Breaking through the Learning Plateaus of In-context Learning in Transformer

Jingwen Fu[1] **Tao Yang**[1] **Yuwang Wang**[2] **Yan Lu**[3] **Nanning Zheng**[1]

Abstract

In-context learning, i.e., learning from context examples, is an impressive ability of Transformer. Training Transformers to possess this in-context learning skill is computationally intensive due to the occurrence of *learning plateaus*, which are periods within the training process where there is minimal or no enhancement in the model's in-context learning capability. To study the mechanism behind the learning plateaus, we conceptually separate a component within the model's internal representation that is exclusively affected by the model's weights. We call this the "weights component", and the remainder is identified as the "context component". By conducting meticulous and controlled experiments on synthetic tasks, we note that the persistence of learning plateaus correlates with compromised functionality of the weights component. Recognizing the impaired performance of the weights component as a fundamental behavior that drives learning plateaus, we have developed three strategies to expedite the learning of Transformers. The effectiveness of these strategies is further confirmed in natural language processing tasks. In conclusion, our research demonstrates the feasibility of cultivating a powerful in-context learning ability within AI systems in an eco-friendly manner.

1. Introduction

This paper is centered on the in-context learning ability of Transformer, which stands as one of the most significant abilities for current applications of Transformer models (Brown et al., 2020; Dong et al., 2022; Shin et al., 2022;

[1]National Key Laboratory of Human-Machine Hybrid Augmented Intelligence, National Engineering Research Center for Visual Information and Applications, and Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University [2]Tsinghua University [3]Microsoft Research Asia. Correspondence to: Nanning Zheng <nnzheng@mail.xjtu.edu.cn>.

Min et al., 2021). Fig. 1A gives examples of the in-context learning tasks. The in-context learning ability has a confusing property that it is emergent when increasing training FLOPs (Wei et al., 2022). By exploring the learning dynamic of in-context learning, previous works (Edelman et al., 2022; Michaud et al., 2023; Kirsch et al., 2022; Singh et al., 2023; Reddy, 2023) discover that the Transformer learns the in-context learning ability abruptly. We summarize the phenomenon during the learning process as plateaus and transition pattern, illustrated in Fig. 1B. This pattern indicates that there is negligible or no enhancement in the in-context learning ability during the initial learning period, which we call the learning plateaus, and these plateaus are then followed by rapid and substantial gains, which are referred to as the transition process. Often, the strategy to shorten learning plateaus involves expanding the model's scale, which consequently demands greater computational resources and energy consumption. *In this paper, our objective is to investigate the possibility of overcoming the learning plateaus without scaling the model's size.* Addressing this challenge could lead to a novel approach to creating environmentally sustainable intelligent systems. Since learning plateaus are not typically observed in conventional supervised learning, concentrating on the difference may be crucial for unraveling the mechanisms behind the learning plateaus.

Weights and context components A principal distinction between in-context learning and traditional supervised learning lies in the fact that in-context learning outcomes are shaped by both the model's parameters and the specific context examples provided. In traditional supervised learning, for a given input sample $\mathbf{x}_p$ with its corresponding label y_p, the goal is to find a parameterized function $f_{\mathbf{w}}$ with weights $\mathbf{w}$, such that the prediction $f_{\mathbf{w}}(\mathbf{x}_p)$ is equal to the label y_p. In this scenario, the weights $\mathbf{w}$ hold all the information needed to perform the task at hand. Conversely, within the in-context learning paradigm, there is an additional source of information, which is the context examples s_c. Hence, the prediction model is represented as $f_{\mathbf{w},s_c}(\mathbf{x}_p)$, indicating that both the weights and the context examples have the potential to affect the prediction outcome. To examine how the weights and in-context examples impact the prediction, we assume that the function $f_{\mathbf{w},s_c}(x)$ can be **conceptual decomposite** into $f_{\mathbf{w},s_c}(x_p) =$
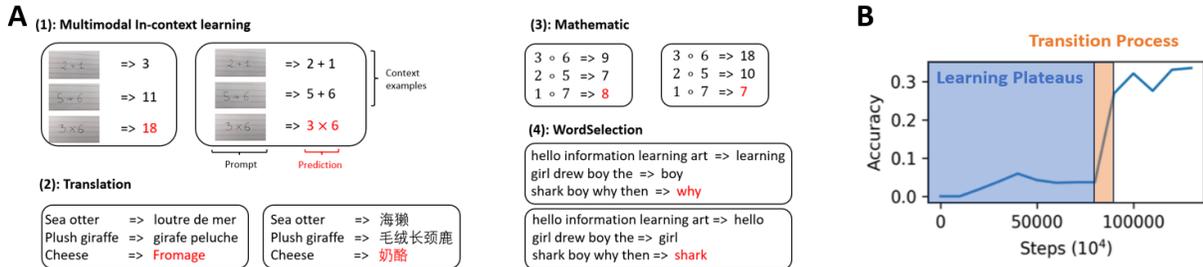
Figure 1. **A: Examples of the in-context learning tasks.** Examples of (1) comes from Alayrac et al. (2022), Examples of (2),(3),(4) come from Brown et al. (2020). **B: Illustration of learning plateaus and transition pattern.** We evaluate the in-context learning ability of Pythia 13B model (Biderman et al., 2023) trained on pile dataset (Gao et al., 2020) using WordSelection task (Detail in Appendix B) during the training process.

$g_{comb}(g_{weights}(\mathbf{x}_p), g_{context}(s_c))$. This decomposition allows us to separate the component $g_{weights}(\cdot)$, which is part of the Transformer solely dependent on the weights, from the component $g_{context}(\cdot)$, which is influenced by both the weights and the context examples. We refer to $g_{weights}(\cdot)$ as the weights component and $g_{context}(\cdot)$ as the context component. Owing to the design of the Transformer's architecture, there is an interaction between $g_{weights}(\cdot)$ and $g_{context}(\cdot)$. **Importantly**, conceptual decomposition implies that the decomposition isn't physical; rather, it's solely for analytical purposes.

Key observations Owing to the practical challenges of directly investigating the components, as well as the inability to regulate the complexity of these tasks, we depend on a synthetic task to conduct controlled experiments. Through conducting experiments involving tasks of varying difficulty and monitoring the performance of both the weights and the context components, we have made a critical observation: *the duration of learning plateaus are often associated with a dysfunction of the weights component.*

Break through the learning plateaus. Drawing from our observations, we consider the dysfunction of the weights component during learning to be the primary cause of the extended learning plateaus. Based on this, we suggest three methods for enhancing the weights component and all these methods can mitigate the learning plateaus. The effectiveness of these methods is also verified in NLP tasks.

Contributions Our main contributions can be summarized as follows: **a)** We give formulations of weights and context components with a new synthetic task that enables the study of the mechanism behind learning plateaus. **b)** We study the learning process of synthetic tasks with different complexity. The experiments reveal the relation between the weights component and the learning plateaus. **c)** To further verify the causal relation between the learning plateaus and the weights component, we propose different methods to improve the weights component and we observe the mitigat-

ing of the learning plateaus. The discoveries are verified in the NLP task.

2. Related works

In this section, we explore the works most closely related to our study. Additional related literature can be found in Appendix C. This appendix encompasses a) an examination of the weights and context components in relation to the previously proposed division of in-weights and in-context learning, b) a review of evidence from prior studies that supports the importance of both context and weights components in practical applications, and c) a compilation of related works aimed in understanding Transformers.

Analyzing the transition phenomenon of in-context learning The emergence of in-context learning capabilities during learning has captivated numerous researchers (Olsson et al., 2022; Michaud et al., 2023; Kirsch et al., 2022; Singh et al., 2023; Reddy, 2023). Olsson et al. (2022) has found that this transition process is linked to the creation of inductive heads within Transformer models. Meanwhile, other researchers have determined that the occurrence of such a transition is influenced by the properties of the dataset (Chan et al., 2022a). Li et al. (2022); Lu et al. (2023) go further to associate the emergent capabilities seen during the learning with those that appear during the scaling of the model. Our research, however, is not centered on this transition phenomenon. We aim to delve into the learning plateaus phenomenon, which serves as a vital complement to understanding the intricacies of the transition phenomenon.

Understanding of the mechanism of in-context learning Considerable research has been devoted to this vital topic, with most prior studies focusing on understanding the mechanism of in-context learning from the perspective of algorithm implementation. For example, a number of recent papers (von Oswald et al., 2022; Dai et al., 2023; Akyürek et al., 2022) have described in-context learning as
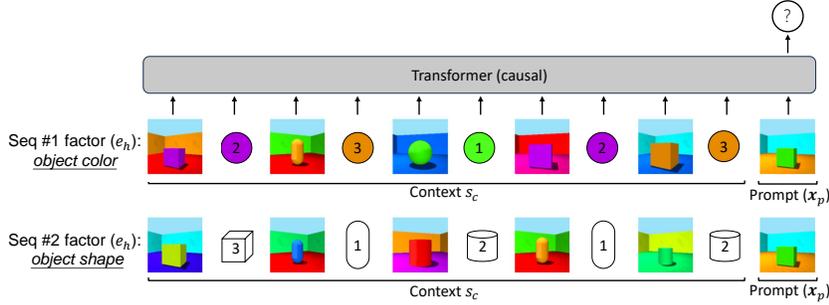
Figure 2. **Synthetic task**. In the task, Transformer is required to predict the label of $\mathbf{x}_p$ given context examples s_c. The images from the 3D Shapes dataset are synthesized based on six factors. The output factor is determined by the context. In this case, we provide two sequences of factors: "object color" and "object shape," respectively.

akin to performing gradient descent. Additional research (Li et al., 2023; Bai et al., 2023) has interpreted in-context learning in terms of algorithm implementation and choice. Our study, however, takes a novel approach by examining the in-context learning mechanism through the lens of the weights and context components, offering a distinct perspective from earlier works.

3. Experimental Design

This paper employs a synthetic task to investigate the fundamental mechanisms behind learning plateaus. The use of a synthetic task is due to the fact that the intricacies of real tasks pose challenges in monitoring and comprehending the precise factors that influence the emergence of learning plateaus.

3.1. Dataset Construction

We propose a task using the Shapes3D (Kim & Mnih, 2018) dataset for a more controllable study. The experimental setting is shown in Fig. 2. Specifically, given a sequence of image and label pairs as context, the task involves predicting the label of the prompt image. Each image contains six different factors: object color, object shape, object scale, background color, floor color, and pose. We denote the factor as e and the factor value of factor e as $v^{(e)}$. For each sequence, we randomly choose a factor to generate the labels of the images, referring to this factor as the **hidden factor** e_h for this sequence. For the two context sequences in Fig. 2, the hidden factor of Seq #1 is object color, and the correct label for the prompt image is 1 (object color is green). In Seq #2, for the same prompt image, the correct label is 3 (the object shape is a cube).

We give a formal formulation of the data generation process below.

Notations We denote $\mathbf{x}_p$ as the prompt example with ground truth label y_p. The context examples are $s_c =$

$\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\}$. The prediction of the model is denoted as $f_{\mathbf{w}, s_c}(\mathbf{x}_p)$. We denote the factor values of $\mathbf{x}$ as $v_\mathbf{x}$ and the corresponding factor value for factor e as $v_\mathbf{x}^{(e)}$. v_p is short for $v_{\mathbf{x}_p}$. The hidden factor is denoted as e_h. We denote the mapping function as m, which maps the factor value to the corresponding label, i.e. $y_p = m(v_p^{(e_h)})$. We denote the probability as $\mathbb{P}$.

Definition 3.1. The data is generated according to the equation that

$$\mathbb{P}(\mathbf{x}_p, y_p, s_c) = \sum_{m, e_h} \mathbb{P}(m)\mathbb{P}(e_h)\mathbb{P}(\mathbf{x}_p, y, s_c | m, e_h), \quad (1)$$

where $\mathbb{P}(m), \mathbb{P}(e_h)$ are manually setted distributions and $\mathbb{P}(\mathbf{x}, y, s_c | m, e_h)$ is a fixed distribution. $\mathbb{P}(m), \mathbb{P}(e_h)$ are uniform distributions over all possible values by default. In this paper, we rely on changing $\mathbb{P}(m)$ to obtain the tasks with different complexity.

