# Skill Path: Unveiling Language Skills from Circuit Graphs

**Hang Chen**[1], **Xinyu Yang**[1], **Jiaying Zhu**[2], **Wenya Wang**[3*]
[1]Xi'an Jiaotong University
[2]The Chinese University of Hong Kong
[3]Nanyang Technological University
albert2123@stu.xjtu.edu.cn, yxyphd@mail.xjtu.edu.cn, jyzhu24@cse.cuhk.edu.hk, wangwy@ntu.edu.sg

## Abstract

Circuit graph discovery has emerged as a fundamental approach to elucidating the skill mechanistic of language models. Despite the faithfulness of circuit graphs, they suffer from atomic ablation, which causes the loss of causal dependencies between connected components. In addition, their discovery process, designed to preserve output faithfulness, inadvertently captures extraneous effects other than an isolated target skill. To alleviate these challenges, we introduce **skill paths**, which offers a more refined and compact representation by isolating individual skills within a linear chain of components. To enable skill path extracting from circuit graphs, we propose a three-step framework, consisting of **decomposition**, **pruning**, and **post-pruning causal mediation**. In particular, we offer a complete linear decomposition of the transformer model which leads to a disentangled computation graph. After pruning, we further adopt causal analysis techniques, including counterfactuals and interventions, to extract the final skill paths from the circuit graph. To underscore the significance of skill paths, we investigate three generic language skills—Previous Token Skill, Induction Skill, and In-Context Learning Skill—using our framework. Experiments support two crucial properties of these skills, namely stratification and inclusiveness.

**Code** —
https://github.com/Zodiark-ch/Language-Skill-of-LLMs
**Appendix** — https://arxiv.org/abs/2410.01334v3

## 1 Introduction

Circuit Discovery (Elhage et al. 2021; Bhaskar et al. 2024) is pivotal for comprehending language model functionality. Current methods (Yao et al. 2024; Syed, Rager, and Conmy 2023; Bhaskar et al. 2024) achieve this by selectively pruning low-effect edges or nodes within the computational graph, ensuring a precise circuit graph that preserve task outputs, thereby elucidating the model's task execution mechanism. Typically, these tasks align with specific language model skills, such as the induction task (Conmy et al. 2023), greater than task (Hanna, Liu, and Variengien 2024), and reverse task (Lindner et al. 2023).

Although the circuit graph ensures output faithfulness, capturing the target skill mechanism has two **constraints**:

- **1)** Current circuit graphs, particularly derived from real-world datasets, include unrelated skills to the intended task skill (Arora and Goyal 2023), as depicted in Figure 1.
- **2)** These graphs use edge- or node-ablation pruning, which ignores causal dependencies between synergistic components crucial for a skill (Mueller 2024). However, due to MLPs, the computational graph is not fully linear, making it difficult to isolate the effect of a path that contains multiple components.

To effectively isolate the target skill mechanism, we introduce the **skill path**, a compact and finer "subcircuit" consisting of a linear sequence of components, which reflects the precise location of a target skill in circuits. Additionally, skill paths naturally facilitate the exploration of skill evolution, such as how advanced skills evolve from basic skills.

Therefore, we propose a novel framework consisting of 3 steps, namely **Decomposition, Pruning,** and **Post-pruning Causal Mediation**, to extract skill paths from circuit graphs. Specifically, we present the **compensation component**, enabling a linear breakdown of MLP ($\text{mlp}(\text{attn}(x)+x)$) inputs into $\text{mlp}(\text{attn}(x))$ and $\text{mlp}(x)$, thus isolating paths within the same MLP into separate activations. Furthermore, we define **functional components** to encapsulate all fundamental components of the computation graph, ensuring each path is constituted by these functionally identical components (**Decomposition**). Based on the functional components, we collected a large number of samples containing the target skill and obtained the circuit graph sets using existing pruning strategies with activation patching (**Pruning**). For these circuit graphs, we use counterfactuals and interventions to remove the paths responding to other skill effects (**Post-pruning Causal Mediation**), and the resulting **skill paths** represent the complete effect of a target skill.

The advantage of the three-step framework lies in:

- 1) **Decomposition** provides a fully linear factorization of the computational graph, such that the effects of the underlying functional components are isolated. As a result, ablating a specific path does not affect components outside that path, thereby mitigating constraint 2.
- 2) **Post-pruning causal mediation** introduces counterfactuals and interventions to remove subcircuits that are irrelevant to the target skill, thus addressing constraint 1.

To show the potential capability of skill paths, we select

*Corresponding Author

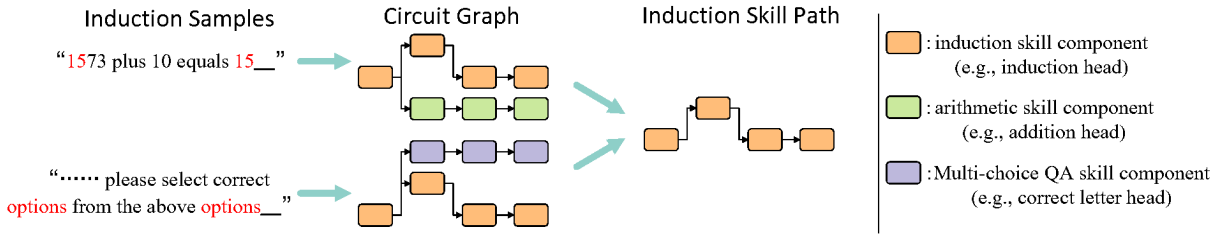



Figure 1: The difference and correlation between skill path and circuit graph. We use the induction dataset as an example and present two types of samples of induction. When the induction dataset contains a certain number of samples related to arithmetic skills, the final circuit graph may contain parts of the paths for arithmetic skills in addition to the induction skill, as may those samples containing multi-choice skills or other potential skills. We verified in Appendix G that even cross-domain randomly sampled datasets still have the confounding of other skills.

three generic and progressively complex skills which have been introduced in (Crosbie and Shutova 2024; Ren et al. 2024; Edelman et al. 2024): a) *Previous Token skill* which is responsible for receiving information from the previous token; b) *Induction Skill* which duplicates tokens with the same prefix; and c) *ICL Skill* which perform inference based on similar patterns presented in demonstrations. Using our 3-step framework, we unveil the skill paths of these skills. These skill paths have better interpretability in skill interaction, providing stronger evidence to confirm 2 conjectures that have long remained unverified: **Stratification**: Simpler language skills reside in shallower layers, whereas more complex language skills are in deeper layers. **Inclusiveness**: More complex skill paths are formed on top of simpler skill paths.

