

# WASD: Locating Critical Neurons as Sufficient Conditions for Explaining and Controlling LLM Behavior

Haonan Yu<sup>1</sup>, Junhao Liu<sup>2</sup>, Zhenyu Yan<sup>3</sup>, Haoran Lin<sup>4</sup>,  
Xin Zhang\*<sup>5</sup>,

Key Lab of High Confidence Software Technologies (Peking University), Ministry of Education  
School of Computer Science, Peking University, Beijing, China  
{a616156<sup>1</sup>, zhenyuyan<sup>3</sup>, haoranlin<sup>4</sup>}@stu.pku.edu.cn, {liujunhao<sup>2</sup>, xin<sup>5</sup>}@pku.edu.cn

## Abstract

Precise behavioral control of large language models (LLMs) is critical for complex applications. However, existing methods often incur high training costs, lack natural language controllability, or compromise semantic coherence. To bridge this gap, we propose WASD (unWeaving Actionable Sufficient Directives), a novel framework that explains model behavior by identifying sufficient neural conditions for token generation. Our method represents candidate conditions as neuron-activation predicates and iteratively searches for a minimal set that guarantees the current output under input perturbations. Experiments on SST-2 and CounterFact with the Gemma-2-2B model demonstrate that our approach produces explanations that are more stable, accurate, and concise than conventional attribution graphs. Moreover, through a case study on controlling cross-lingual output generation, we validated the practical effectiveness of WASD in controlling model behavior.

## 1 Introduction

As large language models (LLMs) become integral to complex applications, there is a critical demand for precise behavioral control. Existing black-box approaches, such as alignment fine-tuning, struggle to guarantee reliable outcomes under novel conditions. On the other hand, mechanistic interpretability (Naseem, 2026; Kowalska and Kwaśnicka, 2025), particularly circuit analysis, has recently gained significant popularity as a paradigm for reverse-engineering the internal mechanisms of neural networks. Among existing approaches, Circuit Tracer (Ameisen et al., 2025) represents the state of the art, identifying computational pathways by tracing the linear contributions of individual components to a final output token.

However, these methods are attribution-based, which face significant limitations. High attribution guarantees neither necessity nor sufficiency: components with massive linear contributions are not

inherently causally required for the target token. Furthermore, these dense graphs are highly unstable under minor input perturbations and lack the clarity of rule-based explanations (Ribeiro et al., 2018).

Figure 1 (left) illustrates these shortcomings, demonstrating that using Circuit Tracer to control output language is unreliable. For example, when intervening on the top-five neurons in the target language’s attribution graph to complete a phrase, this baseline frequently fails, generating either the original input language or incoherent text.

To resolve these limitations and enable the precise manipulation of LLMs, we propose a framework named unWeaving Actionable Sufficient Directives from tangled circuit graphs (WASD). Rather than relying on linear contributions, WASD identifies a minimal set of neural predicates—each defined by a neuron and its activation constraint—whose satisfaction is sufficient to maintain the target output token. To avoid the prohibitive computational cost of a direct search, the attribution scores from Circuit Tracer are leveraged as a heuristic. By ranking neurons according to their contribution, a greedy search procedure is guided to iteratively incorporate predicates into candidate explanations, efficiently extracting sufficient conditions.

Returning to the example, Figure 1 (right) shows how WASD successfully enforces the target language constraint. By intervening directly on the identified neural predicates, the model consistently and accurately generates text in the specified language, regardless of the input.

WASD is evaluated on the SST-2 and CounterFact datasets using the Gemma-2-2B model. Results demonstrate that the produced explanations are significantly more stable, accurate, and concise than traditional attribution graphs. Furthermore, the framework is extended to enforce broader behavioral conditioning. By identifying

## Controlling Model Output Language

Baseline: Circuit Tracer			WASD 		
Input	Target	Output	Input	Target	Output
 A journey of a thousand miles	Spanish	starts with a single step. ❌	 A journey of a thousand miles	Spanish	comienza con un paso. ✅
 Un viaje de mil millas	French	commence con un paso. ❌	 Un viaje de ml millas	French	commence avec un pas. ✅
 Un voyage de mille lieues	Chinese	c'est un voyage de mille lieues. ❌	 Un voyage de mille lieues	Chinese	,始于一步. ✅
 千里之行,	English	börder først en fot. ❌	 千里之行,	English	beginnings with the first step. ✅

Figure 1: Comparison of language control capabilities between WASD and Circuit Tracer. In each subplot, the left panel displays the input, and the right panel displays the model’s output following intervention. WASD controls the output by fixing specific neurons associated with the target language. In contrast, Circuit Tracer fixes the five highest-scoring neurons derived from the target language prompt’s contribution map.

and fixing neurons that guarantee a specific constraint—such as Chinese generation—the model consistently produces the desired output across diverse cross-lingual inputs.