3.2. Analysis

To successfully tackle the in-context learning task, the network is required to discern the values of the six factors present in the prompt image, which relate to the weights component, as well as accurately determine the appropriate hidden factor for output based on the given contexts, which pertains to the context component. We break down the distribution $\mathbb{P}(y_p | \mathbf{x}_p, s_c)$ as follows.

Proposition 3.2. *The probability of $\mathbb{P}(y_p | \mathbf{x}_p, s_c)$ can be decomposite as:*

$$\mathbb{P}(y_p | \mathbf{x}_p, s_c) =$$
$$\sum_{v_p, m, e_h} \underbrace{\mathbb{P}(y_p | v_p, m, e_h)}_{\text{Properties of Task}} \underbrace{\mathbb{P}(v_p | \mathbf{x}_p)}_{\text{weights}} \underbrace{\mathbb{P}(e_h | s_c, m)\mathbb{P}(m | s_c)}_{\text{context}},$$
$$(2)$$

where $\mathbb{P}(v_p | \mathbf{x}_p)$ is weights related information, and $\mathbb{P}(e_h | s_c, m)\mathbb{P}(m | s_c)$ is context related information. $\mathbb{P}(y_p | v_p, m, e_h)$ is related for the properties of task, and we have $\mathbb{P}(y_p | v_p, m, e_h) = 1$ if $m(v_p^{e_h}) = y_p$ else $\mathbb{P}(y_p | v_p, m, e_h) = 0$.

3

Ideally, if we can approximate the distribution $\mathbb{P}(v_p|\mathbf{x}_p)$ and the distribution $\mathbb{P}(e_h|s_c, m)\mathbb{P}(m|s_c)$, we can obtain the distribution $\mathbb{P}(y_p|\mathbf{x}_p, s_c)$. Recall that we denote $g_{weights}(\mathbf{x}_p)$ as weights component and $g_{context}(s_c)$ as context component. Based on the decomposited results of $\mathbb{P}(y_p|\mathbf{x}_p, s_c)$, we define our expectation for the components to be good:

Definition 3.3. If $f(\cdot)$ has a good weights component in its representation, for any $\mathbf{x}$, we can infer $\mathbb{P}(v_{\mathbf{x}}|\mathbf{x})$ from $g_{weights}(\mathbf{x})$, and if it has good context component, we can infer $\mathbb{P}(e_h|s_c, m)$ from $g_{context}(s_c)$.

3.3. Evaluation framework

In practice, since we cannot find the specific form of $g_{weights}$ and $g_{context}$, such that $f_{\mathbf{w}, s_c}(x_p) = g_{comb}(g_{weights}(\mathbf{x}_p), g_{context}(s_c))$. We leverage the probe method to measure the goodness of the components in the inner representation of the Transformer as defined in Definition 3.3. We choose the layers in the Transformer that can produce the best probe results. We give the details of the probing framework in the Appendix A.3.

Probing methods and metrics   We use three metrics here. **weights comp. score**: accuracy of the probe model to predict $v_{\mathbf{x}}$ given $\mathbf{x}$, where "comp." is short for component. **context comp. score**: the accuracy of the probe model to predict e_h given $s_c, \mathbf{x}_p$. **Accuracy**: We evaluate the prediction accuracy of the prompt example as the metric to evaluate the in-context learning performance. We give 39 context examples when evaluate context comp. score and accuracy.

All the results given in this paper are evaluated in the test set by default, as the performance of the test set is more aligned with the model's performance on real situations.

3.4. Model, training and dataset detail

To simulate the auto-regression framework, we calculate the loss for the sequence $s = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}$ as:

$$\mathcal{L}(\theta, s) = \frac{1}{L}\sum_{i=1}^{L} l(f_{\mathbf{w}, s^{(i-1)}}(\mathbf{x}_i), y_i), \tag{3}$$

where $s^{(j)} \triangleq \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_j, y_j)\}$, l denotes the loss function. $\mathbf{x}$ will be tokenized by VAE (Kingma & Welling, 2013) before being passed to Transformer. Note that the weights of VAE are pretrained and fixed during the whole experiments. The training loss in the dataset S, which contains n sequence, is calculated as the average of loss over all training sequences, i.e.,

$$\mathcal{L}(\theta, S) = \frac{1}{n}\sum_{s \in S} \mathcal{L}(\theta, s). \tag{4}$$

We leave the details of the model, dataset, training design and configuration in the Appendix A.

4. Learning plateaus and weights component

Recall that we define learning plateaus as periods during the learning process in which the model experiences minimal or no improvement in performance on **test data**. Conversely, the transition process is characterized by a time span in which the model's performance rapidly enhances. Typically, a learning plateau precedes a transition process. When a learning process includes several instances of learning plateaus and transition processes, the terms "learning plateaus" and "transition process" refer by default to the first occurrence of each.

4.1. Controlling of complexity

We aim to understand when the learning plateaus will happen by controlling the complexity of the tasks. Therefore, we will present the method for controlling the complexity in this part. Recall from Definition 3.1 that the data is generated following the formula $\mathbb{P}(\mathbf{x}, y, s_c) = \sum_{m, e_h} \mathbb{P}(m)\mathbb{P}(e_h)\mathbb{P}(\mathbf{x}, y, s_c|m, e_h)$. By altering the probabilities $\mathbb{P}(m)$, we can manipulate the resulting dataset. We first give two baseline configurations:

- D_{fix}: There exsits a m_0 such that $\mathbb{P}(m_0) = 1$ and for all $m \neq m_0$, we have $\mathbb{P}(m) = 0$.

- D_{rnd}: $\mathbb{P}(m)$ is a uniform distribution over all possible values.

Clearly, introducing greater randomness in the selection of m increases the complexity of the problem. We quantify this complexity using entropy, defined as $\mathbb{H} \triangleq \sum_m -\mathbb{P}(m)\log\mathbb{P}(m)$. It is noted that $\mathbb{H}(D_{\text{fix}}) = 0$. Since our primary objective is to adjust the complexity of the in-context learning task rather than to examine the effects of various distributions of $\mathbb{P}(m)$, we simplify $\mathbb{P}(m)$ to a uniform distribution. The entropy is then managed by varying the size of the support set, that is, the number of possible mapping functions.

When considering two distinct data configurations D_1 and D_2, the notation $D_1 \Rightarrow D_2$ denotes the performance evaluation of a model on data setting D_2 after it has been trained on data setting D_1. Unless specified otherwise, we assume that the model is both trained and tested on the same data configuration.

4.2. Learning plateaus of task with different complexity

Learning Plateuas and Transition Point   The primary issue at hand involves learning plateaus and transition patterns. To begin with, we investigate the capability of the synthetic task to mimic the specific pattern depicted in Fig. 1B. Our scrutiny is directed toward the learning trajectory of the synthetic task. As demonstrated in Fig. 3A, it becomes evi-
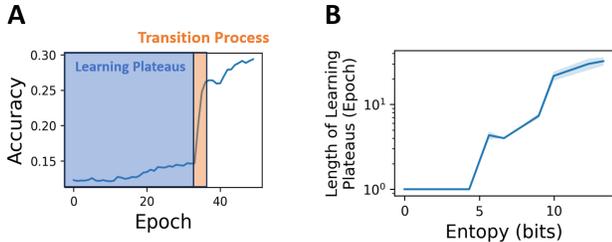
Figure 3. **Learning plateus.** **A.** We reproduce the learning plateaus and transition pattern in our synthetic task, similar to Fig 1B. **B.** The length of learning plateaus increase with the complexity of the task measured by entropy of $\mathbb{P}(m)$.

dent that the synthetic task is successful in replicating the plateaus and transition pattern observed in the actual task.

The length of learning plateaus increases when increases the complexity of task We delved into the relationship between task complexity and the duration of learning plateaus. We employed the entropy of the mapping function m's distribution as an indicator of task complexity. To pinpoint the transition process, we identified the first epoch at which the model achieved a test accuracy greater than 0.17. This threshold was selected because the model's accuracy remains below 0.17 before reaching the transition process and rises above this thereafter. As anticipated, more complex tasks necessitate longer learning plateaus (Fig. 3B). However, the relationship between the length of the plateaus and the entropy is not linear. With every unit increase in entropy, the extension of the learning plateau is marginal when the entropy is either low or relatively high; conversely, the growth in plateau length is more pronounced at intermediate levels of entropy.

4.3. Dysfunction of weights component

In the preceding section, we analyzed the plateaus-and-transition pattern in in-context learning concerning task complexity. In this section, our objective is to delve deeper into the role that internal mechanisms—specifically, the quality of the weights and context components—have in influencing the plateaus-and-transition pattern.

Confusing pattern of weights component We executed the task under two conditions: $D_{\text{fix}} \Rightarrow D_{\text{fix}}$ and $D_{\text{rnd}} \Rightarrow D_{\text{rnd}}$, with the outcomes presented in Fig. 4A. As expected, in the simpler scenario of $D_{\text{fix}} \Rightarrow D_{\text{fix}}$, both the context and weights components exhibit improvement throughout the learning process, leading to enhanced in-context learning performance. However, in the more challenging setting of $D_{\text{rnd}} \Rightarrow D_{\text{rnd}}$, the weights component deteriorates over time, with its score remaining below the initial value for the entire duration of the learning process.

This differs from the context component, which improves with the rise in in-context learning performance. We refer to the situation where the weights component score falls below the starting value as a **dysfunction** of the weights component. This outcome is intriguing because training has no effect in improving the weights component. To gain better insight into this phenomenon, we carried out additional experiments across varying levels of task complexity. The model was trained for 50 epochs on these tasks, and we monitored the weights component score post-training. We found that the weights component score gradually declines as the entropy increases, eventually stabilizing at around 0.8, as shown in Fig. 4B.

Weight component degradation is linked to duration learning plateaus. Our primary concern is the duration of learning plateaus, and we seek to comprehend its connection to the in-weights component. To investigate this, we graphed the relationship between learning plateau length and weights component score after 50 epochs, as shown in Fig. 4C. Our analysis reveals an approximately linear correlation between the weights component score and the learning plateau duration. Short learning plateaus occur when there is a significant improvement in the in-weights component score. Conversely, long learning plateaus arise when the weights component is dysfunctioning or on the edge of dysfunction, that is when the weights component score is at or below the initial value.

Why the weights component is related to learning plateaus. We are attempting to comprehend this phenomenon, yet theoretically analyzing the learning process of a multilayer transformer on intricate data proves to be a formidable challenge. Recent theoretical studies (Tian et al., 2023; Deora et al., 2023; Huang et al., 2023) focus on the learning dynamics of Transformers with one or two layers using simple datasets. Given these limitations, we propose a more attainable, albeit weaker, construction analysis in Appendix E. Our approach is grounded in the notion that if a model with a good weights component can enhance its in-context learning capabilities with just a few additional parameters, then the weights component must be pivotal for achieving in-context learning prowess. To test this idea, we hypothesize that the in-weights component has been perfectly learned within a specific layer of the Transformer. Our findings reveal that by adding at most three extra Transformer layers specifically tailored for processing contextual information, the model demonstrates significant in-context learning performance. These outcomes suggest that in-context learning abilities are more readily achieved when the weights component is well-optimized.
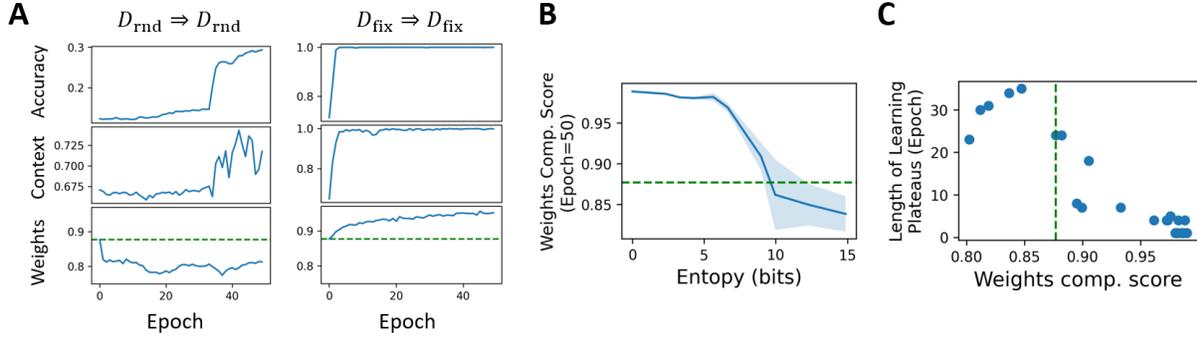
Figure 4. Weights component and learning plateaus. **A.** The weights component score is increasing under $D_{\text{fix}} \Rightarrow D_{\text{fix}}$, while the weights component score is descreasing under the $D_{\text{rnd}} \Rightarrow D_{\text{rnd}}$ setting. Note that the "Weights" and "Context" in the figure are short for weights comp. score and context comp. score respectively. **B.** The weights component score after 50 epoch training decreases when increasing the complexity of the task. The dashed green line indicates the weights component score at the initialization point. **C.** The weights component score at 50 epoch negative correlates with the length of learning plateaus. The dashed green line indicates the weights component score at the initialization point.