In summary, our contributions are threefold.

- We propose a theoretical derivation for the complete and lossless linear decomposition of computation graphs, which can decouple all decoder-based language models into finer-grained "functional components".
- We propose a novel three-step framework for discovery of skill paths, including complete paths of a generic skill.
- Our analysis and experiments verify 2 properties among the Previous Token Skill, Induction Skill, and ICL Skill, which include stratification and inclusiveness.

## 2 Related Work

Existing methods typically utilize corrupted output to replace the activation of some key nodes (such as attention, MLP, etc.) or edges to observe changes in the final output, thus determining whether to prune the corresponding nodes or edges in the computation graph (Conmy et al. 2023; Syed, Rager, and Conmy 2023; Bhaskar et al. 2024; Yao et al. 2024). However, as it is challenging to ensure completely uniform and random sampling in real datasets, there will always be some samples that contain other potential skills. This leads to the possibility that the circuit graph may contain skill paths unrelated to the target skill, which cannot be detected by metrics in circuit discovery, such as task accuracy.

Additionally, some methods have proposed "path patching" (Goldowsky-Dill et al. 2023) and "treeification" (Chan et al. 2022) techniques, which observe changes in the entire path by replacing the sender node. However, these methods still fall within the scope of 'edge- or component-based causal analysis" because they can only ablate the activation of a particular node or edge. Thus, they cannot avoid the pitfalls of "preemption" and "non-transitivity" (Mueller 2024)(Example of these two pitfalls are elaborated in Appendix A). Unlike existing causal analysis, we directly consider the causal effects of the entire path under corrupted output.

## 3 Method

In this paper, we propose a novel three-step framework to extract the target skill paths.

- **Step 1** (Section 3.1, 3.2): We decouple the architecture of language models into completely linear "functional components". This results in a *Computation Graph*, $\mathcal{G}$.
- **Step 2** (Section 3.3): Utilizing the existing pruning strategies, we transform the $\mathcal{G}$ into a *Circuit Graph*, $\mathcal{G}*$.
- **Step 3** (Section 3.4): We select those paths rendering the most significant causal effect in $\mathcal{G}*$ on the target skill. The final graph formed by the skill paths is named *Skill Graph*, denoted $\mathcal{G}^S$.

### 3.1 Decomposition

In each layer of the language model, the input of MLP includes the input $X$ of this layer and the output $\text{attn}(X)$ of attention, that is, the activation of the MLP is represented as $\text{mlp}(\text{attn}(X) + X)$. However, due to the presence of nonlinear activations in MLP, its activation cannot be directly decomposed into $\text{mlp}(\text{attn}(X))$ and $\text{mlp}(X)$, which hinders the ablation of path-level structures. For example, if a path contains a sub-path "attention $\rightarrow$ MLP", its ablation could affect the activations of another path with this MLP. Therefore, we introduce a **Compensation Component** $Cps$, which is capable of representing the input to the MLP in the form of a linear combination, for example: $\text{mlp}(\text{attn}(X) + X) = \text{mlp}(\text{attn}(X)) + \text{mlp}(X) + Cps(X)$. In practice, when $\text{attn}(X)$ is decomposed into the sum of each head, there is an additional compensation component (see Table 1 and Appendix B for more details).

Building on the foundation of the Transformer Circuit (Elhage et al. 2021), we propose a complete decomposition of the transformer model including the MLP layers with omission of compensation components. Using tensor products

| Index | Category | Implementation ($X$=input representation in each layer) |
|---|---|---|
| $C^0$ | Self | $X$ |
| $C^{1-12}$ | Attention | $A^h\text{LN}(X)W_VW_O + A^hb_VW_O$ |
| $C^{13}$ | MLP | $\text{atv}(\text{LN}(X)W_{M1})W_{M2}$ |
| $C^{14-25}$ | Attention+MLP | $\text{atv}(\text{LN}(A^h\text{LN}(X)W_VW_O + A^hb_VW_O)W_{M1})W_{M2}$ |
| $C^{26}$ | Compensation | $(\text{atv}(\text{LN}((\sum_{h=1}^{12} C^h)W_{M1})) - \sum_{h=1}^{12}\text{atv}(\text{LN}(C^h)W_{M1}))W_{M2}$ |
| $C^{27}$ | Compensation | $(\text{atv}((\text{LN}(C^{0-13})W_{M1}) - \text{atv}(\text{LN}(C^0)W_{M1}) - \text{atv}(\text{LN}(\sum_{h=1}^{12} C^h)W_{M1}))W_{M2}$ |
| $C^{28}$ | Bias | $b_v + \text{atv}(b_{M1})W_{M2} + b_{M2} + \sum_{h=1}^{12}\text{atv}(b_VW_{M1})W_{M2}$ |

Table 1: Specific component index and corresponding implementation in each layer of GPT2-small. $W$ and $b$ represent weight and bias parameters, atv represents the activation of MLP. LN$(\cdot)$ is the layernorm function. $A = \text{softmax}(XW_QW_K^TX^T + b_QW_K^TX^T + XW_Qb_K^T + b_Qb_K^T)$. Functional Component are $C^{1-25}$.

