.8}$
	TRANSFER-1	$\mathbf{94.9}_{3.1}$	$\mathbf{42.6}_{4.7}$	$64.3_{7.9}$
	ORACLE-1	$96.9_{0.7}$	$48.8_{2.3}$	$66.2_{5.7}$
Llama2 13B	FULL	$95.9_{1.4}$	$46.8_{9.6}$	$72.0_{7.6}$
	TRANSFER-1	$\mathbf{96.0}_{1.8}$	$\mathbf{55.9}_{4.0}$	$\mathbf{72.3}_{6.5}$
	ORACLE-1	$97.1_{0.4}$	$57.0_{3.7}$	$73.0_{6.1}$
Mistral 7B	FULL	$\mathbf{97.0}_{0.5}$	$57.3_{5.7}$	$\mathbf{74.6}_{3.5}$
	TRANSFER-1	$95.6_{1.6}$	$\mathbf{61.9}_{4.8}$	$73.7_{3.7}$
	ORACLE-1	$97.1_{0.4}$	$62.7_{4.1}$	$74.5_{3.6}$

Table 3: The average ICL accuracy and standard deviation on OOD test sets. The components selected on the in-distribution test sets (TRANSFER-1) can transfer to OOD sets, performing similarly to the oracle components (ORACLE-1) directly selected on the OOD sets.

Table 3 shows that TRANSFER-1 performs very close to ORACLE-1 overall, suggesting that the top-performing components are consistent over data distribution. Moreover, TRANSFER-1 sometimes substantially outperforms FULL, especially on MedNLI (random: 33.3%), suggesting that when LLMs perform poorly in ICL, it is likely that some internal components can do much better.

4.3 Transfer between Two Opposite Tasks

We conduct a case study of component transferability across instructions using Task069 and Task070 of Super-NaturalInstructions (Wang et al., 2022). The instruction for Task 069 asks for correct answers, while Task070 asks for incorrect ones ("pick the one that makes less sense;" Figure 6). The examples for the two tasks are not parallel.

We find that Mistral-Instruct-7B achieves good accuracy across 15 runs on Task069 (76.8 ± 2.4), but below chance on Task070 (40.6 ± 5.4). We observe a strong negative correlation (-0.60 on average) between accuracies of every pair of runs from the two tasks. The worst-performing components in Task069 become the top-performing in Task070 and vice versa. The correlation suggests that the model has the ability to solve Task070, but misunderstands negation. Thus, we apply the

TRANSFER-1 method (§4.2) but select the *worst*-performing component from Task069 and then calculate its individual accuracy on Task070, achieving 58.7 ± 4.8 accuracy across the 15 runs, an **improvement of 18.1%** over the full model. These results suggest that components behave consistently even across tasks with opposite instructions.

5 Component Reweighting

5.1 Proposed Method

Our findings in §4 show the promising direction of selecting internal components to improve ICL. Therefore, we propose a method that reweighs components by learning a weight $w_j \in \mathbb{R}$ on every component activation C_j. Reweighting is a soft version of selection, which can be learned by gradient descent on very few examples.

Given K labeled examples, instead of using all of them as ICL demonstrations, we divide them into a demonstration set $\mathcal{D}_{\text{demo}}$ and a training set $\mathcal{D}_{\text{train}}$. We first randomly sample $K' = \{3, 4\}$ examples with balanced labels as demonstrations and use the remaining examples as $\mathcal{D}_{\text{train}}$ to train the component weights. Specifically, we can rewrite Eq. 8 as logits $= \sum_j w_j (U \cdot C_j)$, where $w_j = 1$ for all j. Because of the existence of good and bad-performing components, weighing all components equally may not be optimal. Therefore, we tune the weights $w \in \mathbb{R}^N$ of N components on $\mathcal{D}_{\text{train}}$ with cross-entropy loss and L_1 regularization, while keeping the LLM frozen:

$$\mathcal{L} = \sum_{(x,y)\in\mathcal{D}_{\text{train}}} -\log P_{rw}(y|x) + \lambda \|w\|_1, \quad (9)$$

$$P_{rw}(y|x) = \text{softmax}\left(\sum_{j=1}^{N} w_j \, (U_{\mathcal{Y}} \cdot C_j)\right)_y,$$

where $U_{\mathcal{Y}} \in \mathbb{R}^{|\mathcal{Y}| \times d}$ is a submatrix of U that comprises the output embeddings of label words, P_{rw} is the probability distribution of the LLM after reweighting, and λ is the hyperparameter of the L_1 loss to encourage sparsity on the component weights. We obtain the activations $\{C_j\}_{j=1}^N$ of all components in one K'-shot forward pass, computed on the prompt derived from $\mathcal{D}_{\text{demo}}$, followed by x. Our method scales each component's direct contributions to the logits ($U_{\mathcal{Y}} \cdot C_j \in \mathbb{R}^{|\mathcal{Y}|}$) by w_j. In practice, we cache these contributions on the entire training set as input features to the linear layer w, which allows us to discard the entire LLM while training w (line 9 and 13 in Algorithm 1), saving tremendous training time and GPU memory. The cache only requires $O(|\mathcal{Y}| \times N \times |\mathcal{D}_{\text{train}}|)$ space. At inference, the overhead of our method over K'-shot ICL is to early decode N components and apply the learned weights, i.e., $\sum_{j=1}^N w_j \, (U_{\mathcal{Y}} \cdot C_j)$. As both $|\mathcal{Y}|$ and N are small[3], the overhead is negligible compared to the computation of the LLM itself.