5. Breaking Through Learning Plateaus

The previous section examined the mechanisms behind learning plateaus and identified their connection to the weights component. In this section, our goal is to investigate *whether we can shorten the learning plateaus or improve the performance increasing for each transition process without scaling models*. We have chosen the D_{rnd} configure as the configuration of the test set by default, as it has demonstrated pronounced learning plateau behavior, and this particular scenario is known to be more effective in evaluating in-context learning capabilities, as discussed by Wei et al. (2023); Min et al. (2022).

5.1. Weights warm-up method

Previous experiments demonstrate that the model better learns the weight component more effectively in the D_{fix} setting. A straightforward intuition is whether we can use D_{fix} to improve the weights component in D_{rnd} test set. We give a further analysis of the relation between D_{rnd} and D_{fix}:

Increasing weights component with D_{fix}: Recall that we have $\mathbb{P}(y_p|\mathbf{x}_p, s_c) \sim \mathbb{P}(v_p|\mathbf{x}_p)\mathbb{P}(e_h|s_c, m)\mathbb{P}(m|s_c)$. Under D_{fix} setting, we only has one mapping function m_0. Consequently, our model can readily learn that $\mathbb{P}(m_0|s_c) = 1$. This simplification allows the model to concentrate on mastering $\mathbb{P}(v_\mathbf{x}|\mathbf{x})$. Therefore, it is anticipated that the model will develop a more refined weights component under this setting. **Knowledge transfering between D_{fix} and D_{rnd} settings:** The knowledge of $\mathbb{P}(v_\mathbf{x}|\mathbf{x})$ is shared between these two settings. The reason is that $\mathbb{P}(v_\mathbf{x}|\mathbf{x})$ is unrelated to the mapping function m and the context s_c.

Based on this analysis, we propose the following data configuration to improve the weights component on D_{fix} before training on D_{rnd}. This setting is denoted as $D_{\text{fix}\to\text{rnd}}$, which

means that we initially train the model on D_{fix} for a specific epoch (weights warm-up), and then, we train the model on D_{rnd}. We have the following discovery based on the experiments on $D_{\text{fix}\to\text{rnd}} \Rightarrow D_{\text{rnd}}$.

Weights warm-up helps to mitigate the learning plateaus. From Fig. 5A Top, we observe a notable enhancement in the weights component during the initial "warm-up" phase. Subsequent to this phase, there is a swift improvement in the accuracy of the in-context learning. Nonetheless, a subsequent decline in the weights component score post-warm-up is apparent. This further verify that the Transformer can not learn a good weights component under D_{rnd} settings. Further investigation is conducted by training the Transformer with various "warm-up" durations under the D_{fix} setting. As shown in Figure 5A Bottom, training the Transformer for a brief number of epochs effectively eliminates the initial learning plateaus. Furthermore, this "warm-up" phase contributes to a reduction in the duration of the subsequent learning plateaus. The application of the "warm-up" technique yields significantly better results compared to those without it, particularly after a training duration of 50 epochs.

5.2. mixed training method

In this section, we try to improve the weights component during the whole training process by designing the $D_{\text{fix}\wedge\text{rnd}} \Rightarrow D_{\text{rnd}}$ setting. In $D_{\text{fix}\wedge\text{rnd}}$ setting, half training data comes from D_{fix} setting, and half training data comes from D_{rnd} setting. For a fair comparison, the total number of training data is the same as the training set that uses D_{rnd} or D_{fix} setting only.

The mixed training method significantly boosts the learning process. As anticipated, both the weights component and the task accuracy exhibit improvements throughout the
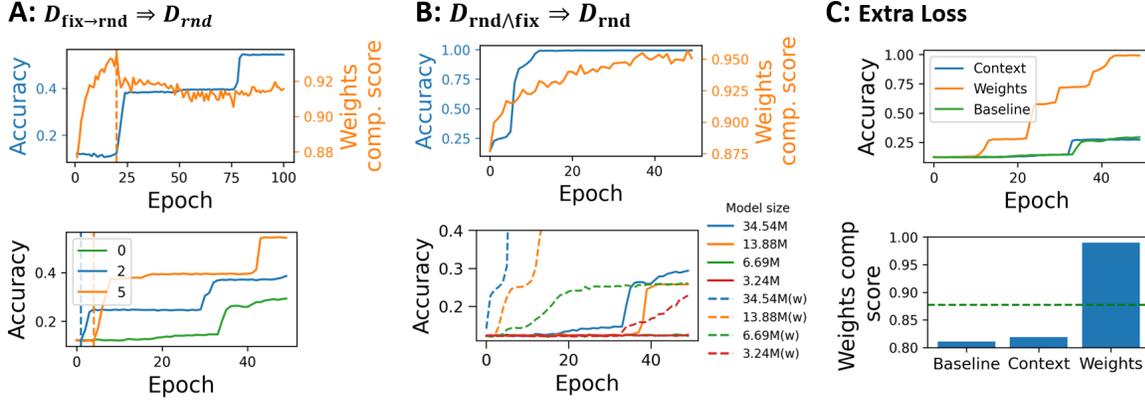
Figure 5. Three methods are proposed to assist in overcoming learning plateaus. **A: Effective of the warm-up method.** *Top:* Employing D_{fix} as a warm-up for the Transformer significantly mitigates learning plateaus. The dashed line indicates the transition point from D_{fix} to D_{rnd}. *Bottom:* We execute the transition from $D_{\text{fix}\to\text{rnd}} \Rightarrow D_{\text{rnd}}$ at various switching points. The curve labeled "2" signifies the switch from D_{fix} to D_{rnd} at epoch 2. The curve labeled "0" serves as the baseline, that is, $D_{\text{rnd}} \Rightarrow D_{\text{rnd}}$. The dashed lines highlight the respective switching points. **B: Combining D_{fix} and D_{rnd}.** *Top:* Mixed training substantially improves the weights component score during the learning process and eliminates learning plateaus. *Bottom:* Boosting the weights component can promote the development of in-context learning capabilities in smaller models. The dashed line depicts the task configuration $D_{\text{fix}\wedge\text{rnd}} \Rightarrow D_{\text{rnd}}$, while the solid line represents the $D_{\text{rnd}} \Rightarrow D_{\text{rnd}}$ setting. **C: Extra Loss.** *Top:* Incorporating a weights loss can significantly enhance learning, whereas adding context loss does not have a noticeable impact. The baseline is $D_{\text{rnd}} \Rightarrow D_{\text{rnd}}$. *Bottom:* With the weights loss, the Transformer can attain a commendable weights component score after 50 epochs of training. The green dashed line indicates the weights comp. score at the initialization point.

learning process, as depicted in Fig.5B Top. And the results do not show a pronounced learning plateau. This suggests that concurrently enhancing the weights component is beneficial. In Fig.5B Bottom, we compare models of various sizes trained on the $D_{\text{rnd}} \Rightarrow D_{\text{rnd}}$ setting (represented by the solid line) against those trained on the $D_{\text{rnd}\wedge\text{fix}} \Rightarrow D_{\text{rnd}}$ setting (depicted with a dashed line). It is observed that a model of size 6.69M trained on $D_{\text{fix}\wedge\text{rnd}}$, after 20 epochs of training, can attain an accuracy comparable to a model of size 13.88M trained on D_{rnd} that has undergone 50 epochs of training. These findings suggest that overcoming learning plateaus can lead to a reduction in the computational resources required for training.

5.3. Extra loss method

The previous methods require another data setting D_{fix} to improve the weights component. Here, we consider another alternate method by providing an extra supervision signal to improve the weights component when we cannot find the D_{fix} setting. We consider two extra loss settings. The $f'_{\mathbf{w}',s_c}(x)$ is denote as the subnetwork of $f_{\mathbf{w},s_c}(x)$ without the output classifier. We denote the cls_e as the classifier for the factor value of factor e. The weights loss is defined as

$$\mathcal{L}_w(\theta, s) = \frac{1}{L|E|} \sum_{e \in E} \sum_{i=1}^{L} l(cls_e(f'_{\mathbf{w}',s^{(i-1)}}(\mathbf{x}_i)), v_{\mathbf{x}}^{(e)}) \tag{5}$$

The context loss is defined as

$$\mathcal{L}_c(\theta, s) = \frac{1}{L} \sum_{i=1}^{L} l(cls_{e_h}(f'_{\mathbf{w}',s^{(i-1)}}(\mathbf{x}_i)), e_h^{(s)}), \tag{6}$$

where cls_{e_h} is the classifier to predict hidden factor, and $e_h^{(s)}$ is the hidden factor for sequence s. Then, the original loss function of Equation 3 is modified into $\mathcal{L}(\theta, s) + \lambda \mathcal{L}_c(\theta, s)$ or $\mathcal{L}(\theta, s) + \lambda \mathcal{L}_w(\theta, s)$. The λ is chosen as 0.1 in our experiments.

Add weights loss speedup the learning process while adding context loss fails. Fig. 5C Top reveals that incorporating an additional weights loss significantly aids the Transformer model in overcoming learning plateaus. In contrast, adding an extra context loss yields only a marginal benefit. These outcomes further substantiate our hypothesis that enhancing the weights component is crucial for breaking through learning plateaus, rather than concentrating on the context component. As shown in Fig. 5C Bottom, we note a marked improvement in the weights component when an extra weights loss.

5.4. Further Exploration: Simple Functions Tasks

In this section, we give further exploration of the relation between the weights component and the learning plateaus on the simple function tasks (Garg et al., 2022) and further understand the role of weights component in learning plateaus.
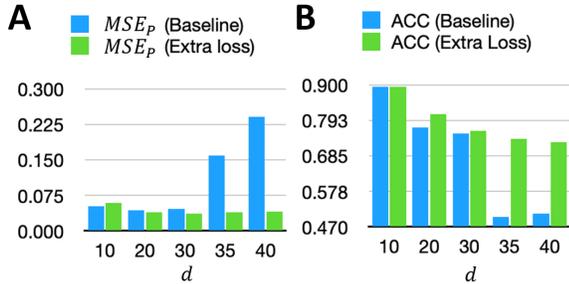
Figure 6. **Experiments results on simple function tasks after 50 epochs. A:** the dysfunction of the weights component happens when $d > 30$. **B:** the effect of Extra Loss technique is significant only when the dysfunction of weights component happens.

Two Roles of Weights Component Initially, I'd like to emphasize that the weights component serves two primary functions: 1) It takes the examples x_i into the internal working of the Transformer. 2) It aims to find a better representation for x_i. Considering this, the dysfunction of the weights component can stem from these two sources correspondingly. In the previous analysis of the Shape3D task, the weights component draws from two sources. Since simple function tasks lack representation learning, the dysfunction of the weights component might primarily be attributed to the first reason.

Date generation process In this task, we generate a dataset with n sequences. The generation process for each sequence is:

- Initially, we sample a vector $\mathbf{w}$ from the Gaussian distribution $\mathcal{N}(0, \mathbf{I}_d)$ in $\mathbb{R}^d$. Each component of $\mathbf{w}$ is drawn independently.