$(\otimes)$, we can represent any layer of the transformer model:

$$\text{output} = (\text{Id} + \sum_{h\in H} A^h \otimes W_{OV}^h + \text{Id} \otimes W_{\text{MLP}} + \sum_{h\in H} A^h \otimes W_{\text{MLP}}W_{OV}^h) \cdot X$$

where $H$ represents the number of attention heads. The matrix $A$ is given by the attention mechanism $A = \text{softmax}((XW_Q)(XW_K)^T)$, and $W_{\text{MLP}}$ involves the MLP operation with activation given by $\text{atv}(XW_{M1})W_{M2}$. $W_{OV} = W_OW_V$ refers to an "output-value" matrix which computes how each token affects the output if attended to, while $W_Q, W_K, W_V$ are parameter matrices for query, key and value. $W_{M1}$ and $W_{M2}$ are weight parameters in two linear layers. This equation simplifies both the attention and MLP modules into linear matrix mappings, describing how the paths from input to output for each layer are decoupled into four independent circuits: 1) $C^{self} = \text{Id} \cdot X$; 2) $C^{\text{attn}} = \sum_{h\in H} A^h \otimes W_{OV}^h \cdot X$; 3) $C^{\text{mlp}} = \text{Id} \otimes W_{\text{MLP}} \cdot X$; 4) $C^{\text{attn+mlp}} = \sum_{h\in H} A^h \otimes W_{\text{MLP}}W_{OV}^h \cdot X$.

Moreover, $C^{\text{attn}}, C^{\text{mlp}}, C^{\text{attn+mlp}}$ can be factorized as:

$$C^{\text{attn/mlp/attn+mlp}} = f(X) \cdot W \tag{1}$$

We use $f$ to represent a function that can be considered equivalent to an activation function, for instance, $C^{\text{attn}} = \sum_{h\in H} f_{W_{QK}}^{\text{attn}}(X) \cdot W_{OV}$, $f_{W_{QK}}^{\text{attn}}(X)$ represents the softmax-normalization of the input $X$ through a weighted accumulation performed by $QK$ values, i.e., $f_{W_{QK}}^{\text{attn}}(X) = \text{softmax}((XW_Q)(XW_K)^T)X$.

The function $f(X)$ possesses the ability for non-linear transformations, while $W$ is an input-agnostic parameter, which can be understood as a memory learned during training (Geva et al. 2021). Therefore, these three components represent the minimum and complete unit for manipulating how much memory to read (i.e., memory-reading function), and are independent of each other, which we refer to as **"functional component"** (We elaborate the functional component in detail in Appendix B.)

## 3.2 Implementation of Computation Graph

In this paper, we select GPT2-small as the target language model, containing 12 layers ($L = 12$) and 12 attention heads ($H = 12$). To provide a complete dissection of the the model at each layer which can precisely recover the original output, we introduce *bias components* apart from *functional component* and *compensation component*, to compensate for the remaining information not covered. Table 1 shows the specific components and their implementation for each layer. Notably, our component dissection leads to a lossless decomposition of the original LM layer into fully linear combinations: $LM_l(X) = \sum_{i=0}^{28} C^i$.

We treat functional components as the smallest units and build the computation graph $\mathcal{G} = \{\mathcal{C}, \mathcal{E}\}$, where $\mathcal{C}$ stands for the set of 29 components ($C^{0-28}$ shown in Table 1) and $\mathcal{E}$ represents the edge between any two components in successive layers. Any functional component $C^i (0 \leqslant i \leqslant 25)$ in any layer $l (0 \leqslant l \leqslant 11)$, denoted as $C^{l,i}$, receive information from all components in previous layers, i.e., $\mathcal{E} = \{(C^{l_1,i} \to C^{l_2,j})\} (0 \leqslant l_1 < l_2 \leqslant 11, 0 \leqslant i, j \leqslant 25)$. Notably, lossless decomposition ensures that the insights gained from our linear decomposition computation graph accurately reflect the behavior of the original language model.

## 3.3 Implementation of Pruning

For the $\mathcal{G}$ obtained in Section 3.2, we default to using the Automated Circuit Discovery (Conmy et al. 2023) toolkit to construct a circuit graph $\mathcal{G}*$. In simple terms, we eliminate those components that cause significant changes to original output in KL divergence through patching with interchange ablation in the computation graph and finally get the subgraph to completely reflect the task mechanisms as circuit graphs. To support subsequent post-pruning causal mediation, instead of deriving a single circuit graph from the average of a dataset, we obtain a set of circuit graphs by computing the average of every 10 samples within the dataset.

## 3.4 Post-pruning Causal Mediation

In this section, we employ the post-pruning causal mediation to eliminate the effects of potential skills and other noise effects contained in real-world samples. Motivated by efforts in causal effect analysis (Wang et al. 2023; Vig et al. 2020), we divide the circuit graphs set (obtained from section 3.3) into 3 sets of paths (sub-circuits): the paths of **skill effects**, the paths of **background effects**, and the paths of **self effects for the last token** (abbreviated as **self effects**). **Skill effects**

refer to the effect of the target skill on the output, which is the focus of this paper. **Self effects** denote the impact of only using the last token to predict by memorization, which functions like a "bi-gram model" (e.g., inputting "Francis" and outputting "Bacon"). **Background effects** refer to the effects of other potential skills in the input text (we illustrate these three effects through example texts in Appendix C.).

We use the typical example of the "Induction" skill for illustration, which works with an input in the form of "...*A B*...*A*", where *A* and *B* refers to different tokens. Here, the language model is expected to repeat the pattern "*A B*" it has seen in the context and predict token "*B*" as the output token.