Algorithm 1 Component Reweighting

1: **Input:** K labeled examples, a test set $\mathcal{D}_{\text{test}}$, a set of label words $\mathcal{Y}$, an LLM $\mathcal{M}$, the number of components N
2: **Output:** $\mathcal{Z}$, the predictions of $\mathcal{M}$ on $\mathcal{D}_{\text{test}}$
3: Split K examples into a *prompt* consists of K' demonstrations and a training set $\mathcal{D}_{\text{train}}$ of $K - K'$ examples
4: $U_{\mathcal{Y}} \leftarrow$ concatenate the output embeddings of $\mathcal{Y}$ in $\mathcal{M}$
5: Initialize $\mathcal{G}^{\text{dev}} \leftarrow \varnothing$
6: **for** $(x, y) \in \mathcal{D}_{\text{train}}$ **do**
7: $\quad \{C_j\}_{j=1}^N \leftarrow \mathcal{M}(\textit{prompt}, x)$ ▷ K'-shot ICL
8: $\quad$ **for** $j \leftarrow 1$ to N **do**
9: $\quad\quad \mathcal{G}^{\text{dev}} \leftarrow \mathcal{G}^{\text{dev}} \cup (U_{\mathcal{Y}} \cdot C_j)$ ▷ early decode
10: $\quad$ **end for**
11: **end for**
12: Initialize $w \leftarrow [1, \ldots, 1] \in \mathbb{R}^N$
13: Train the weights w on $\mathcal{G}^{\text{dev}}$ with Eq. 9
14: Initialize $\mathcal{Z} \leftarrow \varnothing$ ▷ Start Inference
15: **for** $(x, y) \in \mathcal{D}_{\text{test}}$ **do**
16: $\quad \{C_j\}_{j=1}^N \leftarrow \mathcal{M}(\textit{prompt}, x)$ ▷ K'-shot ICL
17: $\quad$ Initialize $\mathbf{g} \leftarrow [0, \ldots, 0] \in \mathbb{R}^{|\mathcal{Y}|}$
18: $\quad$ **for** $j \leftarrow 1$ to N **do** ▷ Test-Time Overhead
19: $\quad\quad \mathbf{g} \leftarrow \mathbf{g} + w_j (U_{\mathcal{Y}} \cdot C_j)$ ▷ early decode
20: $\quad$ **end for**
21: $\quad \hat{y} \leftarrow \arg\max_{y \in \mathcal{Y}} \mathbf{g}$
22: $\quad \mathcal{Z} \leftarrow \mathcal{Z} \cup \hat{y}$
23: **end for**
24: **return** $\mathcal{Z}$

5.2 Baselines

Standard ICL. The simplest baseline is to use all the K labeled examples as demonstrations. Since the other methods use K' examples as demonstrations, we report the accuracy of standard K'-shot ICL using the same $\mathcal{D}_{\text{demo}}$ for reference.

Prompt Selection. Liu et al. (2022) improve ICL accuracy by selecting demonstrations from a pool of labeled data for each test example. Here, we select from the given K labeled examples. Following Rubin et al. (2022), we use SBERT (Reimers and Gurevych, 2019) to encode examples into sentence embeddings and select the $K' = \{3, 4\}$ nearest neighbors under cosine similarity as the demonstrations for each test example.

Calibration. As LLMs tend to predict a certain class over others, Zhao et al. (2021) reweight the output class probabilities. They use context-free

[3] $|\mathcal{Y}| \leq 4, N < 2000$ for all tasks and LLMs in this paper.