In summary, the primary contributions of this work are threefold:

- The proposal of WASD, a novel mechanistic interpretability framework that explains model behavior by extracting sufficient neural conditions for token generation.
- Verification through experiments on multiple datasets, demonstrating that the sufficient conditions identified by WASD outperform Circuit Tracer in terms of accuracy, stability, and simplicity.
- A case study showing that the neurons identified by WASD can be used to control model outputs, such as ensuring consistent cross-lingual text generation.

## 2 Background

This section provides the necessary background knowledge that is integral to understanding our approach. We first briefly introduce the architecture of Transformer and the Circuit Tracer, then give a formal definition of the output of Circuit Tracer, and finally give a formal definition of the important components introduced by our method.

### 2.1 Architecture of Transformer

Transformer-based architectures (Vaswani et al., 2017) form the foundation of modern large language models. A Transformer processes an input sequence of tokens by mapping each token to a vector embedding and passing these representations through a stack of layers. Each layer typically consists of a multi-head self-attention module followed by a feed-forward network (often implemented as a multi-layer perceptron, or MLP), with residual connections and layer normalization applied between components.

During inference, these embeddings propagate through the layers. The final hidden representation is projected into a logit vector over the vocabulary, and a softmax function determines the next token’s probability distribution. Understanding how intermediate neurons and attention mechanisms contribute to these final logits is a core challenge in mechanistic interpretability.

### 2.2 Circuit Tracer and Attribution Graphs

A recent approach to mechanistic interpretability is Circuit Tracer, proposed by Anthropic, which aims to reveal the computational pathways responsible for specific model outputs.

**Replacement Model.** Circuit Tracer constructs an interpretable replacement model that approximates the behavior of a Transformer while exposing internal computational dependencies. In this approach, parts of the original model are replaced

with more interpretable components that are trained to approximate the original computation. This replacement model enables tracing how intermediate activations influence one another and ultimately affect the output. Using this model, Circuit Tracer decomposes the computation into a set of interactions between interpretable units, enabling the construction of a linear attribution graph describing how information flows through the network during inference on a specific prompt. Importantly, the replacement model makes the features and neurons of the model more correlated, meaning that a single neuron often only represents a single feature. As a result, it becomes possible to analyze the model’s overall behavior by examining just a small fraction of neurons.

**Attribution Graph.** The output of Circuit Tracer is an attribution graph, which provides a structured representation of the computation performed by the model on a given prompt. Formally, the attribution graph can be represented as a directed weighted graph

$$G = (V, E)$$

where  $V$  denotes a set of nodes corresponding to neurons (or other computational units, such as input and output tokens) in the model. Each neuron  $v \in V$  has an associated activation value  $a_v$ .  $E$  denotes directed edges representing influence relationships between neurons.

Each edge  $(u, v) \in E$  has an associated contribution weight  $c_{u \rightarrow v}$ , which measures how the activation of neuron  $(u)$  contributes to the activation of neuron  $(v)$ . In addition, neurons may have direct contributions to the output logits, representing their influence on the probability of generating specific tokens. This graph captures the causal structure of computation for a specific input prompt, allowing researchers to identify important neurons and pathways responsible for particular model behaviors.

### 2.3 Sufficient Conditions for Model Behaviors

To analyze and control model behaviors, we introduce the following formal definitions.

Let  $\mathcal{V}$  denote a finite vocabulary of tokens.

**Definition 2.1 (Prompt).** A prompt  $x$  is a finite sequence of tokens  $x = (t_1, t_2, \dots, t_n) \in \mathcal{V}^n$ , where  $n \in \mathbb{N}$ . The space of all possible prompts is denoted as  $\mathcal{X} = \bigcup_{n \in \mathbb{N}} \mathcal{V}^n$ .

Let  $f : \mathcal{X} \rightarrow \mathcal{V}$  represent a language model. Let  $\mathcal{H}$  be the set of all neurons in  $f$ , and let  $\Phi_v(x) \in \mathbb{R}$

denote the activation value of a specific neuron  $v \in \mathcal{H}$  during the forward pass of prompt  $x$ . Furthermore, let  $C_v(x) \in \mathbb{R}$  denote the contribution of neuron  $v$  to the logits of the output token.

Since our method only requires the neuron activation values and their contributions to the output token obtained by the Circuit Tracer, we formally define it as a function mapping a prompt to these specific internal states:

**Definition 2.2 (Weight Extraction).** The weight extraction function *WtExt* takes a prompt  $x \in \mathcal{X}$  as input and outputs a set containing the activation value and the corresponding logit contribution for each neuron:

$$WtExt(x) = \{(\Phi_v(x), C_v(x)) \mid v \in \mathcal{H}\}$$

**Definition 2.3 (Predicate).** A predicate  $p$  is a tuple  $(v, a) \in \mathcal{H} \times \mathbb{R}$ , denoted symbolically as  $p_{v,a}$ , representing the logical constraint  $\Phi_v(x) = a$ .