Figure 2 illustrates that the model outputs "*question*" when given the input "*Generate a question with a*". However, the vocabulary distribution in the output given by the language model does not merely result from the induction skill, but is also confounded by other effects such as the background effect and the self effect. To compute the target effect for a specific skill path, let $\text{Path}^i$ be any directed paths in $\mathcal{G}*$ (e.g., $C^{1,19} \rightarrow C^{2,14} \rightarrow C^{6,5}$ $s.t.$ circuit edges $(C^{1,19}, C^{2,14})$ and $(C^{2,14}, C^{6,5})$ are in $\mathcal{G}*$). $\text{Path}^i$ then symbolizes the flow of information across layers amongst the components it encompasses. We use the occurrence rate of $\text{Path}^i$ in all samples to compute the effect:

$$\text{Eff}(\text{Path}^i_{\mathcal{G}*}) = \frac{N_{\text{Path}^i_{\mathcal{G}*}=1}}{N_{all}} \tag{2}$$

$N_{\text{Path}^i_{\mathcal{G}*}=1}$ represents the number of samples encompassing $\text{Path}^i$ while $N_{all}$ represents the number of all samples. Each path contributes differently to the three effects. Hence, our aim is to find those paths that contribute to the skill effect rather than the other two effects.

Specifically, for each input text as a sample $s$, we perturb it to create a background text $s_{\text{Bkg}}$ and a self text $s_{\text{Self}}$ (i.e., background and self text are two types of corrupted texts, and the process for generating these two text for all types of skills is described in Appendix D). Any sample is augmented with two more perturbed versions, rendering three types of input (i.e., original text, background text, and self text), each of which is subjected to the pruning as discussed in Section 3.3. The pruning produces three distinct circuit graphs: $\mathcal{G}_{\text{Ori}}*$ (from the original input text), $\mathcal{G}_{\text{Bkg}}*$ (from the background text) and $\mathcal{G}_{\text{Self}}*$ (from the self text). Therefore, the skill effect (e.g., *Induction Skill*) of $\text{Path}^i$ can be defined as:

$$\text{Eff}_{\text{Skill}}(\text{Path}^i) = \frac{N_{\text{Path}^i_{\mathcal{G}_{\text{Ori}}*}=1, \text{Path}^i_{\mathcal{G}_{\text{Bkg}}*}=0, \text{Path}^i_{\mathcal{G}_{\text{Self}}*}=0}}{N_{all}} \tag{3}$$

Finally, we get all skill paths and compose them into a Skill Graph $\mathcal{G}^S = \{\mathcal{C}, \mathcal{E}^S\}$. With $\delta$ as the threshold parameter: $\mathcal{E}^S = \{\text{Path}^i | \text{Eff}_{\text{Skill}}(\text{Path}^i) > \delta\}$ (we provide a detailed analysis of $\delta$ in Appendix D).

## 4 Experimental Design

This paper focuses on 3 language skills, from basic to high:

- **Previous Token Skill**: Receive information from the previous token.

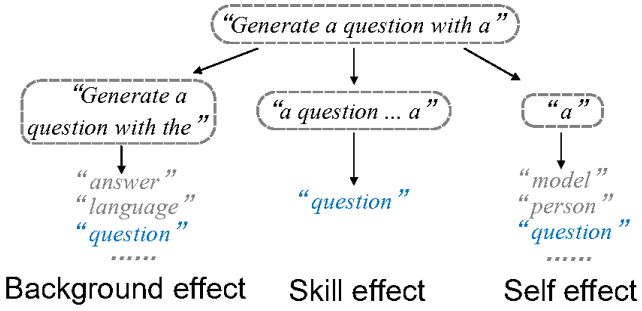



Figure 2: A case text about causal effects.

- **Induction Skill**: Identify patterns in prefix matching and replicate recurring token sequences.
- **ICL Skill**: Recognize and replicate the demonstration, thereby producing outputs based on similar patterns.

Extensive research has shown that these three skills build on one another in a sequentially encompassing manner (Crosbie and Shutova 2024; Olsson et al. 2022; Ren et al. 2024; Edelman et al. 2024). The Induction Skill inherently includes the Previous Token Skill. In simple terms, for induction to occur in the sequence "*A B ... A*", the token *B* must retrieve information from the preceding token *A*. Likewise, ICL must be capable of identifying similar patterns across different demonstrations to generate analogous outputs.

We selected more than 10k samples that encompass one of the three skills mentioned above from corpora such as WIKIQA (Yang, Yih, and Meek 2015), SST-2 (Socher et al. 2013), BIG-BENCH (Srivastava, Rastogi, and Abhishek Rao 2023), OpenOrca (Lian et al. 2023), and OpenHermes (Teknium 2023). For each instance, we create a corrupted text of the background effect and a corrupted text of self effect (discussed in Section 3.4). For simplicity, **PVT** represents the sample set involving the Previous Token Skill and **IDT** represents the sample set related to Induction Skill. **ICL1** represents the ICL sample set from SST-2 datasets; **ICL2** represents the ICL sample set from object_counting task; **ICL3** and **ICL4** represents those from qawikidata and reasoning_about_colored_objects task. The details of data preparation and implementation are elaborated in Appendix D.

## 5 Validation

Since skill paths do not encompass all the skill mechanisms within a circuit graph, they function more as a sub-circuit and are unable to preserve the faithfulness of the original output. A complete circuit should include three: the background, skill, and self effects. However, the skill path only encompasses the skill effect, and therefore, it is unable to fully recover the output. This makes existing circuit justification methods (such as the KL divergence and task accuracy between the skill path and the original output of the model forward) **unfeasible**. Fortunately, the linearly decoupled computational graph can recover the model's output losslessly with complete theoretical support, and the adopted pruning strategies have also been thoroughly validated in existing research. Therefore, **this paper focuses on verifying whether the skill paths**

**can effectively reflect the mechanism of the target skill rather than the task dataset**. Specifically, we conducted the following three **Validation** experiments.