- Subsequently, we sample $\{\mathbf{x}_1, \mathbf{x}_2, \ldots\ldots, \mathbf{x}_L\}$, where each $\mathbf{x}_i \in \mathbb{R}^d$ is drawn from a Gaussian distribution $\mathcal{N}(0, \boldsymbol{I}_d)$.

- The labels are then determined using the formula $y_i = sign(\mathbf{w}^T\mathbf{x}_i)$. Finally, we obtain a sequence $s = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_L, y_L)\}$.

Exploring Framework The training methodology aligns with that described in the paper (Garg et al., 2022). We'll modify the dimension of $\mathbf{x}_i$, i.e. the value of d to modulate the difficulty of the task. The evaluation of the weights component follows the same process outlined in Appendix A.3, but with the utilization of the metric $MSE_p = \mathbb{E}_{\mathbf{x}_i}\|\tilde{\mathbf{x}}_i - \mathbf{x}_i\|$ to assess its performance, where $\tilde{\mathbf{x}}$ is the prediction of the probe model. We use the Transformer with 6 layers and we probe at the layer 3. It's worth noting that unlike the weight component score, where higher is better, lower values of MSE_p indicate better weights component.

Results The MSE_p and test accuracy are given in Figure 6. We find that: 1) Dysfunction in the weights component is evident in the SimpleFunction dataset when $d > 30$. The Transformer exhibits poor performance in these situations. 2) the effect of the Extra Loss technique is significant only when the dysfunction of weights component happens. The effect is incremental when the Transformer doesn't experience a significant weights dysfunction. This further verifies the causal relation between the weights component and learning plateaus.

5.5. Beyond Synthetic Task

To ensure that the proposed methods are broadly applicable, this section examines their adaptability to natural language processing (NLP) tasks.

SST-ICL task The task employs the SST dataset as outlined by Socher et al. (2013). An illustration of the task is given in Fig. 7A. Comprehensive details regarding the dataset structure, training methods, and model configurations are available in Appendix B. We assess the results for both the $D_{\text{fix} \wedge \text{rnd}} \Rightarrow D_{\text{rnd}}$. It is noted that improving the weights component through training on D_{fix} similarly aids in-context learning within this domain. Nevertheless, it should be acknowledged that variations exist between the outcomes of the SST-ICL task and those of the synthetic task; specifically, D_{fix} settings worse the performance when trained on small epochs, which is evident in the Figure 7 that the $D_{\text{fix} \wedge \text{rnd}}$ performs worse than D_{rnd} when training epoch is less than 30. The key reason to this divergence is that the weights component is more difficult to improve compared that in the synthetic one. As a result, more epochs of training are needed to improve the weights component so as to make it come into effect. This explanation is evident by the result in Figure 7 that the model trained on $D_{\text{fix} \wedge \text{fix}}$ setting significantly outperforms that trained on D_{rnd} settings after the 35 epochs.

WordSelection task We have devised an additional task named the WordSelection Task, which requires selecting a single word from a group of four options. For example, given the input "hello information learning art $\rightarrow$ learning", the task for the model is to identify "learning" as the correct choice from the provided set. The model must infer the correct answer by considering context examples. We offer five such in-context examples, all selecting the words at the same position as the prompt example. Creating a specific D_{fix} setting to improve the weights component poses a challenge in this task. Consequently, we opt for the implementation of the extra weights loss technique. We define the factor as the position and the factor values of a factor is the corresponding word in this position. For example, with $\mathbf{x}$ being "hello information learning art", we define $v_{\mathbf{x}}^{(e_1)}$ as "hello", $v_{\mathbf{x}}^{(e_2)}$ as "information", $v_{\mathbf{x}}^{(e_3)}$ as "learning" and
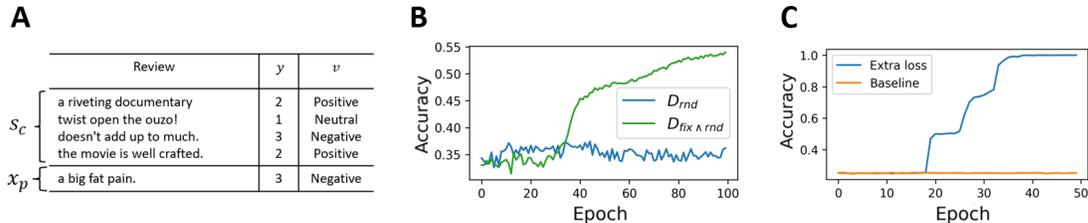
Figure 7. **Experiments on natural language tasks. A:** Example of dataset SST-ICL task. The example label y is obtained from v by map function m, i.e. $y = m(v)$. **B:** Results on SST-ICL task. We explore the $D_{\text{fix} \wedge \text{rnd}} \Rightarrow D_{\text{rnd}}$ and $D_{\text{rnd}} \Rightarrow D_{\text{rnd}}$ settings. The dashed line denotes the time when we transit from D_{fix} to D_{rnd}. **C:** Results on WordSelection Task. Adding extra weights loss has a significant effect in shortening the learning plateaus.

$v_{\mathbf{x}}^{(e_4)}$ as "art". The extra weights loss is then applied in a manner consistent with what is described in Eq. (5). Our experimental findings align with those from the synthetic task, indicating that the advantage of integrating an extra weights loss is not limited to synthetic environments. The results highlight that adding extra weights loss can be a beneficial strategy across different task types.

6. Discussion

Q1: What are the reasons and intuitions behind the dysfunction of weights? The observed dysfunction in the Transformer's performance may stem from the confluence of two crucial factors: 1) The problem-solving process relies on the information derived from both context examples and query examples. 2) The contextual information is hard to obtain. In situation 1, where both contextual and weight information are critical, the Transformer is compelled to harness both effectively. Yet, under condition 2, where the contextual information proves difficult to learn, the Transformer finds itself unable to fully extract this vital knowledge. Consequently, it lacks the incentive to refine the weight component at this stage, as doing so would not yield appreciable improvements in overall performance. This phenomenon aligns with our observation in Figure 4C that the weight dysfunction occurs predominantly when the in-context tasks exhibit a high degree of complexity, as measured by the high entropy of the distribution of m. As a result, we witness the manifestation of weight dysfunction, characterized by the stagnation or regression of the weight component during the learning process, despite the Transformer's ongoing attempts to adapt and solve the given problem.

Q2: Whether the decomposition of weights and context components is general? The conceptual decomposition is general based on the following reasons:

1) Firstly, the decomposition of the weights component and the context component is conceptual instead of physical and only for analysis purposes. The decomposition stems from the understanding that the in-context learning task necessitates information from both its context samples and the query example.

2) Secondly, our evaluation of the weights component and context component is not based on the physical decomposition of these components. We employ complex probe methods (see Section A.3) to analyze these two components because we have only conceptually decomposed them.

3) We tested our method across various scenarios, including the Shape3D task, SST-ICL task, Word Selection Task, and SimpleFunction task, which consistently validated the efficacy of our approach.

7. Conclusion

This paper establishes a connection between the weights component and learning plateaus. Building on this connection, the paper proposes three strategies to overcome these plateaus. These strategies have proven to be effective in both synthetic and natural language tasks.

8. Limitation

In this paper, we mainly focus on understanding of the learning plateaus of in-context learning with Transformer, where the Transformer requires the information from the context examples and the query example to make a prediction. Therefore, our method cannot explain the learning plateaus phenomenon outside this score. There are some works (Nanda et al., 2023; Power et al., 2022) that discuss the learning plateaus phenomenon in supervised learning. Our work fails to explain these phenomena.

Impact Statement

The goal of this paper is to understand the mechanism of learning plateaus in Transformers and find a method to avoid learning plateaus. Our work doesn't have a direct influence on society. However, future works based on our work may influence society but it is unpredictable currently.

References

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *arXiv preprint arXiv:2306.04637*, 2023.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Bietti, A., Cabannes, V., Bouchacourt, D., Jegou, H., and Bottou, L. Birth of a transformer: A memory viewpoint. *arXiv preprint arXiv:2306.00802*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Brunner, G., Liu, Y., Pascual, D., Richter, O., Ciaramita, M., and Wattenhofer, R. On identifiability in transformers. *arXiv preprint arXiv:1908.04211*, 2019.

Chan, S., Santoro, A., Lampinen, A., Wang, J., Singh, A., Richemond, P., McClelland, J., and Hill, F. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891, 2022a.

Chan, S. C., Dasgupta, I., Kim, J., Kumaran, D., Lampinen, A. K., and Hill, F. Transformers generalize differently from information stored in context vs in weights. *arXiv preprint arXiv:2210.05675*, 2022b.

Dai, D., Sun, Y., Dong, L., Hao, Y., Ma, S., Sui, Z., and Wei, F. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023.

Deora, P., Ghaderi, R., Taheri, H., and Thrampoulidis, C. On the optimization and generalization of multi-head attention. *arXiv preprint arXiv:2310.12680*, 2023.

Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., and Sui, Z. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

Edelman, B. L., Goel, S., Kakade, S., and Zhang, C. Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning*, pp. 5793–5831. PMLR, 2022.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Giannou, A., Rajput, S., Sohn, J.-y., Lee, K., Lee, J. D., and Papailiopoulos, D. Looped transformers as programmable computers. *arXiv preprint arXiv:2301.13196*, 2023.

Huang, Y., Cheng, Y., and Liang, Y. In-context convergence of transformers. *arXiv preprint arXiv:2310.05249*, 2023.

Kim, H. and Mnih, A. Disentangling by factorising. In *International Conference on Machine Learning*, pp. 2649–2658. PMLR, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kirsch, L., Harrison, J., Sohl-Dickstein, J., and Metz, L. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*, 2022.

Li, Y., Ildiz, M. E., Papailiopoulos, D., and Oymak, S. Transformers as algorithms: Generalization and stability in in-context learning. *International Conference on Machine Learning*, 2023.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

Lu, S., Bigoulaeva, I., Sachdeva, R., Madabushi, H. T., and Gurevych, I. Are emergent abilities in large language models just in-context learning? *arXiv preprint arXiv:2309.01809*, 2023.

Michaud, E. J., Liu, Z., Girit, U., and Tegmark, M. The quantization model of neural scaling. *arXiv preprint arXiv:2303.13506*, 2023.

Min, S., Lewis, M., Zettlemoyer, L., and Hajishirzi, H. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*, 2021.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Press, O., Smith, N. A., and Levy, O. Improving transformer models by reordering their sublayers. *arXiv preprint arXiv:1911.03864*, 2019.

Reddy, G. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. *arXiv preprint arXiv:2312.03002*, 2023.

Richter, O. and Wattenhofer, R. Normalized attention without probability cage. *arXiv preprint arXiv:2005.09561*, 2020.

Schouten, S., Bloem, P., and Vossen, P. Probing the representations of named entities in transformer-based language models. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 384–393, 2022.

Shin, S., Lee, S.-W., Ahn, H., Kim, S., Kim, H., Kim, B., Cho, K., Lee, G., Park, W., Ha, J.-W., et al. On the effect of pretraining corpora on in-context learning by a large-scale language model. *arXiv preprint arXiv:2204.13509*, 2022.

Singh, A. K., Chan, S. C., Moskovitz, T., Grant, E., Saxe, A. M., and Hill, F. The transient nature of emergent in-context learning in transformers. *arXiv preprint arXiv:2311.08360*, 2023.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. 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, 2013.

Tian, Y., Wang, Y., Chen, B., and Du, S. Scan and snap: Understanding training dynamics and token composition in 1-layer transformer. *arXiv preprint arXiv:2305.16380*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Voita, E. and Titov, I. Information-theoretic probing with minimum description length. *arXiv preprint arXiv:2003.12298*, 2020.

Voita, E., Sennrich, R., and Titov, I. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. *arXiv preprint arXiv:1909.01380*, 2019.

von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*, 2022.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Wei, J., Wei, J., Tay, Y., Tran, D., Webson, A., Lu, Y., Chen, X., Liu, H., Huang, D., Zhou, D., et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023.

Wiegreffe, S. and Pinter, Y. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*, 2019.