**Definition 2.4 (Rule).** A rule  $r$  is a finite set of predicates defining a logical conjunction. For an index set  $K = \{1, \dots, k\}$ , a rule is defined as:

$$r := \bigwedge_{i \in K} p_{v_i, a_i}$$

Let  $\Omega$  denote a set of discrete edit operations (deletion or replacement).

**Definition 2.5 (Neighborhood).** For a given prompt  $x \in \mathcal{X}$ , the neighborhood  $\mathcal{N}(x)$  is defined as:

$$\mathcal{N}(x) = \{x' \in \mathcal{X} \mid x' \text{ is derived from } x \text{ via } \Omega\}$$

Let  $do(r)$  denote intervention, which hardcodes the internal model state such that  $\forall p_{v_i, a_i} \in r, \Phi_{v_i}(\cdot) = a_i$ . Let  $f(\cdot \mid do(r))$  represent the model function operating under this fixed intervention. Let  $\mathcal{D}_{\mathcal{N}}$  be a probability measure over  $\mathcal{N}_\epsilon(x)$ .

**Definition 2.6 (Precision).** The precision of a rule  $r$  with respect to a baseline prompt  $x$ , denoted  $Prec(r, x)$ , is the probability of output invariance under the intervention  $do(r)$  across the local neighborhood:

$$Prec(r, x) = \mathbb{P}_{x' \sim \mathcal{D}_{\mathcal{N}}} [f(x' \mid do(r)) = f(x)]$$

**Definition 2.7 (Sufficiency).** Given a threshold  $\tau \in (0, 1]$ , a rule  $r$  constitutes a sufficient condition for the model’s behavior on  $x$  if:  $Prec(r, x) \geq \tau$

---

**Algorithm 1** Generating Predicates

---

**Input:** Original prompt  $x = (t_1, t_2, \dots, t_n)$ , the model to explain  $f$

**Parameter:** The scaling factor  $\lambda$

**Output:** Set of candidate predicates with contributions  $\mathbb{P}$

```
1:  $\mathcal{N}(x) = \text{Perturbation}(x)$ 
2: Initialize activation map  $A = \{\}$ 
3: Initialize contribution map  $C = \{\}$ 
4: # Calculate activations and contributions
5: for  $x' \in \mathcal{N}(x)$  do
6:   Activations, Contributions =  $\text{WtExt}(x')$ 
7:   for Each neuron  $n$  do
8:      $A[n].\text{add}(\text{Activations}[n])$ 
9:      $C[n].\text{add}(\text{Contributions}[n])$ 
10:  end for
11: end for
12: Initialize predicates  $\mathbb{P} = \{\}$ 
13: # Formulate predicates
14: for Each neuron  $n$  do
15:    $A_{max,n} = \text{MAX}(A[n])$ 
16:    $p_n = A_{max,n} * \lambda$ 
17:    $c_n = \text{MEAN}(C[n])$ 
18:    $\text{Add}(p_n, c_n)$  to  $\mathbb{P}$ 
19: end for
20: Return  $\mathbb{P}$ 
```

---

Our goal is to identify a minimal set of jointly active neurons to maintain a decision, which can be formalized as the following optimization problem:

$$\arg \min_r |r| \quad \text{subject to} \quad \text{Prec}(r, x) \geq \tau$$

### 3 Methodology

This section outlines WASD, our proposed method for extracting the sufficient conditions that govern model behavior. As illustrated in Figure 2, the methodology operates in two primary phases: generating candidate predicates via prompt perturbation and activation tracking, followed by an attribution-guided heuristic search and iterative reduction loop. Together, these steps construct a minimized rule set that is both concise and sufficient to maintain model precision.

#### 3.1 Generating Predicates

Algorithm 1 details the predicate generation process. Inspired by traditional perturbation-based interpretation techniques like LIME (Ribeiro et al.,

2016) and Anchors (Ribeiro et al., 2018), a local neighborhood is first constructed by editing the input prompt (Line 1), and maps are initialized to track activations and logit contributions (Lines 2–3). For each perturbed prompt, Circuit Tracer computes the attribution graph (Line 6), recording individual neuron activations and contributions across varying inputs (Lines 8–9). From these statistics, candidate predicates are constructed. Each neuron’s maximum recorded activation is multiplied by a scaling factor  $\lambda$  to define its predicate threshold. This predicate, alongside the neuron’s average contribution, is then stored in the candidate predicate set  $\mathbb{P}$  (Lines 14–19).