### 5.1 Path Ablation

To understand whether the skill paths are responsible for the corresponding language skills, we design an intervention experiment by removing different sets of paths and observe the output of the LM.

Table 2 displays the accuracy under different configurations of the Circuit Graphs when treating the original output as ground truth. For each language skill $S$, we randomly select 500 samples from its corresponding dataset. As a result, 9 different configurations of Circuit Graphs are tested: $\mathcal{G}*$ which represents the original output; $-R50$ which signifies the removal of 50 paths at random from $\mathcal{G}*$; $-R500$ after the deletion of 500 paths randomly from $\mathcal{G}*$, which approximately equals the number of skill paths[1]. The remaining 6 configurations encompass the removal of paths from $\mathcal{G}*$ that correspond to the skill of Previous Token, Induction, ICL1, ICL2, ICL3, and ICL4, respectively (For additional data for this validation test, please refer to Appendix E.).

The results indicate that almost all samples were unable to produce the original token when these skill paths were excluded (as indicated in the last 6 columns), yet random removal of paths does not lead to such significant impact. Additionally, Figure 3 visualizes the t-SNE representation of the outputs associated with different Circuit Graphs. It is clear that when a skill path is removed, the output (blue) shifts from red toward green (or yellow), indicating a transition from a text output distribution that includes skills to a distinct space resulted from the removal of these skills.

### 5.2 Validation with Benchmark Task

In Appendix F, we performed validation on samples from the benchmark task, IOI. Using our method, we discovered the paths of some confirmed skills in the IOI circuit graph, including duplication skill, previous token skill, induction skill, inhibition skill, and name mover skill. We verified that the inclusion relationships of these skills are consistent with the edges in the original circuit graph. For instance, the paths of the induction skill include those of the previous token skill, and the paths of the inhibition skill include those of the induction skill, etc.

### 5.3 Validation for Effects of Other skills

In Appendix G, we visualized the probability density functions of the paths corresponding to the three effects (skill effects, background effects, and self effects). The experimental results confirmed the existence of high-frequency paths of background effect and self effect in the original circuit graph, which act as confounders affecting the extraction of skill paths. It also demonstrated that counterfactual and intervention corrupted texts can effectively reduce the influence of background effect and self effect.

[1]The exact number of removed paths is: $\mathcal{G}^{S,\text{PVT}} - 325$, $\mathcal{G}^{S,\text{IDT}} - 466$, $\mathcal{G}^{S,\text{ICL1}} - 589$, $\mathcal{G}^{S,\text{ICL2}} - 622$, $\mathcal{G}^{S,\text{ICL3}} - 603$, $\mathcal{G}^{S,\text{ICL4}} - 537$.

## 6 Discovery of Language Skills

Two conjectures about language skills have been increasingly recognized. They are:

1. **Stratification**: Simpler language skills reside in shallower layers, whereas more complex language skills are found in deeper layers.
2. **Inclusiveness**: More complex language skills are formed on top of simpler skills.

For **Stratification** and **Inclusiveness**, many works have already discovered their traces. For instance, Wang et al. (2023) uses causal tracing to show that the previous token head constitutes a subcomponent of the induction circuit in the shallow layers, while Crosbie and Shutova (2024); Olsson et al. (2022) demonstrates that the ICL capability emerges as a higher-order skill built upon induction. Taken together, these studies support the notion that these three skills form a progression of increasing complexity.

However, so far, there has been no **quantitative** evidence for these 2 conjectures. E.g., in which specific layers do simple skills reside? What specific paths in simple skills are encompassed by complex skills? Our work further confirms these via quantitative discoveries.

### 6.1 Quantitative Results

Table 3 displays the nodes (receivers) receiving more than 10 paths in the skill graphs. We use $[l, i]$ to denote the $l$-th layer and $i$-th components ($C^{l,i}$). The complete Skill Graph can be found in Appendix M. From Table 3, we provide quantification results for Stratification and Inclusiveness.

**Quantification of Stratification**: The Previous Token Skill (PVT) is one of the simplest language skills and is located across layers 0-2. The Induction Skill (IDT) is slightly more complex, and thus spreads across layers 0-6. Meanwhile, ICL is the most complex skill and has key receivers across nearly all layers.

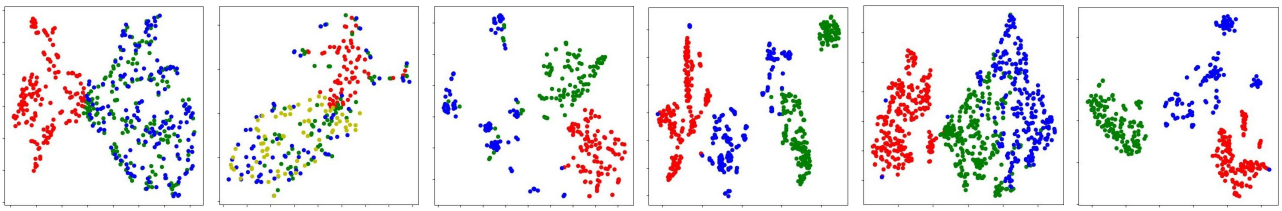
**Quantification of Inclusiveness**: Components such as [2, 14], [2, 18], and [2, 20] (presented in PVT) can be found in the IDT, indicating that the PVT is an integral part of the IDT. Similarly, the ICL skill encapsulates the PVT and IDT as necessary sub-skills, which is why components that are evident in the PVT (such as [2, 14], [2, 20], [2, 24]) and those identified in the IDT (such as [3, 14], [4, 5]) can be found in the ICL Skill Graph. Furthermore, we list all multi-step paths with inclusive sub-path in Appendix K.