A. Detail of experiments on sythetic tasks

A.1. 3DShape

3dshapes[1] (Kim & Mnih, 2018) is a dataset of 3D shapes procedurally generated from 6 ground truth independent latent factors. These factors are floor colour, wall colour, object colour, scale, shape and orientation. All possible combinations of these latents are present exactly once, generating N = 480000 total images. Latent factor values including: 1) floor hue: 10 values linearly spaced in [0, 1], 2) wall hue: 10 values linearly spaced in [0, 1], 3) object hue: 10 values linearly spaced in [0, 1], 4) scale: 8 values linearly spaced in [0, 1], 5) shape: 4 values in [0, 1, 2, 3], and 6) orientation: 15 values linearly spaced in [-30, 30].

Shape3D dataset We employ the Adam optimizer (Kingma & Ba, 2014) and mini-batch training to optimize the loss function $\mathcal{L}(\theta, S)$. Here, we use cross-entropy as the loss function. We utilize a batch size of 128 and set the learning rate to 0.0001. For training purposes, we use 10^5 sequences and, for evaluation, 4×10^4 sequences. There is no overlap between images in the training sequences and those in the evaluation sequences.

A.2. Architecture Detail

We employ a pre-trained Variational Autoencoder (VAE) to transform images into tokens. The encoder of the VAE comprises seven convolutional layers with ReLU activations, followed by three linear layers two ReLU activations. Conversely, the VAE's decoder is structured with three linear layers and two ReLU activations, which precede a sequence of seven convolutional layers also with ReLU activations. The resulting latent representation serves as the token representation of the input image. The label embedding is learned during the in-context training process. Our study's primary goal is to explore the characteristics of in-context learning. To this end, we utilize a causal Transformer architecture that restricts each token to only interact with preceding tokens. Specifically, the default configuration of the Transformer, denoted as f, includes 12 layers, 4 attention heads, and an embedding dimension of 128. As depicted in the lower section of Figure 5B, we experiment with varying model sizes. Detailed configurations for these model sizes can be found in Table 1.

Table 1. Model configure with different size in Fig 5B Bottom

Number of Layers	12	6	6	3
Attention head	4	4	2	2
Embedding size	128	64	32	16
Model Size	34.53	13.88	6.69	3.24

A.3. More detail about probe framework
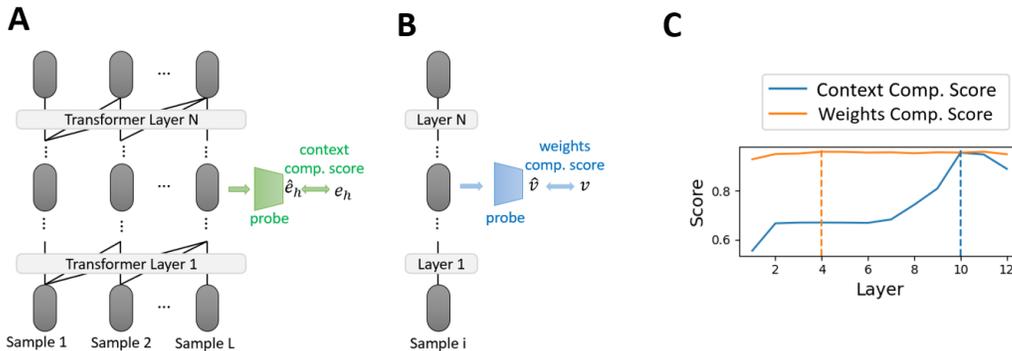
Figure 8. Probe method. **AB:** illustration of the probe method. **C:** The weights and in-context score when we probe at different layers. We choose the $D_{\text{fix} \land \text{rnd}} \Rightarrow D_{\text{rnd}}$ settings. The dashlines marked the chosen layers in the experiments.

We employ metrics for numerical evaluation of components and in-context learning performance. Since the components are

---

[1]https://github.com/deepmind/3d-shapes

hidded in the representation, we use the probe method (Alain & Bengio, 2016). The probe classifier has a single linear layer, with softmax and cross-entropy calculating the loss. **Because we don't know the specific form of $g_{weights}$ and $g_{context}$, we choose the representation within the Transformer that given the high score for weights and context components for approximating.** We use the linear probe because using more complex probe model doesn't have significant improve in accuracy and the change of the accuracy during the training process is more important than the absolutely value. The probe model is trained utill totally converge. The probe is trained for 2 epoch for context component and 1 epoch for weights component. The in-weight probe predicts values of six factors of all images, while the in-context probe identifies the hidden factor for each sequence. The details are as follows.

context comp. score Context comp. score measures whether the Transformer can capture the information from all the context examples, i.e., whether the inner representation within Transformer can capture the distribution $\mathbb{P}(y_p|\mathbf{x}_p, s_c)$. Given the test set S', the context comp. score is calculated as $\frac{1}{|S'|} \sum_{s \in S'} \mathbf{1}_{\hat{e}_{h,s}=e_{h,s}}$, where $\mathbf{1}_{expr}$ is indicator function, s is the sequence in the dataset S, $e_{h,s}$ is the hidden factor for the sequence s, and $\hat{e}_{h,s}$ is the prediction of probe classifier. We use $|\cdot|$ to denote the corresponding size of a set.

weights comp. score The weights comp. score measures whether the Transformer can learn the information from individual examples, i.e. whether the inner representation of Transformer can capture the distribution $\mathbb{P}(v_x|\mathbf{x})$. Because the distribution $\mathbb{P}(v_x|\mathbf{x})$ is unrelated to the context examples s_c, we remove the influence of context examples by removing the context examples when evaluate the weights component score. The weights comp score is calculated as $\frac{1}{|S'|} \sum_{s \in S'} \frac{1}{|s||E|} \sum_{(x,y) \in s} \sum_{e \in E} \mathbf{1}_{\hat{v}_x^{(e)}=v_x^{(e)}}$, where v_x^e is factor value of factor e and sample x , $\hat{v}_k^{(e)}$ is the prediction of probe classifier, $s = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}$ is the sequence in the dataset S' and E is the set of all factors.

Accuracy We measure the accuracy of the prediction of in-context learning task given a fix number of context example as the measure for the in-context performance. In this paper, we choose the number of context examples as 39.

A.4. Dataset split

In-context training We first split all the the images in Shape3D into two part: the training image set (80 %) and the test image set (20 %). Then, we organize all the training images into $S_{\text{fix}}, S_{\text{rnd}}, S_{\text{fix}\wedge\text{rnd}}$, corresponding to $D_{\text{fix}}, D_{\text{rnd}}, D_{\text{fix}\wedge\text{rnd}}$ settings. $S_{\text{fix}}, S_{\text{rnd}}, S_{\text{fix}\wedge\text{rnd}}$ Test image set are also organized into $S'_{\text{fix}}, S'_{\text{rnd}}, S'_{\text{fix}\wedge\text{rnd}}$. Each of $S_{\text{fix}}, S_{\text{rnd}}$, and $S_{\text{fix}\wedge\text{rnd}}$ contains 10^5 sequences. Each of $S'_{\text{fix}}, S'_{\text{rnd}}, S'_{\text{fix}\wedge\text{rnd}}$ contains 4×10^4 sequences.

Probe model training If we want to probe a model $f_{\mathbf{w},s_c}(\cdot)$ on setting D_{rnd} (test setting), we will first train the probe model on S_{rnd} with $f_{\mathbf{w},s_c}(\cdot)$ and we evaluate the probe model on S'_{rnd} with $f_{\mathbf{w},s_c}(\cdot)$. The same for D_{fix} and $D_{\text{fix}\wedge\text{rnd}}$ settings.

B. Detail of experiments on langugae task.

A: SST-ICL Task

\<y=3\>

Transformer (causal)

a riveting documentary \<y=2\> $\cdots$ $\cdots$ $\cdots$ a big fat pain \<cls\>

s_c x_p

B: WordSelection Task

the

Transformer (causal)

cost disease meet teach -\> meet $\cdots$ $\cdots$ a bit the life -\>

s_c x_p
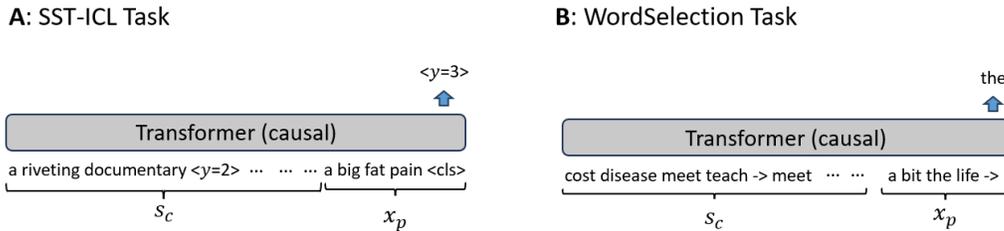
Figure 9. Example of the SST-ICL task and WordSelection task.

B.1. Experiments on Pythia

We first indroduce the experiment design on the Pythia 13B model, because setup of this experiment is totally different from the other experiments on Section 5.5. We leverage the opensouce of the Pythia 13B checkpoints. The Pythia is trained using autoregression framework on the Pile dataset. We evaluate the performance of the Pythia model for each 10k steps. We construct 400 sequence of 4 wordselection task. All the words are randomly sampled from 2000 words. 4 context examples are given for each prompt. The prediction of the Pythia model is processed to ensure that it has same form as the given label.

B.2. Model Struture

For both tasks, we use the GPT2 model in this setting. The model is consist of 6 layers, 4 attention heads with 368 embedding size.

B.3. Dataset Detail

SST-ICL dataset   The dataset is contructed based on SST (Socher et al., 2013) datasets. We remove the long review in the datasets and transform the original labels into "Negative", "Positive" and "Neutral". Then, we organize the reviews follow the same way as that in Subsection A.4. We produce 10^4 sequence for training and 4×10^3 for testing. Each sequence contains 5 reviews. We illustrate the example of the dataset in Fig. 7 AB.

WordSelection Dataset   We choose 2000 words for the experiments. The 2000 words is organzed into 10^5 training sequence and 5×10^4 training sequence. Each sequence contain six examples and five examples served as in-context examples and the rest served as the prompt example.

B.4. Training Detail

For SST-ICL task and WordSelection task, the models are both trained using AdamW optimizer with learning rate $2e - 5$. We choose the batch size as 64.

C. Other Related works

C.1. In-weights and in-context learning

Previous studies (Chan et al., 2022a;b) examined the relationship between in-weights learning and in-context learning. The division of in-weights learning and the in-context learning process is conceptually similar to our distinction between the influence of the weights and context component. In in-context learning, the Transformer relies on a combination of both its internal weights and the context provided to address a given task. In contrast, for in-weights learning, the model depends exclusively on its weights. **Nevertheless, there are notable differences between these two notion**s: 1) The internal weights and context components are concerned with capturing information within the Transformer, whereas the focus of in-weights and in-context learning is on how the Transformer tackles a task. In-weights learning is a rename for regular supervised learning. 2) Both weights and context components coexist and play roles within the paradigm of in-context learning. While in-weights learning is distinguished from in-context learning.

C.2. Evidence of previous works regarding weights and context component are both important for in-context learning

In this section, we provide evidence about that the in-context and weights components in practice tasks.

Intuition 1: Influence of words replacing   A key difference between the weights and context components lies in the susceptibility of the weights component to word substitution. The weights component can be easily disrupted if a word is replaced with a token that was not present during the training phase, as the weights lack information about this new token. On the other hand, if the context examples are rich in information, the meaning of this new token can still be deduced. This mirrors the human ability to infer the meaning of an unknown word based on its context. If word substitution leads to a decline in performance, it suggests that the Transformer's prediction relies heavily on the weights component.

Intuition 2:Influence of number of in-context examples   The efficiency of the context component is expected to rise with the inclusion of more context-specific examples, a characteristic not shared by the weights components, which remain unaffected by the addition of in-context examples. Therefore, if performance improves with the integration of more context-specific examples, it would suggest that the Transformer's prediction is heavily influenced by the context component.

Intuition 3: Zero-shot performance   The zero-shot performance can directly indicate the effectiveness of the weights component. This is because no in-context examples are provided in this scenario, reducing the problem to a traditional supervised one

Based on the intuitions above, we collect the related experiments in practice paper.

1. Min et al. (2022) discovered that (1) performance can be improved by increasing the number of in-context examples. (2) Changing the labels of in-context examples does not influence the predicted label. The first discovery indicates that the prediction relies on the context components. The second discovery suggests that the Transformer uses the weights component for label prediction, given that there is no observed change when the labels of in-context examples are altered.