		SST2	BoolQ	QQP	WiC	RTE	MNLI	AGNews	ARC-Easy	Avg.
Llama-2-7B	STANDARD 3, 4	75.8 18.1	69.2 12.0	61.3 9.9	52.4 3.0	68.9 3.2	34.4 1.7	70.0 19.9	57.5 14.4	61.2
	STANDARD 12	77.8 19.6	**71.6** 8.0	63.6 7.8	52.5 2.4	**71.1** 2.1	37.0 2.8	69.0 20.8	**59.6** 13.9	62.8
	PROMPTS 12	73.8 19.2	69.4 10.5	62.2 6.1	53.1 2.7	65.5 1.8	35.5 1.6	59.1 28.7	58.7 11.9	59.7
	CALIB+ 12	85.1 6.0	69.2 13.6	**73.6** 6.1	55.1 5.1	70.3 2.7	45.5 7.8	77.8 12.2	58.6 14.6	66.9
	COMPRW 12	**88.5** 2.8	70.4 11.2	71.4 5.4	**56.3** 3.4	70.0 2.8	**48.3** 4.8	**87.4** 2.3	58.3 13.6	**68.8**
	STANDARD 24	77.8 19.5	71.6 7.3	66.4 5.0	53.2 3.3	**71.9** 1.5	39.9 3.6	71.1 20.0	58.3 16.2	63.8
	PROMPTS 24	74.2 20.4	68.9 10.2	62.1 4.9	53.6 1.9	64.8 0.9	36.4 1.5	57.5 30.2	58.0 12.5	59.4
	CALIB+ 24	87.6 5.0	70.3 11.9	**73.4** 5.5	55.8 4.9	70.4 2.7	46.4 6.7	78.4 11.8	**59.2** 14.4	67.7
	COMPRW 24	**90.6** 1.7	**71.7** 9.4	71.9 4.4	**57.1** 3.0	70.0 4.1	**49.8** 4.0	**88.1** 2.1	58.8 13.6	**69.8**
Llama-2-13B	STANDARD 3, 4	89.0 5.3	77.6 6.8	71.0 6.8	55.0 3.8	75.1 2.3	45.7 7.9	70.8 20.6	73.2 13.7	69.7
	STANDARD 12	**91.3** 1.9	78.1 7.4	70.5 7.3	**59.6** 2.4	74.4 3.5	55.1 6.2	84.7 7.8	71.2 16.4	73.1
	PROMPTS 12	83.8 10.2	74.9 6.6	64.6 5.7	57.0 2.1	69.5 3.5	48.1 5.4	64.4 29.6	74.2 9.3	67.1
	CALIB+ 12	89.4 3.2	**78.4** 6.1	72.1 4.1	58.1 5.1	75.3 1.9	57.3 4.5	81.5 8.7	74.7 9.3	73.3
	COMPRW 12	89.1 3.2	77.7 6.7	**72.7** 3.3	58.7 4.0	**76.2** 2.0	**60.2** 3.7	**88.1** 1.7	**76.2** 6.8	**74.9**
	STANDARD 24	**91.9** 0.6	77.7 8.2	69.5 8.5	60.6 1.6	74.7 3.3	58.2 7.0	85.8 4.4	69.1 17.7	73.5
	PROMPTS 24	81.9 13.2	75.1 5.7	64.9 4.8	57.3 1.8	69.5 1.7	49.8 5.1	65.2 28.9	74.2 9.4	67.2
	CALIB+ 24	90.7 2.1	**78.6** 6.2	73.1 4.3	**59.5** 3.2	75.9 1.9	58.4 2.8	82.0 8.4	75.2 9.1	74.2
	COMPRW 24	91.0 1.8	78.2 6.4	**74.2** 3.1	58.5 4.1	**77.1** 1.8	**62.0** 3.7	**88.8** 1.4	**76.1** 7.2	**75.7**
Mistral-Instruct-7B	STANDARD 3, 4	90.1 2.9	81.3 2.1	70.9 7.2	58.5 4.2	80.5 1.7	56.1 5.0	83.0 5.7	79.8 1.4	75.0
	STANDARD 12	91.4 0.9	81.2 2.2	67.9 8.7	57.7 2.8	79.1 1.6	57.2 3.6	85.4 3.6	77.7 5.6	74.7
	PROMPTS 12	90.3 2.5	81.1 1.9	68.7 5.8	57.1 2.7	79.1 1.6	56.7 3.2	84.9 3.0	79.0 3.0	74.6
	CALIB+ 12	**91.5** 1.6	**81.3** 1.8	**75.8** 2.6	58.3 6.6	81.0 1.3	61.9 4.7	85.4 4.0	**79.6** 1.6	76.9
	COMPRW 12	89.9 2.7	80.7 2.7	75.1 2.9	**60.0** 4.9	**81.1** 1.3	**64.7** 4.6	**87.6** 2.1	79.2 1.2	**77.3**
	STANDARD 24	91.2 1.0	80.8 2.3	65.3 8.4	57.4 4.0	75.6 1.7	56.6 6.5	85.8 4.3	68.8 16.9	72.7
	PROMPTS 24	90.5 2.6	**81.3** 2.0	68.9 5.6	57.1 2.1	79.1 1.7	57.4 3.1	86.0 2.1	78.7 3.3	74.9
	CALIB+ 24	**91.6** 1.5	80.9 2.0	76.1 2.4	59.5 5.4	81.2 0.9	62.7 4.3	85.9 3.7	**80.1** 1.2	77.2
	COMPRW 24	90.8 1.8	80.6 2.1	**76.4** 1.7	**60.7** 4.4	**81.6** 1.0	**65.3** 3.4	**88.0** 1.8	79.0 1.6	**77.8**

Table 4: ICL accuracy of 8 classification tasks and the average accuracy (**Avg.**). The number after a method denotes the number of labeled data used. We run 15 prompts for each task (5 disjoint sets of K labeled data and 3 templates) and report the mean accuracy and standard deviation. COMPRW achieves the best average accuracy in all setups.

inputs, such as "N/A", to calibrate the probability distribution. However, Fei et al. (2023); Zhou et al. (2023) find context-free inputs sometimes ineffective, because in-domain context is important for calibration. Thus, we introduce CALIB+, which calibrates the original probabilities $\mathbf{p} \in \mathbb{R}^{|\mathcal{Y}|}$ with a training set of in-distribution labeled examples, $\mathcal{D}_{\text{train}}$. We train the calibration weights $\mathbf{v} \in \mathbb{R}^{|\mathcal{Y}|}$ on $\mathcal{D}_{\text{train}}$ with cross-entropy loss and obtain the calibrated probabilities $\hat{\mathbf{p}} = \text{softmax}(\mathbf{v} \cdot \mathbf{p})$. For direct comparisons, we split the K examples into the same $\mathcal{D}_{\text{demo}}$ and $\mathcal{D}_{\text{train}}$ sets as component reweighting for CALIB+, where $|\mathcal{D}_{\text{demo}}| = K'$. We include the training details of both methods in A.4.