Based on these statistics, candidate predicates are constructed. For each neuron, our method identifies its maximum recorded activation and multiply it by a scaling factor  $\lambda$  to define the predicate threshold. The predicate, together with the neuron’s average contribution, is then stored in the candidate predicate set  $\mathbb{P}$  (Lines 14–19).

The hyperparameter  $\lambda$  dictates the granularity of the resulting explanations. However, a larger  $\lambda$  does not strictly yield better performance; an excessively high value can disproportionately amplify a single neuron’s importance, obscuring the synergistic network interactions that typically drive the model’s final prediction. To prevent the method from degrading into an analysis of a single neuron’s preferred output, a grid search strategy is applied to empirically determine the optimal  $\lambda$  that maximizes precision.

#### 3.2 Identifying Sufficient Conditions

Once the candidate predicates are generated, Algorithm 2 employs a contribution-based heuristic search to identify the sufficient conditions controlling the model’s output. The generated predicates are first sorted by descending contribution (Line 1). Next, an empty candidate rule set is initialized, and the baseline precision,  $\text{prec}(\emptyset, x)$ , is calculated within the prompt’s neighborhood without any interventions (Lines 2–3). The rule set is then iteratively expanded by introducing new predicates (Lines 5–6). A predicate is retained only if it improves the precision of the rule set; otherwise, it is discarded (Lines 7–11). This additive phase continues until the current rule set’s precision,  $\text{prec}(r, x)$ , meets or exceeds a predefined threshold  $\tau$  (Lines 12–14). Because the candidate pool encompasses neurons representing all input tokens, the search is guaranteed to converge on a rule that satisfies this

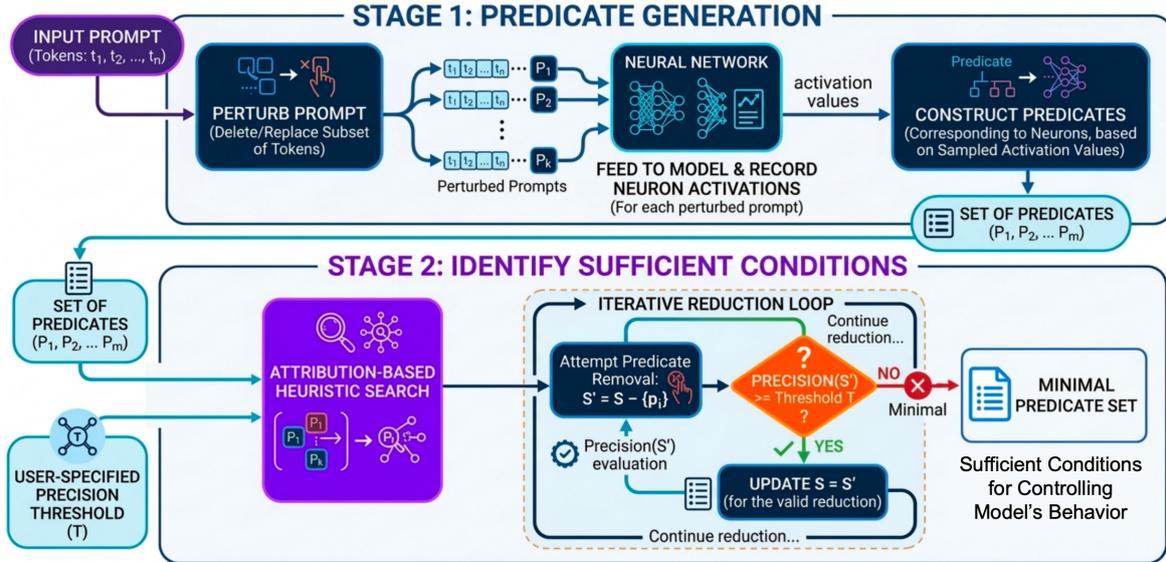


Figure 2: The workflow of WASD.

precision constraint. Finally, to ensure conciseness, a pruning step is performed. The finalized rule set is systematically evaluated to drop one predicate at a time (Line 18). If removing a predicate does not cause the precision to fall below  $\tau$ , it is deleted (Lines 19–21). This strips the rule set of any redundant predicates, ultimately returning a minimized rule that serves as the sufficient condition for controlling the model’s behavior (Line 23).

## 4 Experiment

In this section, we empirically evaluate the effectiveness of our proposed neuron identification method. The experiments are designed to answer two key questions: (1) whether the neurons identified by our method genuinely govern the model’s behavior, and (2) whether our approach provides more stable and concise explanations compared with existing attribution-based methods.