2. Brown et al. (2020) found that larger models are increasingly effective at utilizing in-context information. This suggests that in real-world scenarios, the efficiency of the context component improves with the enlargement of the model's size. Brown et al. (2020) also found that enhancing the model size can boost its zero-shot capabilities. These findings suggest that scaling the model can enhance both the weights and context components, and the model employs these two components to address the problem.

3.(Wei et al., 2023) carried out research on a two-class classification issue. They conducted experiments in which they altered a certain percentage of labels in the context examples to ascertain if the model's prediction would also change. If a change was observed, it would imply that the prediction relies on the context components. If no change was noticed, the prediction would be considered to depend on the weights component. Their results were intermediate, suggesting that both weights and context components contribute. Additionally, they found that enhancing the model size increases the impact of in-context examples.

C.3. Related works for understanding Transformer

Analysis of Transformer   The analysis of Transformers can be broken down into two main components: examining the expressibility of Transformers and comprehending the mechanisms of learned Transformers. To analyze the expressibility of Transformers, a common approach is to determine if they can solve specific problems by constructing appropriate weights. Giannou et al. (2023) demonstrates that Transformers can function as Turing machines, while Liu et al. (2022) shows that they can learn shortcuts to solve automata problems. In addition to expressibility, researchers have also investigated the mechanisms behind learned Transformers. Bietti et al. (2023) examines Transformers from a memory standpoint, and Tian et al. (2023) focuses on single-layer Transformers. While the analysis of Transformers is crucial to our work, our ultimate goal differs; we aim to bridge the gap between representation learning and in-context learning.

Exploration of representation within Transformer.   Owing to the widespread use of Transformers, numerous studies (Li et al., 2022; Voita & Titov, 2020) seek to investigate their internal representations as a means of comprehending their functionality. The most prevalent approach involves utilizing probe models and tasks to discern the information stored within these representations (Voita & Titov, 2020; Schouten et al., 2022). Taking a different perspective, Voita et al. (2019) explores the flow of information across Transformer layers and how this process is influenced by the selection of learning objectives. Our work shares similarities with these studies in that we employ the probe method to examine representations. However, our focus differs in that we do not concentrate on the semantic meaning within the representation. Instead, we investigate how the weights and in-context information impact representation.

D. Proof of Proposition 3.2

Proposition D.1. *Given y_p, probability of $\mathbb{P}(y_p|\mathbf{x}_p, s_c)$ can be decomposite as:*

$$\mathbb{P}(y_p|\mathbf{x}_p, s_c) = \sum_{v_p, m, e_h} \mathbb{P}(y_p|v_p, m, e_h)\mathbb{P}(v_p|\mathbf{x}_p)\mathbb{P}(e_h|s_c, m)\mathbb{P}(m|s_c). \tag{7}$$

Proof.

$$
\begin{aligned}
\mathbb{P}(y_p|\mathbf{x}_p, s_c) &= \sum_{v_p, m, e_h} \mathbb{P}(y_p, v_p, m, e_h|\mathbf{x}_p, s_c) \\
&= \sum_{v_p, m, e_h} \mathbb{P}(y_p|\mathbf{x}_p, s_c, v_p, m, e_h)\mathbb{P}(v_p, m, e_h|\mathbf{x}_p, s_c,) \\
&= \sum_{v_p, m, e_h} \mathbb{P}(y_p|v_p, m, e_h)\mathbb{P}(v_p|\mathbf{x}_p, s_c, m, e_h)\mathbb{P}(m, e_h|\mathbf{x}_p, s_c) \\
&= \sum_{v_p, m, e_h} \mathbb{P}(y_p|v_p, m, e_h)\mathbb{P}(v_p|\mathbf{x}_p)\mathbb{P}(e_h|s_c, m)\mathbb{P}(m|s_c),
\end{aligned}
\tag{8}
$$

where the first equation is due to the law of total probability, the third equation is leverages the formular $\mathbb{P}(y_p|\mathbf{x}_p, s_c, v_p, m, e_h) = \mathbb{P}(y_p|v_p, m, e_h)$.

$\square$

E. Contruction analysis of why the weights component is related to the learning plateaus
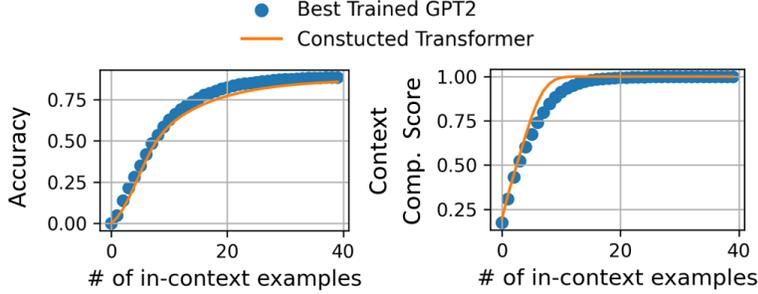
Figure 10. The constructed Transformer can match the performancce of trained Transformer ($D_{\text{fix}\wedge\text{rnd}}$ setting) in experiment part

Intuition   In this section, our objective is to comprehend the connection between the weights component of the model and the occurrence of learning plateaus. Owing to the challenges involved in dissecting the training dynamics of the Transformer, we turn to constructive analysis as a methodology. By examining a scenario where a Transformer possesses an effective weights component at a specific layer, and only a small number of additional layers are required to attain proficient in-context learning performance, we can indirectly infer insights about their relationship.

Notation   The position embedding is denoted as $\mathbf{p}_i = (0, \cdots 0, 1, 0, \cdots)$, where we only have value 1 at i-th position and 0 others. The weights for the attention operation of l-th layer and c-th head in Transformer is denoted as $\mathbf{W}_Q^{(l,c)}$, $\mathbf{W}_K^{(l,c)}$ and $\mathbf{W}_V^{(l,c)}$. The weights of the forward layer in the Transformer are denoted as $\mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}$. We use E to denote all possible values of the factor e. we denote $\mathbf{y}_i$ as the one hot version of y. The vector with all zero values is denoted as $\mathbf{0} \triangleq (0, \cdots, 0)$. We consider the naive Transformer (Vaswani et al., 2017). The hidden representation of token i in Transformer is denoted as $\mathbf{h}_i \in \mathbb{R}^d$. The hidden representation of l-th layer is denoted as $\mathbf{H}^{(l)} = [\mathbf{h}_1^{(l)}, \cdots, \mathbf{h}_{2L}^{(l)}]^\mathrm{T} \in \mathbb{R}^{2L \times d}$. Given a input token $\mathbf{x}$, we denote $\mathbf{h}_\mathbf{x}^{(l)}$ as its corresponding representation at layer l. The factor value of this token is denoted as $v_\mathbf{x}$. The factor value of the corresponding factor e is denoted as $v_\mathbf{x}^{(e)}$. The set of all possible values of factor e is denoted as V_e. The size of the set is denoted as $|V_e|$.

In our analysis, we explore a modified, more relaxed variant of the Transformer model. The rationale behind this relaxation is underpinned by evidence suggesting that 1) employing the ReLU activation function in the feed-forward layers can achieve results comparable to the original model (Press et al., 2019), and 2) the softmax operation may not be essential for the functioning of the Transformer (Wiegreffe & Pinter, 2019; Brunner et al., 2019; Richter & Wattenhofer, 2020). The relaxed Transformer is defined as follows:

Definition E.1. (Transformer) One layer of the Transformer contains one attention layer and one MLP layer. The calculation of the Attention Layer is

$$\text{Attn}^{(l)}(\mathbf{H}^{(l)}) = \mathbf{H}^{(l)} + \sum_{c=1}^{C} \sigma\left(\mathbf{H}^{(l)}\mathbf{W}_Q^{(l,c)}(\mathbf{H}^{(l)}\mathbf{W}_K^{(l,c)})^\mathrm{T}\right)\mathbf{H}^{(l)}\mathbf{W}_V^{(l,c)}\mathbf{W}_O^{(l,c)}. \tag{9}$$

And the calculation of MLP layer is

$$\mathbf{H}^{(l+1)} = \text{Attn}^{(l)}(\mathbf{H}^{(l)}) + \text{Relu}(\text{Attn}^{(l)}(\mathbf{H}^{(l)})\mathbf{W}_1^{(l)})\mathbf{W}_2^{(l)}. \tag{10}$$

Here we consider relaxed case where $\sigma = \text{Id}$.

Remember that a good weights component implies that we have the ability to deduce $\mathbb{P}(v_\mathbf{x}|\mathbf{x})$ based on this component. To streamline the construction, we introduce a stronger assumption: the concept of a perfect weights component. Unlike the

16

definition of a good weights component, which necessitates that the Transformer encapsulates information about $\mathbb{P}(v_{\mathbf{x}}|\mathbf{x})$, a perfect weights component also demands that this information should be readily accessible. If the representations of the images corresponding to different factor values are situated in distinct orthogonal bases, then the factor values' information can be easily decoded. Drawing on this insight, we propose the following definition.

Definition E.2. (Perfect weights component) If a Transformer has perfect weights component in layer l, then for all factor e, any i, j, exists $\mathbf{W}_e \in \mathbb{R}^{d \times |V_e|}$ such that $\mathbf{f}_{\mathbf{x}_1}^{(e)} \cdot \mathbf{f}_{\mathbf{x}_2}^{(e)} = 1$ only when $v_{x_1}^{(e)} = v_{x_2}^{(e)}$, else we have $\mathbf{f}_{\mathbf{x}_1}^{(e)} \cdot \mathbf{f}_{\mathbf{x}_2}^{(e)} = 0$, where $\mathbf{f}_{\mathbf{x}}^{(e)} = \mathbf{W}_e \mathbf{h}_{\mathbf{x}}^{(l)}$.

Under the assumption of a perfect weights component, we can enhance the Transformer by adding at most three extra layers that are specifically designed to learn the context component. The detailed results of this construction are as follows:

Proposition E.3. *We consider the data with n_e factors and each factor has n_v values in D_{rnd} setting. For causal Transformer with the number of heads larger or equal the number of factors with the hidden size $\mathcal{O}(n_e n_v + L)$, if the Transformer can learn a perfect weights component in layer k, then it can learn a representation given i in-context examples with context comp. score $\mathrm{srs}_i = (1 - \mathrm{srs}_{i-1})s_i + \mathrm{srs}_{i-1}$ and $\mathrm{srs}_0 = s_0$ at layer $k + 2$, where $s_i = 1 - \sum_{j=0}^{i} \binom{i}{j} \sum_{k=2}^{|E|} \binom{|E|}{k} \frac{k-1}{k} \left(\frac{(n_v-1)^{i-j}}{n_v^i} \right)^k \left(1 - \frac{(n_v-1)^{i-j}}{n_v^i} \right)^{|E|-k}$, and we can obtain the accuracy as $\mathrm{cls}_i = \frac{(n_v-1)(n_v^{i-1} - (n_v-1)^{i-1})}{n_v^i} \mathrm{srs}_i + \frac{1}{n_v}$ at $k + 3$ layers.*

The constucted Transformer achieve significant performance. To illustrate that the construction is meaningful, we compare the performance of the constructed model with the previously trained model. We choose the performance of the model trained on $D_{\mathrm{fix} \wedge \mathrm{rnd}}$ settings as it is the best-performed model on D_{rnd} setting. We find that the constructed model achieved a comparable performance as that of the trained model. These results indicate that our construction is meaningful.

E.1. Proof Sketch

a) Contruction of the Transformer.

We divided the contruction into two steps. The first step is to estimate the factor in the sequence and the second step is to estimate the label based on the discovered hidden factor. Given the sequence $s = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}$, we short $\mathbf{f}_{\mathbf{x}_j}$ as $\mathbf{f}_j$.