5.3 Results

We set $K = \{12, 24\}$ following common practices. Table 4 compares our component reweighting (COMPRW) with standard ICL (STANDARD), prompt selection (PROMPTS), and calibration (CALIB+). First, we find that simply increasing the number of demonstrations from 4 to 24 has limited improvements in ICL accuracy, while the longer prompt greatly increases the inference time. For example, on Llama-2-7B, STANDARD 24 only improves the average accuracy by 2.6% over STANDARD 3, 4 and the accuracy even decreases on Mistral-Instruct. Second, PROMPTS performs the worst in most setups, likely because it is hard to find similar examples from a small pool of K examples, and a bad selection induces majority label biases. Third, both calibration (CALIB+) and component reweighting (COMPRW) achieve substantially better accuracy than STANDARD 3, 4 with little test-time overhead. Overall, COMPRW achieves the best average accuracy in all setups, outperforming STANDARD 12 by 6.0%, 1.8%, 2.6% on Llama-2-7B, Llama-2-13B, and Mistral-Instruct-7B, respectively, and outperforming STANDARD 24 by 6.0%, 2.2%, 5.1%. We run one-tailed paired t-tests comparing COMPRW with CALIB+ and find that p-values < 0.05 in all 6 setups (see Table 5), showing that COMPRW performs significantly better CALIB+.

6 When Do Good Components Emerge?

We study the dynamics of components during pretraining by monitoring their accuracies on 32 checkpoints of Pythia-6.9B, uniformly sampled from the first to the last checkpoint. For each checkpoint, we run 4-shot ICL on AGNews with 3 templates $\times$ 3 sets of demonstrations. Figure 3 shows the average accuracy shaded by the standard deviation.

While the full model (green) fluctuates and has a large variance across prompts, the top-1 components (solid blue) achieve good accuracy at an early step and plateaus quickly. We also backtrack the top-1 components of different prompts at the last checkpoint (dashed blue), monitoring how they perform on average during pretrain. We observe that they are not the top components at the early stage (there are gaps between the two blue lines before the $75k$ steps), but start to perform steadily well from the middle stage. Our findings also hold on SST2 and Pythia-1.4B (Figure 7 in the appendix), suggesting that the model's ability to do a task emerges before it is apparent from the full model.

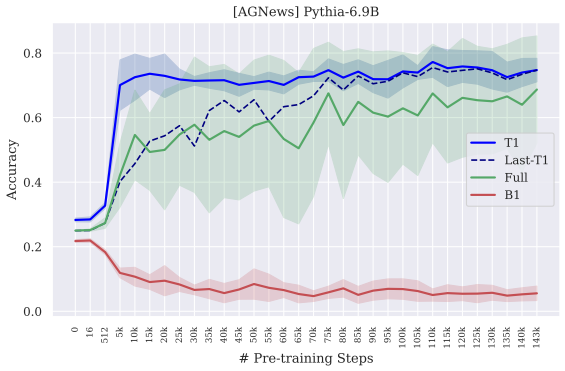

Figure 3: The ICL accuracy of the full model (green) fluctuates greatly during pretraining. However, good-performing components (T1) emerge in the early steps.

7 Related Work

Improving ICL. Prior work shows that ICL performance varies greatly across different choices of demonstrations and templates (Zhao et al., 2021; Lu et al., 2022; Sclar et al., 2024; Voronov et al., 2024). While several approaches, such as prompt selection (Liu et al., 2022; Chang and Jia, 2023), template ensemble (Voronov et al., 2024), and many-shot ICL (Agarwal et al., 2024), address this issue, they treat LLMs like black boxes without studying the internals. Besides, they usually greatly increase inference time or require a large set of labeled data, which deviates from true few-shot learning (Perez et al., 2021). In comparison, our paper studies this problem by looking inside the LLMs. Rather than selecting prompts, we select components in a soft, learnable way. Our method only requires $\{12, 24\}$ examples and has little computation overhead over 4-shot ICL at inference.

Components Interpretation. Components interpretation aims to study the function of different components in a trained model (Csordás et al., 2021; Elhage et al., 2021; Shah et al., 2024), where components considered in prior work include neurons (Radford et al., 2017; Gurnee et al., 2023), attention head (Voita et al., 2019; Olsson et al., 2022), and MLPs (Geva et al., 2021). To analyze the components, pruning (Karnin, 1990), probing (Alain and Bengio, 2017), and early decoding (nostalgebraist, 2020; Geva et al., 2022) are widely used techniques. For example, Michel et al. (2019) prune away a large percentage of attention heads in Transformer models and show that only a few heads are critical to performance at test time. Li et al. (2024) train a linear probe on every attention head to discover the "truthful heads" inside LLMs. Early decoding interprets the investigated components in the textual space by projecting them through the output embedding matrix. Our model decomposition method is based on early decoding and we share some similarities with prior work (Yu et al., 2023; Merullo et al., 2024; Liu et al., 2023; Wang et al., 2023), especially in discovering individual components that perform well. Our contributions lie in providing a new view of ICL via decomposition, revealing the transferability of components across diverse ICL settings.

8 Conclusion

We introduce a new perspective of ICL via decomposing the model output into the sum of individual contributions of components. We then identify three types of component characteristics across 3 LLMs and 8 classification tasks. Our extensive analyses reveal consistency in component accuracy across prompts and suggest the promising direction of improving ICL by selecting components. To this end, we propose component reweighting, which learns to scale components differently on few-shot examples. Our method achieves the best average accuracy compared to prior methods. We hope this work can deepen our grasp of LLMs while motivating more methods for practical use.