### 6.2 Comparison among Pruning Strategies

We investigate the performance of our framework replacing different pruning strategies in validating the 2 conjectures. We investigate existing pruning strategies (ACDC (Conmy et al. 2023), E-pruning (Bhaskar et al. 2024), EAP (Syed, Rager, and Conmy 2023), DiscoGP (Yu et al. 2024), Scrubbing (Chan et al. 2022)), which are prevalent methods for circuit pruning, as detailed in Appendix H. Specifically, we replace our original pruning process with each pruning strategy mentioned above and test on the following three skills: PVT, IDT, and ICL1. Then, we investigate the distribution of the receiver nodes among layers in these circuit graphs and display the normalized results in Figure 4.

| Sample | Circuit Graph | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mathcal{G}*$ | $-R50$ | $-R500$ | $-\mathcal{G}^{S,\text{PVT}}$ | $-\mathcal{G}^{S,\text{IDT}}$ | $-\mathcal{G}^{S,\text{ICL1}}$ | $-\mathcal{G}^{S,\text{ICL2}}$ | $-\mathcal{G}^{S,\text{ICL3}}$ | $-\mathcal{G}^{S,\text{ICL4}}$ |
| PVT | 1.00 | 0.46 | 0.23 | 0.01 | 0.00 | 0.00 | 0.01 | 0.00 | 0.00 |
| IDT | 1.00 | 0.58 | 0.29 | 0.08 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 |
| ICL1 | 1.00 | 0.61 | 0.23 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ICL2 | 1.00 | 0.51 | 0.18 | 0.00 | 0.00 | 0.01 | 0.00 | 0.01 | 0.01 |
| ICL3 | 1.00 | 0.54 | 0.21 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ICL4 | 1.00 | 0.62 | 0.30 | 0.07 | 0.03 | 0.01 | 0.02 | 0.00 | 0.00 |

Table 2: Accuracy of output to original label within different path removing

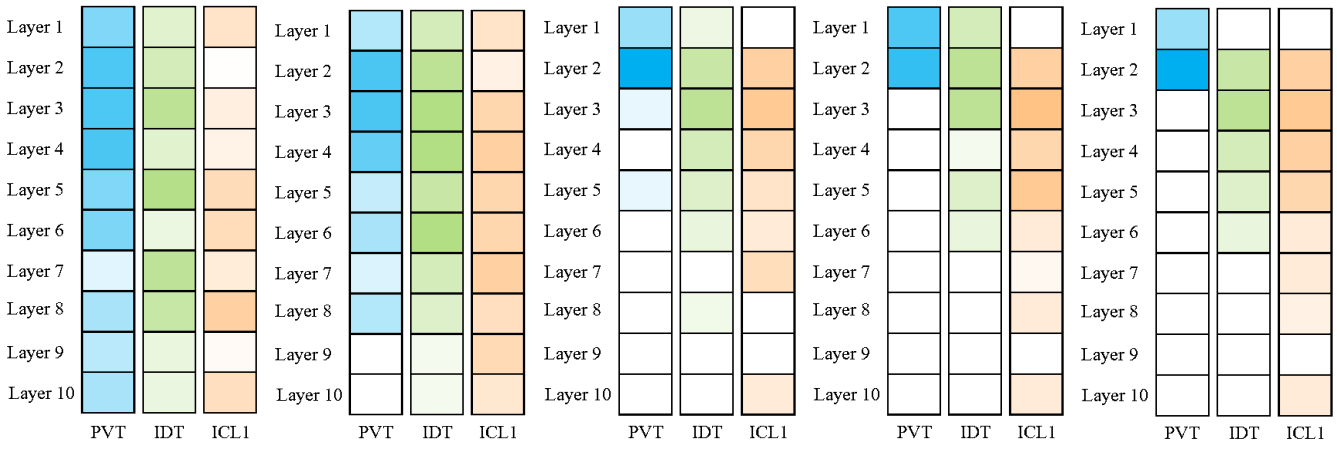


(a) PVT (b) IDT (c) ICL1 (d) ICL2 (e) ICL3 (f) ICL4

Figure 3: T-sne visualization of 6 types of samples on vocabulary candidates. **Red** denotes the original output model ($\mathcal{G}$), while **blue** signifies the output once a corresponding skill path is removed ($\mathcal{G} - \mathcal{G}^S$). The outputs for the background text ($\mathcal{G}_{\text{Bkg}}$) and self text ($\mathcal{G}_{\text{Self}}$) are indicated in **green** and **yellow**, respectively.

| Skill | Receivers receiving more than 10 paths ([#layer, #components]) |
|---|---|
| **PVT** | [1, 8], [1, 18], [1, 19], [1, 20], [1, 21], [2, 1], [2, 7], [2, 14], [2, 18], [2, 20], [2, 22], [2, 24], [11, 1], [11, 14] |
| **IDT** | **[2, 14]**, **[2, 18]**, **[2, 20]**, [3, 14], [3, 17] [4, 5], [4, 12], [5, 11], [6, 5], **[11, 1]**, **[11, 14]** |
| **ICL1** | **[2, 14]**, **[2, 20]**, **[2, 22]**, **[2, 24]**, [3, 3], [3, 4], [3, 5], [3, 11], **[3, 14]**, **[3, 17]**, [4, 3], **[4, 5]**, **[5, 11]**, [8, 5], [10, 10], [11, 8], [11, 9], [11, 10], [11, 11] |
| **ICL2** | **[1, 19]**, **[2, 14]**, **[2, 20]**, **[2, 24]**, [3, 5], [3, 11], **[3, 14]**, **[4, 5]**, [4, 7], [4, 9], [5, 10], **[6, 5]**,[10, 9], [10, 10], [10, 11], **[11, 1]**, [11, 5] |
| **ICL3** | **[1, 8]**, **[1, 18]**, **[1, 19]**, **[1, 20]**, **[1, 21]**, **[2, 14]**, **[2, 20]**, **[2, 24]**, [3,1], **[3, 14]**, [4, 3], **[4, 5]**, [5, 1], [5, 10], **[5, 11]**, [8, 1], [8, 9], [10, 5], [10, 10], [10, 12], **[11, 1]**, [11, 8] |
| **ICL4** | [1, 16], **[1, 20]**, **[2, 20]**, [4, 3], **[4, 5]**, [5, 3], [6, 4], **[6, 5]**, [8, 9], [9, 4], [9, 5], [10, 2], [10, 10], [10, 12], [11, 2], [11, 3], [11, 4], [11, 6], [11, 15] |