### 4.1 Experimental Setup

To evaluate the generality of the proposed method, experiments are conducted on two distinct tasks. The first task involves sentiment classification utilizing the SST-2 dataset (Socher et al., 2013), where its test set comprising 1,820 sentences is used to generate explanations. The second task focuses on text completion using the CounterFact dataset (Meng et al., 2022), specifically employing its paraphrase\_prompts subset of 2,191 sentences as a test set for explanation generation. All experiments are performed on the Gemma-2-2B model (Team, 2024a).

Across all tasks, the parameters of our method are set to  $\tau = 0.9$  and  $\lambda = 6.5$ . These values are determined through grid search to achieve a good balance between explanation size and fidelity. As a baseline, we compare our approach with the original Circuit Tracer method. To ensure a fair comparison and isolate the effect of neurons’ activations, we fix the neurons identified by Circuit Tracer (Top-3, Top-5, and Top-10 contributors) to their original activation values multiplied by coefficients  $\lambda_3$ ,  $\lambda_5$ , and  $\lambda_{10}$ , respectively. These coefficients were also optimized via grid search to yield the highest average precision for the baseline. This setup allows us to demonstrate that the model’s behavior is governed by the specific set of neurons we identify rather than merely the magnitude of activation values.

The experimental procedure is as follows. First, we feed text prompts into the model and analyze the next-token prediction. Using both our method and the baseline, we identify the neurons that govern the model’s output along with their activation values. We then perturb the input prompts (e.g., by adding neutral prefixes, detailed in Chapter B) while fixing the identified neurons to their original activation states. Finally, we observe whether the model’s output remains consistent despite the external perturbations.

### 4.2 Evaluation Metrics

To quantify the performance of our method, we employ the following three metrics:

---

**Algorithm 2** Identifying Sufficient Conditions

---

**Input:** A prompt  $x$ , the candidate predicates  $\mathbb{P}$

**Parameter:** The precision threshold  $\tau$

**Output:** The minimal sufficient rule set  $r$

```
1: Sort  $\mathbb{P}$  by contribution in descending order.
2: Initialize rule  $r = \emptyset$ 
3: Current_Prec =  $prec(\emptyset, x)$ 
4: # Additive Search
5: for  $(p, c) \in \mathbb{P}$  do
6:    $r_{temp} = r \cup p$ 
7:   New_Prec =  $prec(r_{temp}, x)$ 
8:   if New_Prec > Current_Prec then
9:      $r = r_{temp}$ 
10:    Current_Prec = New_Prec
11:   end if
12:   if Current_Prec  $\geq \tau$  then
13:     BREAK
14:   end if
15: end for
16: # Pruning
17: for  $p \in r$  do
18:    $r_{temp} = r - \{p\}$ 
19:   if  $prec(r_{temp}, x) \geq \tau$  then
20:      $r = r_{temp}$ 
21:   end if
22: end for
23: Return  $r$ 
```

---

**Precision (Fidelity).** As defined in Definition 2.6, Precision measures the fidelity of an explanation. Specifically, it is defined as the probability that the model’s output remains unchanged under perturbed conditions when the identified neurons are fixed to a specific activation values.

**Instability.** Instability measures the consistency of explanations under small input perturbation. We prepend neutral prefixes that do not alter the ground-truth output. Let  $r$  and  $r'$  denote the neuron sets identified for the original prompt and the prefix-augmented prompt, respectively. Instability is defined using the Jaccard distance between these sets:

$$Instability = \frac{|r \cup r'| - |r \cap r'|}{|r \cup r'|}$$

A smaller Jaccard distance indicates that the explanation remains more similar after perturbation, resulting in greater stability.

**Size.** This simply measures the number of neurons required to form the explanation. A smaller

size indicates a more concise and interpretable explanation of the model’s internal logic.

### 4.3 Experiment Results and Analysis

The primary results of our comparative analysis are summarized in Table 1.

As indicated by the experimental data, our proposed method consistently achieves higher perturbation accuracy and superior stability compared to the Circuit Tracer baselines across both datasets.

Specifically, in the sentiment classification task (SST2), our method identified a more compact set of neurons that maintained model output consistency even under significant prompt perturbation. In terms of stability, the Jaccard similarity for our identified neurons remained significantly higher than the Top-K neurons selected by Circuit Tracer when neutral prefixes were introduced. This suggests that our method captures the functional core of the model’s decision-making process rather than just transient activations.

Furthermore, our method achieves these gains while maintaining a smaller Size—often requiring fewer neurons than the Top-10 or even Top-5 baselines to reach higher fidelity. This confirms that our approach effectively filters out noise and isolates the specific neurons that govern the model’s behavior, providing a more concise and reliable explanation of LLM internals.