9 Limitations

Our component reweighting method requires a small set of labeled data $\mathcal{D}_{\text{train}}$ to train the component weights w. However, we believe it is not unreasonable to have at least $K = 12$ labeled examples in total and we compare with baselines using the same K examples. On the other hand, we do not compare with fine-tuning-based baselines, such as LM-BFF(Gao et al., 2021) and LoRA (Hu et al., 2021), because they usually require a larger GPU memory and tend to overfit on few-shot examples. Another limitation is that we only experiment with classification tasks for easy evaluation. We leave it for future work to generalize our method to generation tasks by doing decomposition and reweighting at every token during generation.

References

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Stephanie Chan, Ankesh Anand, Zaheer Abbas, Azade Nova, John D Co-Reyes, Eric Chu, et al. 2024. Many-shot in-context learning. *arXiv preprint arXiv:2404.11018*.

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes.

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts. *Preprint*, arXiv:2202.01279.

Amanda Bertsch, Maor Ivgi, Uri Alon, Jonathan Berant, Matthew R Gormley, and Graham Neubig. 2024. In-context learning with long-context models: An in-depth exploration. *arXiv preprint arXiv:2405.00200*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

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

Ting-Yun Chang and Robin Jia. 2023. Data curation alone can stabilize in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8123–8144, Toronto, Canada. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. 2021. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *International Conference on Learning Representations*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1:1.

Yu Fei, Yifan Hou, Zeming Chen, and Antoine Bosselut. 2023. Mitigating label biases for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14014–14031, Toronto, Canada. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages

1307–1323, Online. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

E.D. Karnin. 1990. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Sheng Liu, Lei Xing, and James Zou. 2023. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Circuit component reuse across tasks in transformer language models. In *The Twelfth International Conference on Learning Representations*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

nostalgebraist. 2020. interpreting GPT: the logit lens.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. In *Advances in Neural Information Processing Systems*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. 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), pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *ArXiv preprint*, abs/1704.01444.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Alexey Romanov and Chaitanya Shivade. 2018. Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596, Brussels, Belgium. Association for Computational Linguistics.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*.

Harshay Shah, Andrew Ilyas, and Aleksander Madry. 2024. Decomposing and editing predictions by modeling model computation. *arXiv preprint arXiv:2404.11534*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Anton Voronov, Lena Wolf, and Max Ryabinin. 2024. Mind your format: Towards consistent evaluation of in-context learning improvements. *arXiv preprint arXiv:2401.06766*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Tony Wang, Miles Kai, Kaivalya Hariharan, and Nir Shavit. 2023. Forbidden facts: An investigation of competing objectives in llama 2. In *Socially Responsible Language Modelling Research*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Qinan Yu, Jack Merullo, and Ellie Pavlick. 2023. Characterizing mechanisms for factual recall in language models. In *Proceedings of the 2023 Conference on*

Empirical Methods in Natural Language Processing, pages 9924–9959, Singapore. Association for Computational Linguistics.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine Heller, and Subhrajit Roy. 2023. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. *arXiv preprint arXiv:2309.17249*.

A Appendix

A.1 Tasks and Templates

Table 7 summarizes all the 13 datasets we use in the paper, where we construct balanced test sets by randomly sampling 2000 examples in each task. We form the prompts by concatenating demonstrations in a randomly shuffled order. To avoid the recency bias (Zhao et al., 2021), we keep shuffling the demonstrations until the last two have different labels. For minimally conservative templates (§4.1), Table 8 compares the contrast sets we construct on Llama-2-7B. For our case study on Task069 and Task070, we sample 3 templates from Sclar et al. (2024). In §4.3, we calculate the correlation between component accuracies of every pair of runs from the two tasks under a fixed template. The correlation score ($r = -0.6$) is averaged across runs and templates. Figure 6 compare the prompts of Task069 and Task070, which consist of an instruction followed by K templated demonstrations. Originally, the two tasks have $\sim 4\%$ of parallel examples. To make our task transfer challenging, we discard these overlapped examples.

A.2 Label-Biased Components

We say a component is label-biased when it always predicts a certain label on the entire test set §3.2. We focus on the most biased components in binary classification tasks, i.e., the two components that have the largest value of $(\text{logit}_0 - \text{logit}_1)$ and $(\text{logit}_1 - \text{logit}_0)$, respectively, where $\text{logit}_0 \in \mathbb{R}$ and $\text{logit}_1 \in \mathbb{R}$ are the output logits on the two classes. We name these two components as Biased Component-0 and Biased Component-1, respectively. To understand how biased these two components are, we alter the demonstrations and observe their behavior. Specifically, we consider three settings: demonstrations balanced in labels (green in Figure 5), demonstrations of all positive labels ($[1, 1, 1, 1]$; blue), and demonstrations of all negative labels ($[0, 0, 0, 0]$; red). We fix the template and sample 5 disjoint sets of demonstrations for each setting. Each dot in Figure 5 shows the components' prediction on an example. We show that both Biased Component-0 and Biased Component-1 still insist on predicting a certain label on all examples, regardless of the labels in the prompts. These extremely biased components may be related to the majority label bias of ICL observed by Zhao et al. (2021) on the full model.