1. Estimate the hidden factor. According to the perfect weights component assumption, for any j, we can project the j-th token feature into the space $\mathbf{f}_j^{(e)}$. Assuming the j-th token is not the prompt token, then we have its label information y_j. Then, for i-th token, where $i < j$, obviously i is also not the prompt token. As a result, we also have the information about y_i. If a factor e is the hidden factor, then we would expect $\mathbf{y}_i \cdot \mathbf{y}_j = \mathbf{m}(v_i^{(e_h)}) \cdot \mathbf{m}(v_j^{(e_h)}) = \mathbf{v}_i^{(e_h)} \cdot \mathbf{v}_j^{(e_h)} = \mathbf{f}_i^{(e_h)} \cdot \mathbf{f}_j^{(e_h)}$, where $\mathbf{v}, \mathbf{y}, \mathbf{m}$ is the corresponding one-hot version of v, y, m. Therefore, if we can find e such that $\mathbf{f}_i^{(e)} \cdot \mathbf{f}_j^{(e)}$ can have a same value as $\mathbf{y}_i \cdot \mathbf{y}_j$, for all i, then e can be predicted as hidden factor. Based on this intuition, in the construction, we focus on finding a way to compare the value between $\mathbf{f}_i^{(e)} \cdot \mathbf{f}_j^{(e)}$ and $\mathbf{y}_i \cdot \mathbf{y}_j$. All these operations are done in the first layer.

2. Block unrelated information. After finding the hidden factor, the next step is to block the information that is unrelated to the hidden factor. Blocking unrelated information can remove the influence of it and simplify the following steps. We add a large negative value to the positions of the representation that is unrelated to the hidden factor. Then, through the Relu operation, all these negative values will be removed. These operations are placed in Layer 2.

3. Predict y. The final step is to predict the y_p for the prompt sample $\mathbf{x}_p$. The challenge here is that we don't know the mapping function m that bridges the factor value and the corresponding label. Consider the relation that for any i, j satisfying $i < j$, we have $\mathbf{v}_i^{(e_h)} \cdot \mathbf{v}_j^{(e_h)} = \mathbf{f}_i^{(e_h)} \cdot \mathbf{f}_j^{(e_h)} = \mathbf{y}_i \cdot \mathbf{y}_j$. Therefore, if we can find a i, satisfying that $\mathbf{f}_i^{(e_h)} \cdot \mathbf{f}_j^{(e_h)} = 1$, then we have $y_i = y_j$. Then, we can copy y_i to the representation of j-th token and use the final linear layer to output y_i as the prediction for y_j.

b) Analyzing the performance.

Given the perfect weights component assumption, there are two source that cause the error prediction. We give the separate analysis below.

1. Fail to find the correct hidden factor. Finding the correct hidden factor is essential to make the correct prediction. The correct hidden factor cannot be inferred if there are more than two factors such that the values of $\mathbf{f}_i^{(e)} \cdot \mathbf{f}_j^{(e)}$ and $\mathbf{y}_i \cdot \mathbf{y}_j$ can

be matched according to the construction of first layer.

2. *Make error prediction and give the correct factor.* Given the hidden factor, it is also possible that we cannot make the correct prediction if we fail to associate the correct label the factor value. This will happen if all the context examples don't contain the same factor value of the hidden factor as the prompt image.

E.2. Proof of useful lemma

In this section, we proof some useful lemma for the proof. The lemma E.4 indicates that the attention head can copy part of the representation from its previous token into current token. The lemma E.5 indicates that there exists a construction to make the MLP to only operate on the part of its input.

Lemma E.4. *One attention head can implement copy and past behavior.*

Proof. According to the definition of $\mathbf{p}_i$, we have $\mathbf{p}_i \cdot \mathbf{p}_j = 0$ if $i \neq j$, otherwise, we have $\mathbf{p}_i \cdot \mathbf{p}_j = 1$. We denote

$$
\mathbf{M} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}.
$$

Then we have $\mathbf{p}_i \mathbf{M} = \mathbf{p}_{i-1}$. For $j > i$, we denote the value of $2j$-th token as $\mathbf{h}_{2j} = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{p}_j)$ and $2i$-th token as $\mathbf{h}_{2i} = (\mathbf{h}'_i, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{p}_i)$. If we want to copy the value of $2i$-th token to the value of $2j$-th token, we can set the query matrix as $\mathbf{W}_Q = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{M}^{j-i})$, the key matrix as $\mathbf{W}_K = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{I})$ and value matrix as $\mathbf{W}_V = (\mathbf{W}'_V, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$. Then we have

$$
\mathbf{h}_{2i}^{\mathrm{T}} \mathbf{W}_Q \cdot \mathbf{h}_{2a}^{\mathrm{T}} \mathbf{W}_K = \mathbf{p}_i \cdot \mathbf{p}_a = \begin{cases} 1 & a \neq j \\ 0 & a = j \end{cases} \tag{11}
$$

Therefore, the $2j$-th token can only attend to the token with the position embedding $\mathbf{p}_i$. If $\mathbf{h}_{2i-1} = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{p}_i)$, we have the value of $\mathbf{h}_j$ after attention as $\mathbf{h}_j^{attn} = ((\mathbf{h}'_i)^{\mathrm{T}} \mathbf{W}_V, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{p}_j)$. By setting $\mathbf{W}_V$ as different value, we can copy different part information of i-th to j-th token. Then the lemma is held. □

Lemma E.5. *For the input* $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3)$*, where* $\mathbf{h}_i \in \mathbb{R}^{d_i}$ *and* $d_1 + d_2 + d_3 = d$*, for all* $MLP_s(\mathbf{h}) = \mathbf{W}'_2 \operatorname{Relu}(\mathbf{W}'_1 \mathbf{h}_2)$: $\mathbb{R}^{d_2} \to \mathbb{R}^{d_2}$*, there exists* $MLP(h) = \mathbf{W}_2 \operatorname{Relu}(\mathbf{W}_1 \mathbf{h}) : \mathbb{R}^d \to \mathbb{R}^d$*, such that* $MLP(\mathbf{h}) = (\mathbf{h}_1, MLP_s(\mathbf{h}_2), \mathbf{h}_3)$*.*

Proof. Obviously, for any $\mathbf{W}'_1$, there exists $\mathbf{W}_1$, such that $\mathbf{h}^{(a)} \triangleq \mathbf{h} \mathbf{W}_1 = (\mathbf{h}_1, -\mathbf{h}_1, (\mathbf{W}'_1 \mathbf{h}_2), \mathbf{h}_3^{\mathrm{T}}, -\mathbf{h}_3)$.

Obviously, for any $\mathbf{W}'_2$, There exists $\mathbf{W}_2$, such that $\mathbf{h}^{(b)} = \mathbf{W}_2 \operatorname{Relu}(\mathbf{h}^{(a)}) = ((\operatorname{Relu}(\mathbf{h}_1) + \operatorname{Relu}(-\mathbf{h}_1)), (\mathbf{W}'_2 \operatorname{Relu}(\mathbf{W}'_1)), (\operatorname{Relu}(\mathbf{h}_3) + \operatorname{Relu}(-\mathbf{h}_3))) = (\mathbf{h}_1, MLP_s(\mathbf{h}_2), \mathbf{h}_3)$ □

E.3. Construction of Transformer

Without loss of generality, we assume the representation of the Transformer in layer k is in a form that $\mathbf{h}_{2i-1}^{(k)} = (\mathbf{f}_i, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{p}_i)^{\mathrm{T}}$ and $\mathbf{h}_{2i}^{(k)} = (\mathbf{0}, \mathbf{y}_i, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{p}_i)^{\mathrm{T}}$ (Remind that one sample will take two token, one for x and one for y). Because the representation usually lies in low-dimension space, a simple linear layer can transfer the representation in our defined sparse form. Moreover, it is natural to assume that the position information is stored in the representation since it is given in the input and is essential for attention.

The consider the operations of Transformer in different layers.

1) ** Layer 1 **

Because we assume that $\mathbf{h}_{2i-1}^{(k)}$ is a perfect token representation, then there exists $\mathbf{W}_e$, such that $\mathbf{h}_{2i-1}^{(k)} \mathbf{W}_e = \mathbf{f}_k^{(e)}$, where $\mathbf{f}_i^{(e)}$ satisfies that $\forall e, i$, we have $\mathbf{f}_j^{(e)} \cdot \mathbf{f}_i^{(e)} = 1$ only when $v_i^{(e)} = v_j^{(e)}$ else $\mathbf{f}_j^{(e)} \cdot \mathbf{f}_i^{(e)} = 0$.

Step 1, we use each attention head to obtain the matching information of each factor.

We first consider the query token at the position $2i - 1$ And we assign $\mathbf{W}_Q^{(l,k)} = \mathbf{W}_K^{(l,k)} = \mathbf{W}_{e_k}$ and $\mathbf{W}_V^{(l,k)} = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{I})^{\mathrm{T}}$ so that $(\mathbf{h}_i^{(l)})^{\mathrm{T}} \mathbf{W}_V^{(l,k)} = \mathbf{p}_i$. Then, we have

$$\mathbf{b}_{e_k} = \sum_{a=1}^{2i-2} (\mathbf{h}_i^{\mathrm{T}} \mathbf{W}_Q^{(l,k)} \cdot \mathbf{h}_a^{\mathrm{T}} \mathbf{W}_K^{(l,k)}) \mathbf{h}_i^{\mathrm{T}} \mathbf{W}_V^{(l,k)} = \sum_{a=1}^{i-1} \mathbf{p}_a (\mathbf{v}_a^{(e_k)} \cdot \mathbf{v}_i^{(e_k)}), \tag{12}$$

where $\mathbf{v}_i^{(e)}$ is the one-hot vector of $v_i^{(e)}$. We denote $\text{base} = (2^0, 2^1, \cdots, 2^L)^{\mathrm{T}}$ and $\mathbf{u}_{2i-1} = (\{\text{base} \cdot \mathbf{b}_{e_1}, \cdots, \text{base} \cdot \mathbf{b}_{e_{n_e}}\}$. Obvious, there is $\mathbf{W}_O^{(l,k)}$ such that $\sum_{k=1}^{n_e} \mathbf{b}_{e_k} \mathbf{W}_O^{(l,k)} = (\mathbf{0}, \mathbf{0}, \mathbf{u}_{2i-1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$.

Then, we consider the token at position $2i$ as query token. We assign $\mathbf{W}_Q^{(l,n_e+1)} = \mathbf{W}_K^{(l,n_e+1)} = (\mathbf{0}, \mathbf{I}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})^{\mathrm{T}}$ and $\mathbf{W}_V^{(l,n_e+1)} = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{I})^{\mathrm{T}}$. We have

$$\mathbf{b}_y = \sum_{a=1}^{2i-1} (\mathbf{h}_i^{\mathrm{T}} \mathbf{W}_Q^{(l,n_e+1)} \cdot \mathbf{h}_a^{\mathrm{T}} \mathbf{W}_K^{(l,n_e+1)}) \mathbf{h}_i^{\mathrm{T}} \mathbf{W}_V^{(l,n_e+1)} = \sum_{a=1}^{i-1} \mathbf{p}_a (\mathbf{y}_a \cdot \mathbf{y}_i) \tag{13}$$

Obvious, there is $\mathbf{W}_O$ such that $\mathbf{b}_y \mathbf{W}_O^{(l,n_e+1)} = (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{u}_{2i}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$, where $\mathbf{u}_{2i} = \{\text{base} \cdot \mathbf{b}_y, \cdots, \text{base} \cdot \mathbf{b}_y\}$.

The $n_e + 1$ head doesn't affect the token $2i - 1$, because value of the $\mathbf{h}_{2i-1}$ is $\mathbf{0}$. As a result, we have $\mathbf{h}_{2i-1} = \mathbf{h}_{2i-1} + \sum_{k=1}^{n_e} \mathbf{b}_{e_k} \mathbf{W}_O^{(l,k)} = (\mathbf{f}_i, \mathbf{0}, \mathbf{u}_{2i-1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{p}_i)$ after the operation. Similarily, because the first n_e head dosen't affect the value of $\mathbf{h}_{2i}$, we have $\mathbf{h}_{2i} = (\mathbf{0}, \mathbf{y}_i, \mathbf{0}, \mathbf{u}_{2i}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{p}_i)$ after the operation.