## 5 Case Study

To further demonstrate the efficacy of our approach, we introduce a feature within WASD that allows users to set custom constraints for the model’s output. Specifically, this feature identifies the neuron-level sufficient conditions required to guarantee that the output belongs to a designated target set.

Technically, we integrate a Llama-3-7B-Instruct (AI@Meta, 2024) model into the WASD pipeline as an automated evaluator to determine whether the target model’s output satisfies the user-defined constraints. Based on this evaluation, the framework locates the precise sufficient conditions needed to enforce arbitrary behavioral rules. In this section, we use cross-lingual generation—specifically, forcing the model to output Chinese when prompted in English—as a case study to validate our method’s capacity for fundamental behavior control.

Method	Task	Precision $\uparrow$	Instability $\downarrow$	Size (Neurons) $\downarrow$
Circuit Tracer (Top-3)	SST2	46.37% ( $\pm 1.45\%$ )	0.166 ( $\pm 0.109$ )	3
Circuit Tracer (Top-5)	SST2	40.88% ( $\pm 1.36\%$ )	0.190 ( $\pm 0.076$ )	5
Circuit Tracer (Top-10)	SST2	30.95% ( $\pm 1.35\%$ )	0.052 ( $\pm 0.037$ )	10
WASD	SST2	<b>91.92% (<math>\pm 1.34\%</math>)</b>	<b>0.048 (<math>\pm 0.056</math>)</b>	1.53 ( $\pm 0.11$ )
Circuit Tracer (Top-3)	CounterFact	48.04% ( $\pm 3.28\%$ )	0.212 ( $\pm 0.041$ )	3
Circuit Tracer (Top-5)	CounterFact	48.69% ( $\pm 3.30\%$ )	0.250 ( $\pm 0.035$ )	5
Circuit Tracer (Top-10)	CounterFact	50.33% ( $\pm 3.41\%$ )	0.260 ( $\pm 0.029$ )	10
WASD	CounterFact	<b>92.60% (<math>\pm 0.69\%</math>)</b>	<b>0.103 (<math>\pm 0.032</math>)</b>	2.82 ( $\pm 0.27$ )

Table 1: Performance comparison between WASD and Circuit Tracer on SST2 and CounterFact datasets (mean  $\pm$  95% confidence interval).

## 5.1 Experimental Setup

We define the target user constraint as: "Ensure the model’s output remains in Chinese." We use Gemma-2-2B as the target model for both interpretation and intervention. The process consists of two steps: The process contains two steps:

**Condition Extraction:** We input several Chinese prompts alongside the specified constraint into the WASD framework. The framework then identifies the sufficient conditions—a specific subset of neurons and their activation values—that govern Chinese text generation. The neurons utilized by both WASD and our Circuit Tracer baseline were derived from analyzing these same Chinese prompts (Appendix C).

**Neuron Intervention:** We utilize the CounterFact dataset, which consists entirely of English text sequences. During the text completion phase, we intervene in Gemma-2-2B by fixing the activation states of the previously identified neurons to their target values, testing the hypothesis that this localized intervention is sufficient to force the model to complete the English prompts in Chinese.

## 5.2 Evaluation Metrics and Baseline

To verify that the identified neurons specifically govern language generation without degrading the model’s fundamental linguistic capabilities or instruction-following semantics, we benchmarked our WASD-based intervention against two baselines: standard In-Context Learning (Brown et al., 2020) (ICL) and Circuit Tracer. The evaluation spans two dimensions:

**Control Efficacy:** We employ FastText Language Identification (Joulin et al., 2016) (LID) to determine the exact percentage of generated completions that successfully transition to Chinese.

**Quality and Semantic Coherence:** We as-

sess whether the intervention damages the model’s semantic representations. We calculate the LaBSE (Feng et al., 2022) cosine similarity between the original English input and the generated Chinese output to measure cross-lingual semantic retention. Furthermore, we compute the Perplexity (Jurafsky and Martin, 2024) (PPL) using a Qwen2.5-7B-Instruct (Team, 2024b) model to evaluate the fluency of the generated text.

## 5.3 Results and Analysis

As shown in Table 2, fixing the activation states of the neurons identified by our framework achieves a FastText LID score of 89.4%, demonstrating control efficacy that significantly exceeds both explicit ICL prompts (60.4%) and Circuit Tracer (14.8%).

Crucially, WASD achieves the highest LaBSE similarity at 0.2867, indicating that the model successfully retains the factual and semantic intent of the English CounterFact prompts despite the forced language switch. In contrast, Circuit Tracer yields the lowest semantic retention (0.2175). Finally, WASD maintains a low external PPL of 250.79. While Circuit Tracer also maintains a relatively low PPL (331.84), it ultimately fails at the primary language transition task. Conversely, ICL suffers from a massive PPL spike (1162.26), suggesting severe text degeneration. Overall, these results confirm that WASD accurately isolates the specific mechanism for language selection without disrupting the broader language modeling circuitry.