Table 3: Key Receivers in skill paths, **bold** components are presented in the lower skill.




(a) E-pruning (b) EAP (c) DiscoGP (d) Scrubbing (e) ACDC

Figure 4: Visualization of receivers distributed in layer1-10 in 3 increasingly-complex skills (PVT, IDT, and ICL1).

| Types | Method | ovlp(IDT, PVT) | ovlp(ICL1, PVT) | ovlp(ICL1, IDT) | ovlp(GT,PVT) | ovlp(GT,IDT) | ovlp(GT,ICL1) |
|---|---|---|---|---|---|---|---|
| Circuit | ACDC | 0.19 | 0.06 | 0.17 | 0.11 | 0.05 | 0.11 |
| | E-pruning | 0.05 | 0.14 | 0.17 | 0.18 | 0.10 | 0.05 |
| | EAP | 0.14 | 0.05 | 0.18 | 0.09 | 0.15 | 0.12 |
| | DiscoGP | 0.18 | 0.07 | 0.15 | 0.08 | 0.11 | 0.09 |
| | Scrubbing | 0.10 | 0.08 | 0.17 | 0.16 | 0.11 | 0.10 |
| Paths | ACDC+Ours | 0.74 | 0.81 | 0.63 | 0.68 | 0.13 | 0.11 |
| | E-pruning+Ours | 0.67 | 0.78 | 0.59 | 0.67 | 0.09 | 0.14 |
| | EAP+Ours | 0.70 | 0.68 | 0.67 | 0.74 | 0.17 | 0.05 |
| | DiscoGP+Ours | 0.79 | 0.81 | 0.71 | 0.77 | 0.04 | 0.13 |
| | Scrubbing+Ours | 0.74 | 0.77 | 0.68 | 0.71 | 0.08 | 0.13 |

Table 4: Overlaps between different circuits and between different paths

It is clear that in general PVT is more prominent in the shallow layers, while IDT in the shallow to mid layers. ICL1 tends to cluster in the deep layers. Yet, the paths discovered from the circuits via pruning strategies - DiscoGP, Scrubbing, and ACDC - provide distinct patterns from those using E-pruning and EAP. From (c), (d) and (e), PVT circuits appear almost exclusively in layers 1 and 2, and IDT circuits appear only in layers 1-6. The differences in these pruning methods indicate that circuits from greedy search (ACDC), continuous optimization (DiscoGP), and path patching (Scrubbing) are more conducive to reflecting the stratification in skill paths.

### 6.3 Skill Path vs. Circuit Graph

To further interpret how different methods capture **Inclusiveness** and compare between existing circuit graphs and our skill paths, we investigate the overlap of circuits identified for the 4 skills: PVT, IDT, ICL1, and greater than (GT) in Table 4. GT, with input samples drawn from the *greater than* dataset (Hanna, Liu, and Variengien 2024), assesses the ability of the language model to judge numerical relations. An example from the GT dataset is "The war lasted from 1517 to 15?". To demonstrate the difference between skill paths and circuit graphs, we use the circuit graphs obtained by these methods themselves (rows 2-6) and the skill paths obtained using their pruning strategies in combination with our framework (rows 7-11), to calculate the overlap between skills. Here we use $ovlp(A, B)$ (detailed in Appendix H) to indicate the ratio of edges in circuit/paths $A$ that also exist in $B$. A value of 0 means that no edges are shared, and 1 means all edges are shared.

As discussed in Section 6.1, the three skills PVT, IDT, and ICL are progressively inclusive, that is, IDT includes PVT, while ICL includes both IDT and PVT. In addition, GT only includes PVT and has no relation to IDT and ICL. Table 4 shows that the overlap of circuit graphs discovered by existing methods is rather weak. For example, $ovlp(\text{ICL1}, \text{IDT})$ is only 0.17 in ACDC. In contrast, our approach provides clear empirical evidence for the conjecture of inclusiveness: for instance, $ovlp(\text{IDT}, \text{PVT}) = 0.74$ indicates that 74% of the edges in the Induction skill exist in the previous token skill. Additionally, from the experimental results related to GT, it can be seen that for skills without inclusiveness, our skill paths can also reflect significant differences.

In addition to the main experiments, we conducted **two comparison studies** to further understand the skill paths: one, detailed in Appendix I, compared the dynamic results on the number of paths, KL divergence, and task accuracy as the threshold $\delta$ varied from 0 to 1, revealing that the optimal trade-off among these variables occurs when $\delta = 0.6 - 0.7$; the other, described in Appendix J, involved designing different corrupted texts to assess the background effect, with results indicating that the background effects were similar.

Finally, we present an analysis of why the model fails to execute the skills in Appendix L. It demonstrates that through the stratification and inclusiveness of our skill paths, we can explain that the failure of an LLM to execute a high-level skill is due to the failure of a certain basic skill. The limitations are shown in Appendix N.