Note that $\text{base} \cdot \mathbf{b}_{e_k}$ has the property that $\text{base} \cdot \mathbf{b}_{e_k} = \text{base} \cdot \mathbf{b}_{e_{k'}}$ if and only if $\mathbf{b}_{e_k} = \mathbf{b}_{e'_k}$. This result indicates that all the context examples that have same factor value of factor e_k with the sample i also has the same factor value of factor $e_{k'}$ as sample i. **Therefore, we denote $\mathbf{u}$ as the matching information.** If $\text{base} \cdot \mathbf{b}_{e_k} = \text{base} \cdot \mathbf{b}_y$, we can infer that the factor value of e_k has a similar pattern with the label. Therefore, e_k is regard as the possible hidden factor.

Step 2: compare the $\mathbf{u}_{2i-1}$ and $\mathbf{u}_{2i}$ to infer possible hidden factor.

For embedding of $\mathbf{h}_{2i}$, using the copy past of Lemma E.4, we can obtain $\mathbf{h}_{2i} = (\mathbf{0}, \mathbf{y}_i, \mathbf{0}, \mathbf{u}_{2i}, \mathbf{u}_{2i-1}, \mathbf{0}, \mathbf{0}, \mathbf{p}_i)$. (By setting the copy position as $\mathbf{p}_i$ and therefore the operation will only influence y token.) According to Lemma E.5, there exists $\mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}$, such that we have $\mathbf{h}_{2i} = (\mathbf{0}, \mathbf{y}_i, \mathbf{0}, \mathbf{u}_{2i}, \mathbf{u}_{2i-1}, \mathbf{m}_{2i}, \mathbf{0}, \mathbf{p}_i)$, where $\mathbf{m}_{2i} = \text{Relu}(\mathbf{u}_{2i} - \mathbf{u}_{2i-1}) + \text{Relu}(\mathbf{u}_{2i-1} - \mathbf{u}_{2i})$. Recall that $\mathbf{h}_{2i-1} = (\mathbf{f}_i, \mathbf{0}, \mathbf{u}_{2i-1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{p}_i)$. because all the corresponding terms of $\mathbf{h}_{2i-1}$ are $\mathbf{0}$, this operation won't impact the value of it. This copy past operation can be put into a same layer as the pervious operations is because in this operation we mainly copy the infomation from $2i - 1$ token to $2i$-token. Because $2i - 1$ precede $2i$, the operations of $2i - 1$ is finished before the copy past operation happens.

The value of $\mathbf{m}_{2i}$ has the property that the k-th position in $\mathbf{m}_{2i}$ is equal to 0 if the values of k-th position of $\mathbf{u}_{2i-1}$ and $\mathbf{u}_{2i}$ are equal. After this operation, we have $\mathbf{h}_{2i} = (\mathbf{0}, \mathbf{y}_i, \mathbf{0}, \mathbf{u}_{2i}, \mathbf{u}_{2i-1}, \mathbf{m}_{2i}, \mathbf{0}, \mathbf{p}_i)$ and $\mathbf{h}_{2i-1} = (\mathbf{f}_i, \mathbf{0}, \mathbf{u}_{2i-1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{p}_i)$.

2) ** Layer 2 **

Blocking the information according to $\mathbf{m}$.

First attention head: for $\mathbf{y}$ token, at position $2i$, we apply Lemma E.4 to copy $\mathbf{m}_{2i-2}$ from $\mathbf{h}_{2i-2}$ to $\mathbf{h}_{2i}$. Due to the weights sharing of attention, this yield a iterative effect. We denote $\mathbf{m}'_{2i-1} = 2\mathbf{m}'_{2i-3} + \mathbf{m}_{2i-2}$ and $\mathbf{m}'_{2i} = \mathbf{m}_{2i-1} + \mathbf{m}_{2i}$. Therefore, we have $\mathbf{h}_{2i} = (\mathbf{0}, \mathbf{y}_i, \mathbf{0}, \mathbf{u}_{2i}, \mathbf{u}_{2i-1}, \mathbf{m}'_{2i}, \mathbf{0}, \mathbf{p}_i)$. Because of weights sharing, we have $\mathbf{h}_{2i-1} = (\mathbf{f}_i, \mathbf{0}, \mathbf{u}_{2i-1}, \mathbf{0}, \mathbf{0}, \mathbf{m}'_{2i-1}, \mathbf{0}, \mathbf{p}_i)$.

Second attention head: In this layer, for $\mathbf{y}$ token, at position $2i$, we apply Lemma E.4 to copy $\mathbf{f}_i$ from $\mathbf{h}_{2i-1}$ (This operation only affects $\mathbf{y}$ tokens). We have $\mathbf{h}_{2i} = (\mathbf{f}_i, \mathbf{y}_i, \mathbf{0}, \mathbf{u}_{2i}, \mathbf{u}_{2i-1}, \mathbf{m}'_{2i}, \mathbf{0}, \mathbf{p}_i)$.

MLP Layer: We denote $\mathbf{f}'_{i,x} \triangleq (\mathbf{f}_i^{(e_j)} - M\mathbf{m}'_{2i-1}[1], \cdots, \mathbf{f}_i^{(e_j)} - M\mathbf{m}'_{2i-1}[n_e])$ and $\mathbf{f}'_{i,y} \triangleq (\mathbf{f}_i^{(e_j)} - M\mathbf{m}'_{2i}[1], \cdots, \mathbf{f}_i^{(e_j)} - M\mathbf{m}'_{2i}[n_e])$. M is a large constant value. In $\mathbf{f}'_{i,x}$, we will block the information of j-th factor if $\mathbf{m}'_{2i-1}[j] > 0$. $\mathbf{m}'_{2i-1}[j] < 0$ if and only if $\forall k < i$, $\mathbf{m}_{2k}[j] = 0$. The same for $\mathbf{f}'_{i,y}$ In MLP, we calculate $\text{Relu}(\mathbf{h}_{2i-1}^{\mathrm{T}} \mathbf{W}_1^{(l+2)}) \mathbf{W}_1^{(l+2)} = (\mathbf{f}_i, \mathbf{f}'_i) \mathbf{W}_1^{(l+2)} = (\mathbf{f}'_i - \mathbf{f}_i, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$. Then, we have $\mathbf{h}_{2i-1} = \text{Relu}(\mathbf{h}_{2i-1}^{\mathrm{T}} \mathbf{W}_1^{(l+2)}) \mathbf{W}_1^{(l+2)} + \mathbf{h}_{2i-1} = (\mathbf{f}'_{i,x}, \mathbf{0}, \mathbf{u}_{2i-1}, \mathbf{0}, \mathbf{m}'_{2i-1}, \mathbf{0}, \mathbf{0}, \mathbf{p}_i)$. And similar, we have $\mathbf{h}_{2i} = (\mathbf{f}'_{i,y}, \mathbf{y}_i, \mathbf{0}, \mathbf{u}_{2i}, \mathbf{u}_{2i-1}, \mathbf{m}'_{2i}, \mathbf{0}, \mathbf{p}_i)$.

3) ** Layer 3 **

This layer obtains the logit of the new sample by comparing the similarity between the unblocked feature of this sample and the in-context sample.

Setting $\mathbf{W}_Q^{(l+3,1)} = \mathbf{W}_K^{(l+3,1)} = (\mathbf{I}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$, we have $\mathbf{h}_i^{\mathrm{T}} \mathbf{W}_Q^{(l+3,1)} = \mathbf{h}_i^{\mathrm{T}} \mathbf{W}_K^{(l+3,1)} = \mathbf{f}_i'$. Setting $\mathbf{W}_V^{(l+3,1)} = (\mathbf{0}, \mathbf{I}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0})$ such that $\mathbf{h}_{2i}^{\mathrm{T}} \mathbf{W}_V^{(l+3,1)} = \mathbf{y}_i$ and $\mathbf{h}_{2i-1}^{\mathrm{T}} \mathbf{W}_V^{(l+3,1)} = 0$.

For position $2i - 1$, we have

$$\text{Logit} = \sum_{a=1}^{2i-2} (\mathbf{h}_i^{\mathrm{T}} \mathbf{W}_Q^{(l+3,1)} \cdot \mathbf{h}_a^{\mathrm{T}} \mathbf{W}_K^{(l+3,1)}) \mathbf{h}_i^{\mathrm{T}} \mathbf{W}_V^{(l+3,1)} = \sum_{a=1}^{i-1} (\mathbf{f}_{i,x}' \cdot \mathbf{f}_{a,y}') \mathbf{y}_a'. \tag{14}$$

Note that value $\mathbf{f}_{i,x}' \cdot \mathbf{f}_{a,y}'$ is equal to the number of unblocked factors (both unblocked) that have the same value between a-th sample and i-th sample Obviously, there is a $W_O^{(l+3,1)}$ such that $\mathbf{h}_{2i-1} = (\mathbf{f}_{i,x}', \mathbf{0}, \mathbf{u}_{2i-1}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{m}_{2i-1}', \text{Logit}, \mathbf{p}_i)$.

Finally, we output Logit using the prediction head.

E.4. Performance analysis

Here, we will analyze the sequence representation score and the in-context learning accuracy of our constructed model.

1) **context comp. score**

Propability for same factor value between in-context examples and prompt sample   The probability for an in-context example having the same value of a factor as the prompt sample is $\frac{1}{n_v}$ and the probability of having different values is $\frac{n_v - 1}{n_v}$. Therefore, given i samples, the probability for j samples have the same value of a factor as the prompt sample is $\binom{i}{j} \frac{(n_v-1)^{i-j}}{n_v^i}$.

Probability for connot distinguish factors   Given i in-context examples, we cannot distinguish k factors to decide which one is the hidden factor if the k factors satisfying that of $\forall\, e_1, e_2 \in E_k,\ (x, y) \in s_c$, we have $v_x^{(e_1)} = v_p^{(e_2)} \Leftrightarrow v_x^{(e_2)} = v_p^{(e_2)}$, where E_k is the set of these k factors, s_c is in-context examples, and v is factor value.

Given i in-context examples, the probability for that we cannot distinguish k factors is

$$\binom{|E|}{k} \sum_{j=0}^{i} \binom{i}{j} \left(\frac{(n_v - 1)^{i-j}}{n_v^i} \right)^k \left(1 - \frac{(n_v - 1)^{i-j}}{n_v^i} \right)^{|E|-k}. \tag{15}$$

context comp. score   When we cannot distinguish the hidden factor from k factors, the probability of predicting wrong results is $\frac{k-1}{k}$. Combining the results above, we obtain the error:

$$\sum_{k=2}^{|E|} \binom{|E|}{k} \sum_{j=0}^{i} \binom{i}{j} \frac{k-1}{k} \left(\frac{(n_v - 1)^{i-j}}{n_v^i} \right)^k \left(1 - \frac{(n_v - 1)^{i-j}}{n_v^i} \right)^{|E|-k}. \tag{16}$$

The probability of giving the right prediction is

$$s_i = 1 - \sum_{j=0}^{i} \binom{i}{j} \sum_{k=2}^{|E|} \binom{|E|}{k} \frac{k-1}{k} \left(\frac{(v - 1)^{i-j}}{v^i} \right)^k \left(1 - \frac{(v - 1)^{i-j}}{v^i} \right)^{|E|-k}. \tag{17}$$

In the constructed Transformer, we will autoregressively combine the results of the previous prediction (Corresponding to Layer 2). We have:

$$\text{srs}_i = (1 - \text{srs}_{i-1}) s_i + \text{srs}_{i-1},$$

where $\mathrm{srs}_0 = s_0$.

2) **Accuracy**

The copy-past mechanism is used to predict the answer to the prompt example (Corresponding to layer 3). For the copy-past mechanism, having an in-context example with the same prediction result as the prompt example is necessary. When we correctly predict the hidden factor, the probability to predict correctly is $1 - \left(\frac{n_v - 1}{n_v}\right)^i$. When we predict a wrong hidden factor, the probability is $\frac{1}{n_v}$. Combine the two above, we obtain the accuracy as follows

$$(1 - \left(\frac{n_v - 1}{n_v}\right)^i)\,\mathrm{srs}_i + \frac{1}{n_v}(1 - \mathrm{srs}_i) = \mathrm{cls}_i = \frac{(n_v - 1)(n_v^{i-1} - (n_v - 1)^{i-1})}{n_v^i}\,\mathrm{srs}_i + \frac{1}{n_v} \tag{18}$$