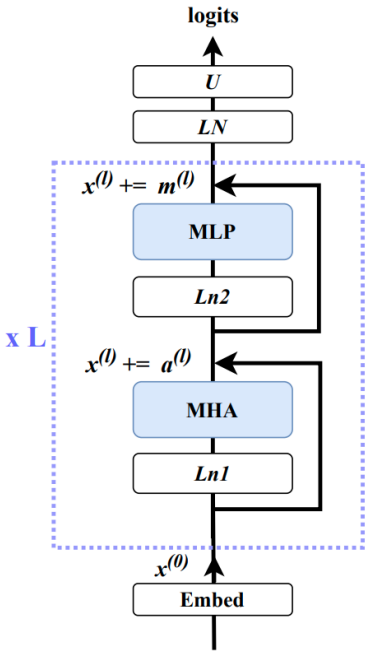

Figure 4: Transformer architecture in GPT2.

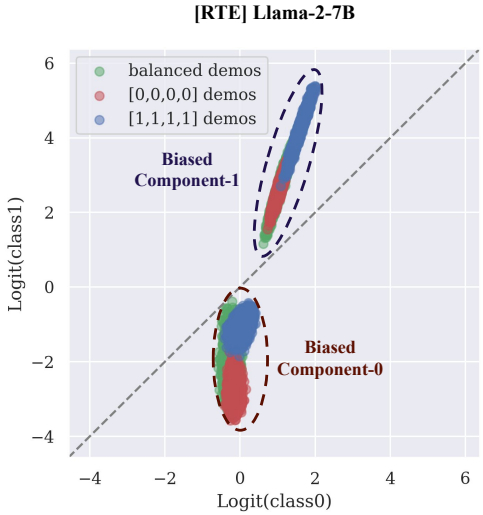

Figure 5: Each dot represents an example in the test set. The two most biased components still insist on predicting the same label on the entire test set regardless of the labels of the demonstrations.

A.3 LayerNorms

Figure 4 shows the transformer architecture in GPT2-like models. Because the layernorms inside each block are before MHA and MLP, known as Pre-LN, Eq. 1 has already taken Ln2 and Ln2 into account, and Eq. 8 only has the term for the final layernorm, $\text{LN}(\cdot)$.

Both Llama-2 and Mistral families use RMSNorm (Zhang and Sennrich, 2019), a layer normalization variant without centering and adding

bias terms. Formally, let $x \in \mathbb{R}^d$ be the input, the root mean square norm $\text{LN}(x)$ is:

$$\text{LN}(x) = \frac{x}{\text{RMS}(x)} \odot \gamma, \tag{10}$$

$$\text{RMS}(x) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}, \tag{11}$$

where $\gamma \in \mathbb{R}^d$ is the affine transform parameters and $\odot$ denotes element-wise multiplication.

A.4 Training Details and Hyperparameters

For both CompRW and Calib+methods, we train a linear layer on $\mathcal{D}_{\text{train}}$ with stochastic gradient descent. Because we do not have an additional dev set to tune the hyperparameters, we use the same hyperparameters on all the tasks and models and do early stopping based on the loss and accuracy on $\mathcal{D}_{\text{train}}$. Specifically, we set learning rate $= 0.05$ for both methods and $\lambda = 0.1$ for the L1 regularization term in CompRW. We run all our ICL experiments on a single RTX A6000 GPU (48G). Both the component reweighting and calibration training processes can be run on a single i7 CPU within a minute.

A.5 Tests of Significance

We run one-tailed paired t-tests to test whether CompRW outperforms Calib+ significantly. In Table 4, we have the results of 15 prompts for each task and 8 tasks in total. For each model, we aggregate the 120 accuracy scores of CompRW and Calib+, respectively, and then calculate the p-values. Table 5 shows that p-values < 0.05 in 6/6 setups, suggesting that CompRW performs significantly better than Calib+.

	Llama-2-7B	Llama-2-13B	Mistral-Instruct-7B
$K = 12$	0.0010	0.0002	0.0470
$K = 24$	0.0003	0.0001	0.0027

Table 5: The p-values < 0.05 in all 6 setups (3 LLMs, with $K = \{12, 24\}$ labeled examples), showing that CompRW performs significantly better than Calib+.

A.6 More Results on Transferability

In §4, we study the transferability of components across different choices of demonstrations and templates. Here, Table 9 shows the full results on all LLMs and tasks. We observe the same findings as Table 2: component accuracies agree well across randomly sampled demonstrations, but have much weaker agreements across randomly sampled templates. Because constructing minimally-contrastive templates requires non-trivial manual efforts, we only build contrast sets for 5 tasks on Llama-2-7B (shown in Table 2), where these tasks have the largest variances across templates.

	SST2	BoolQ	QQP	WiC	RTE	MNLI	AGNews	ARC-Easy
Llama-2-7B	37.8	18.9	43.4	44.2	35.4	28.2	13.4	11.5
Llama-2-13B	32.6	18.7	37.2	39.3	32.1	26.0	12.4	13.1
Mistral-Instruct-7B	31.9	14.0	32.3	32.4	27.6	20.9	14.5	8.4

Table 6: We report the average percentage of label-biased components across 15 prompts for each task. A label-biased component always predicts the same label on the entire test set.