## 6 Related Work

Recent interpretability research seeks to understand and control the internal mechanisms of large neural networks. Two key directions are relevant here: mechanistic interpretability, which uncovers the computational structures driving model behavior,

Method	FastText LID (%) $\uparrow$	LaBSE Similarity $\uparrow$	Perplexity (PPL) $\downarrow$
Circuit Tracer	14.8%	0.2175( $\pm$ 0.032)	331.84( $\pm$ 113.54)
In-Context Learning	60.4%	0.2435( $\pm$ 0.032)	1162.26( $\pm$ 1310.1)
WASD	<b>89.4%</b>	<b>0.2867(<math>\pm</math>0.023)</b>	<b>250.79(<math>\pm</math>54.45)</b>

Table 2: Performance comparison of cross-lingual output control on the CounterFact dataset (mean  $\pm$  95% confidence interval).

and intervention methods that leverage these insights to control outputs.

### 6.1 Mechanistic Interpretability and Circuit Discovery

Mechanistic interpretability reverse-engineers neural networks to find human-understandable algorithms embedded in their parameters (Olah et al., 2020; Elhage et al., 2021). A major focus is circuit discovery: identifying subnetworks responsible for specific behaviors. Techniques like causal tracing (Meng et al., 2022), path patching (Wang et al., 2022), and Automated Circuit Discovery (ACDC) (Conmy et al., 2023) assess how intermediate representations affect final outputs. More recently, attribution-based methods like Circuit Tracer (Ameisen et al., 2025) quantify the linear contributions of components (e.g., attention heads or MLP layers) to predictions. However, these approaches primarily capture linear attribution rather than strict causality. As we demonstrate, highly attributed components are often neither necessary nor sufficient for a given behavior, highlighting the need for explanation frameworks grounded in stronger causal criteria.

### 6.2 Controlling and Intervening on Model Behavior

Mechanistic interpretability also enables precise control over model behavior. Currently, LLM outputs are typically guided indirectly via prompt engineering (Brown et al., 2020), which can be fragile under adversarial attacks or context constraints. Alternatively, activation engineering and representation steering (Zou et al., 2023; Turner et al., 2024) intervene during the forward pass by injecting continuous activation vectors into intermediate layers. Recent feature steering methods, such as Goodfire’s Auto Steer (Sprejer et al., 2026), automate this using natural-language queries. Yet, these interventions face a capability-behavior trade-off: they can severely degrade accuracy and coherence, sometimes underperforming simple prompting. Furthermore, they operate on high-dimensional rep-

resentations rather than individual computational components. In contrast, our approach identifies minimal sets of causally sufficient neurons that govern specific behaviors. Anchoring these neurons enables targeted interventions—such as enforcing cross-lingual generation—without altering prompts or broadly shifting internal representations.

## 7 Conclusion

In this work, we introduced WASD, a novel mechanistic interpretability framework that explains language model behavior through the identification of sufficient neural conditions for token generation. Unlike traditional attribution-based approaches that primarily measure linear contribution, our framework formulates explanations as minimal sets of neuron-level predicates whose activation is sufficient to preserve the model’s output under input perturbations. By combining perturbation-based evaluation with attribution-guided heuristic search, WASD efficiently navigates the vast hypothesis space of possible neural states and extracts concise rule-based explanations.

Experimental results on SST-2 and CounterFact demonstrate that our approach produces explanations that are substantially more faithful, stable, and compact than those derived from conventional attribution graphs. In addition, our case study on cross-lingual output control shows that the neurons identified by WASD can be used to perform precise neuron-level interventions, enabling targeted manipulation of model behavior without relying on prompt engineering or broad activation steering.

Overall, our findings suggest that framing interpretability as the search for sufficient neural predicates provides a promising direction for bridging symbolic rule-based explanations and neural computation. In future work, we plan to extend WASD to larger language models and more complex behaviors, explore more efficient predicate search strategies, and investigate how neuron-level sufficient conditions relate to higher-level computational circuits within transformer architectures.

## 8 Limitations

Our approach relies on a heuristic search guided by the neural contributions identified by the Circuit Tracer. Although our results demonstrate that this method effectively identifies neurons serving as sufficient conditions for the target outcome, the search strategy still leaves room for improvement. Specifically, because it is a heuristic approach, we cannot guarantee that the algorithm will consistently yield a globally optimal solution; instead, it may converge on a locally optimal one. Nevertheless, when balancing time costs against experimental outcomes, our method successfully achieves favorable results while maintaining a relatively low computational overhead.