## 7 Limitation

We identify 3 limitations that need to be addressed in the future: 1) the **time complexity** of our framework; 2) **scalability**; 3) the lack of further examination of the **representational study**.

Assuming the time for an inference using LLM as $O(1)$, the time complexity of a search round would then be $O(L^2N^2)$, i.e., the square of the number of layers times the number of components. If we can overlook this time-consuming process, $\mathcal{G}*$ for each input would effectively facilitate training. In other words, $\mathcal{G}*$ could directly instruct LLM which paths are essential and which are not, thus reducing the training process. Despite the time complexity, we recall our contribution to the analysis of LMs, which is usually more challenging and does not require large-scale inference.

We also recognize the limitations of testing on a single model and specific skills. Although many studies have validated the GPT-2 series to have public trustworthiness for mechanistic interpretability research, making us confident in its ability to support our contribution, pioneering work in discovering the theoretical foundation and experimental design of language skills, there remains ample scope for scalability across a variety of models and skills for future work.

Finally, the lack of research at the representational level hinders our progress in answering more complex questions such as why certain samples fail to trigger a skill. Recognized that this is a rather challenging topic, we leave it as a promising future work.

## References


Arora, S.; and Goyal, A. 2023. A Theory for Emergence of Complex Skills in Language Models. arXiv:2307.15936.

Bhaskar, A.; Wettig, A.; Friedman, D.; and Chen, D. 2024. Finding transformer circuits with edge pruning. *arXiv preprint arXiv:2406.16778*.

Chan, L.; Garriga-Alonso, A.; Goldwosky-Dill, N.; Greenblatt, R.; Nitishinskaya, J.; Radhakrishnan, A.; Shlegeris, B.; and Thomas, N. 2022. Causal scrubbing, a method for rigorously testing interpretability hypotheses. *AI Alignment Forum*. https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing.

Conmy, A.; Mavor-Parker, A.; Lynch, A.; Heimersheim, S.; and Garriga-Alonso, A. 2023. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36: 16318–16352.

Crosbie, J.; and Shutova, E. 2024. Induction heads as an essential mechanism for pattern matching in in-context learning. *arXiv preprint arXiv:2407.07011*.

Edelman, B. L.; Edelman, E.; Goel, S.; Malach, E.; and Tsilivis, N. 2024. The evolution of statistical induction heads: In-context learning markov chains. *arXiv preprint arXiv:2402.11004*.

Elhage, N.; Nanda, N.; Olsson, C.; Henighan, T.; Joseph, N.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; DasSarma, N.; Drain, D.; Ganguli, D.; Hatfield-Dodds, Z.; Hernandez, D.; Jones, A.; Kernion, J.; Lovitt, L.; Ndousse, K.; Amodei, D.; Brown, T.; Clark, J.; Kaplan, J.; McCandlish, S.; and Olah, C. 2021. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Geva, M.; Schuster, R.; Berant, J.; and Levy, O. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 5484–5495.

Goldowsky-Dill, N.; MacLeod, C.; Sato, L.; and Arora, A. 2023. Localizing Model Behavior with Path Patching. arXiv:2304.05969.

Hanna, M.; Liu, O.; and Variengien, A. 2024. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36.

Lian, W.; Goodson, B.; Pentland, E.; Cook, A.; Vong, C.; and "Teknium". 2023. OpenOrca: An Open Dataset of GPT Augmented FLAN Reasoning Traces. https://https://huggingface.co/Open-Orca/OpenOrca.

Lindner, D.; Kramár, J.; Rahtz, M.; McGrath, T.; and Mikulik, V. 2023. Tracr: Compiled Transformers as a Laboratory for Interpretability. *arXiv preprint arXiv:2301.05062*.

Mueller, A. 2024. Missed Causes and Ambiguous Effects: Counterfactuals Pose Challenges for Interpreting Neural Networks. arXiv:2407.04690.

Olsson, C.; Elhage, N.; Nanda, N.; Joseph, N.; DasSarma, N.; Henighan, T.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; Drain, D.; Ganguli, D.; Hatfield-Dodds, Z.; Hernandez, D.; Johnston, S.; Jones, A.; Kernion, J.; Lovitt, L.; Ndousse, K.; Amodei, D.; Brown, T.; Clark, J.; Kaplan, J.; McCandlish, S.; and Olah, C. 2022. In-context Learning and Induction Heads. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Ren, J.; Guo, Q.; Yan, H.; Liu, D.; Qiu, X.; and Lin, D. 2024. Identifying semantic induction heads to understand in-context learning. *arXiv preprint arXiv:2402.13055*.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Srivastava, A.; Rastogi, A.; and Abhishek Rao, e. a. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Syed, A.; Rager, C.; and Conmy, A. 2023. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*.

Teknium. 2023. OpenHermes 2.5: An Open Dataset of Synthetic Data for Generalist LLM Assistants.

Vig, J.; Gehrmann, S.; Belinkov, Y.; Qian, S.; Nevo, D.; Singer, Y.; and Shieber, S. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33: 12388–12401.

Wang, K. R.; Variengien, A.; Conmy, A.; Shlegeris, B.; and Steinhardt, J. 2023. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small. In *The Eleventh International Conference on Learning Representations*.

Yang, Y.; Yih, W.-t.; and Meek, C. 2015. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2013–2018. Lisbon, Portugal: Association for Computational Linguistics.

Yao, Y.; Zhang, N.; Xi, Z.; Wang, M.; Xu, Z.; Deng, S.; and Chen, H. 2024. Knowledge Circuits in Pretrained Transformers. *arXiv preprint arXiv:2405.17969*.

Yu, L.; Niu, J.; Zhu, Z.; and Penn, G. 2024. Functional faithfulness in the wild: Circuit discovery with differentiable computation graph pruning. *arXiv preprint arXiv:2407.03779*.