Dataset	Task	# Classes
SST-2 (Socher et al., 2013)	Sentiment Analysis	2
Yelp-polarity (Zhang et al., 2015)	Sentiment Analysis	2
BoolQ (Clark et al., 2019)	Yes/No QA	2
BoolQ Contrast Set (Gardner et al., 2020)	Yes/No QA	2
QQP (Wang et al., 2018)	Paraphrase Identification	2
WiC (Pilehvar and Camacho-Collados, 2019)	Word Sense Disambiguation	2
RTE (Wang et al., 2018)	Natural Language Inference	2
MNLI (Williams et al., 2018)	Natural Language Inference	3
MedNLI (Romanov and Shivade, 2018)	NLI in Medical Domain	3
AGNews (Zhang et al., 2015)	Topic Classification	4
ARC-Easy (Clark et al., 2018)	Multiple-Choice QA	4
Task069 (Mishra et al., 2022; Wang et al., 2022)	Abductive NLI	2
Task070 (Mishra et al., 2022; Wang et al., 2022)	Abductive NLI	2

Table 7: Summary of all the datasets.

Task	Templates	Labels	Accuracy
SST-2	**T1** Review: {text}\n Do you think the review is positive or negative? {label}	negative/positive	50.6 ± 0.7
	T2 Review: {text} {space}\n Do you think the review is positive or negative? {label}	negative/positive	72.7 ± 6.1
BoolQ	**T1** Based on the following passage, {question}? {passage}\nAnswer: {label}	No/Yes	52.5 ± 2.0
	T2 Based on the following passage, {question}? {passage} Answer: {label}	No/Yes	66.7 ± 2.1
QQP	**T1** Are the questions "{sent1}" and "{sent2}" asking the same thing? {label}	no / yes	54.3 ± 1.1
	T2 Are the questions "{sent1}" and "{sent2}" asking the same thing? {label}	No / Yes	68.7 ± 4.1
AGNews	**T1** {text} \n Is this a piece of news regarding World, Sports, Business, or Technology? {label}	World/ Sports/ Business/ Technology	43.9 ± 8.7
	T2 {text} Is this a piece of news regarding World, Sports, Business, or Technology? {label}	World/ Sports/ Business/ Technology	88.5 ± 0.8

Table 8: We construct minimally contrastive templates that only differ slightly (colored in red) but yield large differences in 4-shot ICL accuracy on Llama-2-7B. We report the average accuracy and standard deviation across 5 ICL runs under the same template.

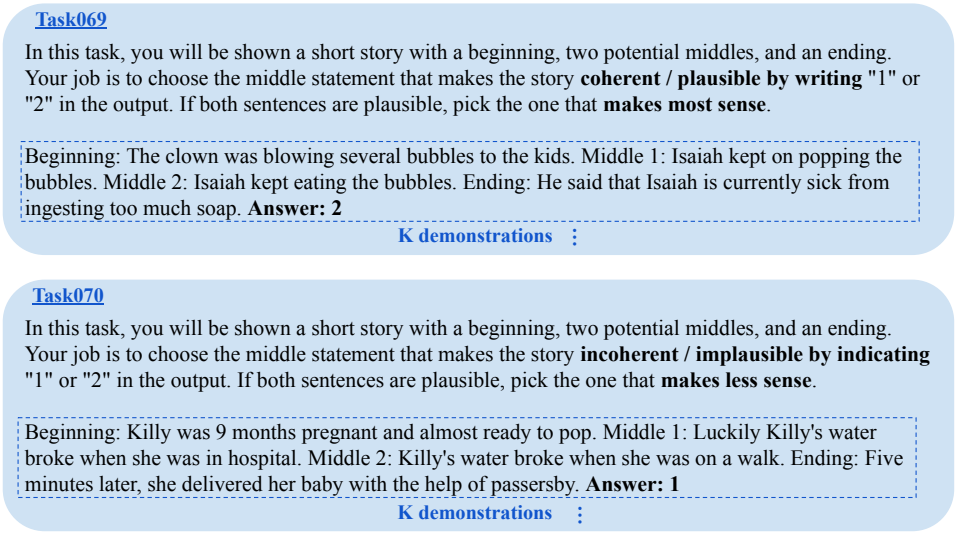

Figure 6: Comparing the prompts of Task069 and Task070. We follow the formats of Sclar et al. (2024) and prepend the task instructions before K demonstrations. We ensure that the two tasks do not have parallel examples to make the transfer experiment (§4.3) challenging.

	SST2	BoolQ	QQP	WiC	RTE	MNLI	AGNews	ARC
Correlation				***Llama-2-7B***				
(1) Demo	0.81	0.84	0.60	0.65	0.75	0.65	0.89	0.88
(2) Temp	0.40	0.16	0.03	0.15	0.19	0.09	0.68	0.44
IoU								
(1) Demo	0.36	0.74	0.27	0.21	0.53	0.24	0.63	0.70
(2) Temp	0.12	0.01	0.01	0.03	0.05	0.01	0.20	0.20
Correlation				***Llama-2-13B***				
(1) Demo	0.83	0.84	0.63	0.67	0.78	0.73	0.91	0.91
(2) Temp	0.57	0.30	0.09	0.19	0.28	0.16	0.76	0.55
IoU								
(1) Demo	0.26	0.71	0.31	0.18	0.46	0.39	0.55	0.65
(2) Temp	0.21	0.11	0.07	0.01	0.21	0.07	0.25	0.30
Correlation				***Mistral-Instruct-7B***				
(1) Demo	0.88	0.91	0.72	0.75	0.87	0.82	0.92	0.97
(2) Temp	0.58	0.44	0.19	0.26	0.40	0.30	0.77	0.60
IoU								
(1) Demo	0.39	0.59	0.27	0.29	0.50	0.45	0.68	0.80
(2) Temp	0.10	0.17	0.06	0.05	0.17	0.09	0.29	0.22

Table 9: Full results of the average correlation and IoU between (1) two random sets of demonstrations and (2) two random templates. **ARC** is short for the ARC-Easy task.