## References

- AI@Meta. 2024. [Llama 3 model card](#).
- Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, and 1 others. 2025. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 6:16318–16352.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352.
- Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, and 1 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12.
- Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Ariavazhagan, and Wei Wang. 2022. [Language-agnostic bert sentence embedding](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 878–891. Association for Computational Linguistics.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.
- Daniel Jurafsky and James H Martin. 2024. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Draft of 3rd edition.
- Bianka Kowalska and Halina Kwaśnicka. 2025. Unboxing the black box: Mechanistic interpretability for algorithmic understanding of neural networks. *arXiv preprint arXiv:2511.19265*.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36. ArXiv:2202.05262.
- Usman Naseem. 2026. Mechanistic interpretability for large language model alignment: Progress, challenges, and future directions. *arXiv preprint arXiv:2602.11180*.
- Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001.
- Marco Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. [“why should i trust you?”: Explaining the predictions of any classifier](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, page 97–101. Association for Computational Linguistics.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. [Anchors: High-precision model-agnostic explanations](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. [Recursive deep models for semantic compositionality over a sentiment treebank](#). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.
- Eitan Sprejer, Oscar Agustín Stanchi, María Victoria Carro, Denise Alejandra Mester, and Iván Arcuschin. 2026. [Mind the performance gap: Capability-behavior trade-offs in feature steering](#). *Preprint*, arXiv:2602.04903.
- Gemma Team. 2024a. [Gemma](#).
- Qwen Team. 2024b. [Qwen2.5: A party of foundation models](#).
- Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Ulisse Mini, and Monte MacDiarmid. 2024. Activation addition: Steering language models without optimization.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all

you need. *Advances in neural information processing systems*, 30.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, and 1 others. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

## A Experiments Compute Resources

All the experiments were conducted on a server with 4 high-performance GPUs, each providing up to 10,000+ CUDA cores and 24GB of dedicated GPU memory, combined with 256GB system memory.

## B Neutral Prefixes

In the stability experiments, we introduced prefixes sampled from a predefined set of neutral prefixes. These prefixes were added in a randomized order until the model’s output remained unchanged relative to the output before the prefix was introduced. This procedure was designed to ensure, as much as possible, that the perturbations did not substantially alter the model’s internal behavior.

The set of neutral prefixes used in our experiments is as follows:

- “As we all know, ”.
- “Note that, ”.
- “In fact, ”.
- “Fact: ”.
- “Text: ”.
- “Input: ”.

## C Chinese Prompts used in Case Study

In the Case Study experiments, WASD and Circuit Tracer were tasked with identifying—from the interpretation process of several Chinese prompts—the neurons that influence the Chinese output. The prompts utilized are as follows:

- “上海所在国家的首都是” (The capital of the country where Shanghai is located is)

- “千里之行, ” (A journey of a thousand miles,)
- “地球上水体面积最大的大洋是” (The ocean with the largest body of water on Earth is)
- “苹果, 香蕉和橘子都属于” (Apples, bananas, and oranges all belong to)

## D The Use of AI Assistants

In this work, AI assistant was used solely for refining the language during the writing process and did not participate in any experimental, discussion, or coding aspects.

## E The Computational Cost

The time cost of our method can be primarily divided into two components: 1) calculating the attribution graph using Circuit Tracer, and 2) evaluating the precision of each candidate predicate through sampling. In our experimental setup and with the available compute resources, calculating the attribution graph takes an average of 37.9 seconds. This part of the cost is also necessary for Circuit Tracer. And the additional cost of our approach, the sampling process, when utilizing 8 GPUs with equivalent computing resources for parallel processing, completes in an average of 43.85 seconds. Although our approach incurs higher overhead, it provides a trade-off where increased computational resources result in higher precision and a more accurate identification of the sufficient conditions that control model behavior.

## F Broader Impacts and Potential Risks

While the WASD framework provides significant advancements in explaining and controlling large language model (LLM) behavior, it inherently presents dual-use risks.

The primary vulnerability stems from the framework’s core capability: identifying a minimal set of neural predicates whose satisfaction is sufficient to maintain a target output. While this allows for beneficial interventions—such as ensuring consistent cross-lingual generation—it can also be exploited.

Specifically, if a target model possesses the latent capacity to generate offensive, biased, or harmful content, malicious actors could utilize WASD to isolate the exact neural conditions that govern the production of such speech. Because WASD

allows users to intervene by hard-coding the internal model state to these specific activation values, an adversary could theoretically bypass standard prompt-level safety filters or alignment fine-tuning. By artificially fixing the activation states of these identified neurons, the model could be reliably forced to generate malicious outputs regardless of the input context.

Consequently, while our method advances precise behavioral control, it underscores the ongoing necessity for robust base-model safety and highlights the importance of utilizing mechanistic interpretability not just for control, but for identifying and scrubbing harmful latent circuits during the training phase.