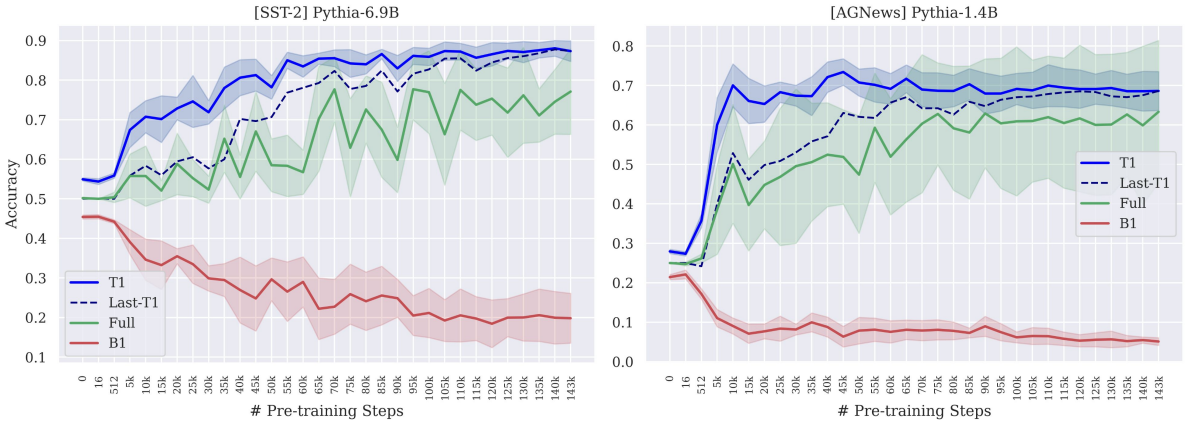

Figure 7: 4-shot ICL accuracy on different pretraining checkpoints. We compare the full model (green) with the top-1 (solid blue) and bottom-1 (red) components. The dashed blue line tracks how the top-1 components of the last checkpoint (Last-T1) perform across time.

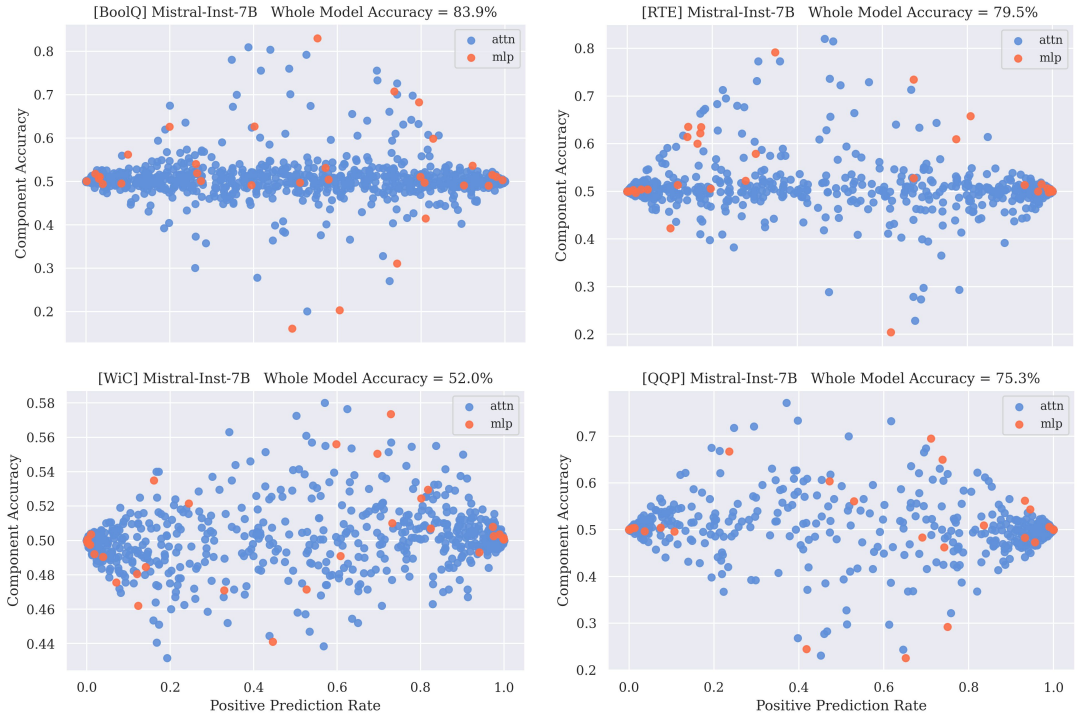

Figure 8: Each dot represents a component (attention head: blue; MLP: orange) under 4-shot ICL on Mistral-Instruct-7B. The x-axis shows how often a component predicts "positive" on the test set.