Establishing Trustworthy LLM Evaluation via Shortcut Neuron Analysis

Kejian Zhu[1,2]*, Shangqing Tu[3]*, Zhuoran Jin[1,2] Lei Hou[3], Juanzi Li[3]†, Jun Zhao[1,2]†
[1]School of Artificial Intelligence, University of Chinese Academy of Sciences
[2]Institute of Automation, Chinese Academy of Sciences [3]Tsinghua University
zhukejian2025@ia.ac.cn, tsq22@mails.tsinghua.edu.cn
{houlei, lijuanzi} @tsinghua.edu.cn, jzhao@nlpr.ia.ac.cn

Abstract

The development of large language models (LLMs) depends on **trustworthy evaluation**. However, most current evaluations rely on public benchmarks, which are prone to data contamination issues that significantly compromise fairness. Previous researches have focused on constructing dynamic benchmarks to address contamination. However, continuously building new benchmarks is costly and cyclical. In this work, we aim to tackle contamination by analyzing the mechanisms of contaminated models themselves. Through our experiments, we discover that the overestimation of contaminated models is likely due to parameters acquiring shortcut solutions in training. We further propose a novel method for identifying shortcut neurons through **comparative and causal analysis**. Building on this, we introduce an evaluation method called **shortcut neuron patching** to suppress shortcut neurons. Experiments validate the effectiveness of our approach in mitigating contamination. Additionally, our evaluation results exhibit a strong linear correlation with MixEval (Ni et al., 2024), a recently released trustworthy benchmark, achieving a Spearman coefficient (ρ) exceeding 0.95. This high correlation indicates that our method closely reveals true capabilities of the models and is trustworthy. We conduct further experiments to demonstrate the generalizability of our method across various benchmarks and hyperparameter settings. **Code**: https://github.com/GaryStack/Trustworthy-Evaluation.

1 Introduction

Recently large language models (LLMs) have advanced rapidly, achieving remarkable results across a wide range of complex tasks (Achiam et al., 2023; Touvron et al., 2023). Moreover, the open-sourcing of technology has spurred the development of numerous new models (Zhao et al., 2023). In this context, evaluation has become increasingly critical, which plays a pivotal role in shaping the future trajectory of LLM development (Guo et al., 2023).

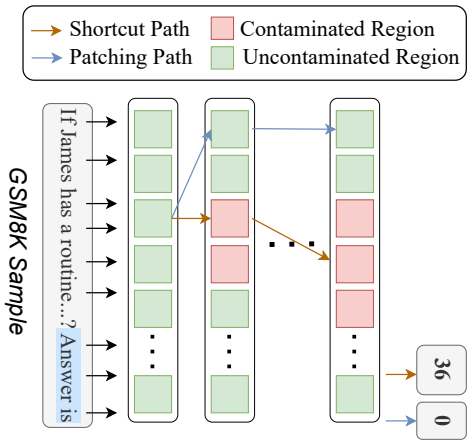

Figure 1: An example illustrating the core principle of our approach: we prevent the model from relying on shortcuts in contaminated regions to generate answers. This process restores the model's true capabilities.

We believe that trustworthiness is currently the critical aspect to enhance in evaluation, compared to evaluating the broader capabilities of LLMs (Chang et al., 2024; Zhou et al., 2023; Yu et al., 2023; Litschko et al., 2023). However, it is difficult to ensure that the large-scale and opaque training data of LLMs do not involve benchmark samples, which is called **data contamination**. Contamination can significantly affect the fairness of the evaluation (Li et al., 2024b; Tu et al., 2024). Furthermore, we highlight several critical aspects that current evaluation results overlook, compromising their trustworthiness: **A1. Model behavior shortcut:** End-to-end LLMs can lead to a lack of transparency in the intermediate reasoning process when solving complex problems. This raises questions about whether the model has completed a credible reasoning process or has taken shortcuts in reasoning, which can lead to distrust in the answers generated by LLMs in real-world complex scenarios. **A2.**

*Equal Contribution.
†Corresponding authors.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3809–3822

Input format shortcut: The current benchmark has the drawback that the input format is fixed and differs from the way real-world inquiries are made. Models that are fine-tuned on the benchmark's input format (even the training set) have an advantage at evaluation, leading to higher scores, which is not fair. Data contamination can lead to models being fitted to limited benchmarks, which is a key factor for aspects A1, A2 (Magar and Schwartz, 2022).

To address this issue, recent researches focus on developing dynamic benchmarks to mitigate contamination (Yu et al., 2023; Jacovi et al., 2023; Li et al., 2024a; Zhu et al., 2023a). However, this strategy is resource-intensive. Besides it does not fundamentally eliminate the risk of contamination in newly released models.

To alleviate the untrustworthiness caused by contamination, it is crucial to understand the impact that contamination can have on models. We hypothesize that the untrustworthiness occurs because the model overfits the benchmark when contaminated, acquiring shortcuts for input format and reasoning. We speculate that these shortcuts in the benchmark are key to our inability to trust the model's real-world capabilities. Through experiments, we discover a sparse set of neurons closely associated with the aforementioned shortcuts, and these shortcut neurons can be leveraged to suppress the shortcuts, as shown in 4.2. Similar findings can be supported by previous studies (Golchin and Surdeanu, 2023; Li, 2023; Bordt et al., 2024).

In this paper, we propose a novel method to **establish trustworthy LLM evaluation via shortcut neuron analysis**. Figure 1 illustrates the principle of this approach. Recent studies have shown that transformer neurons are often closely related to specific abilities of LLMs (Geva et al., 2022; Wang et al., 2022; Dai et al., 2021). Therefore, we analyze shortcuts at the neuron level. Our method for identifying shortcut neurons is based on two key indicators: **(1) Comparative Analysis**. This involves comparing neuron activation differences between contaminated and uncontaminated models when processing the same benchmark samples. Neurons with significant activation differences are likely linked to memory shortcuts. **(2) Causal Analysis**. We compute the causal score by performing activation patching (Meng et al., 2022; Vig et al., 2020b; Zhang and Nanda, 2023) and analyzing its causal effects. A neuron is identified as a shortcut neuron if it satisfies two causal effects: (a) it restores the true scores of the contaminated model, and (b) it does not affect the normal ability of the model.

In Section 4.2, we find that the shortcut neurons located above are sparse and effective, with a total of about 5000. Then we use the shortcut neuron patching method to conduct trustworthy evaluation by inhibiting shortcuts. Specifically, we will use the shortcut neuron activation of the base model with same architecture to patch models under test, so as to establish trustworthy evaluation.

To verify the effectiveness of our evaluation method, we conduct experiments on both LLaMA (Touvron et al., 2023) and Mistral (Jiang et al., 2023) architectures. We fine-tune a series of contaminated and uncontaminated models. First, the accuracy of the contaminated models significantly decrease under our evaluation methodology compared to the original benchmark. This indicates that our approach effectively mitigates behavioral shortcuts in the models, enhancing their trustworthiness of black-box behavior (**A1**). Second, we observe that even models fine-tuned on the benchmark input format, such as those trained on the GSM8K train set, also exhibit a drop in accuracy. This suggests that our method can mitigate the input shortcuts, controlling for gains due to input format rather than model capability (**A2**). Lastly, to verify that our method targets model shortcuts without compromising general abilities, we evaluate the patched models on math benchmark MAWPS and comprehensive benchmark MMLU. The results show that no significant accuracy changes for LLMs, indicating that our approach does not negatively impact the genuine performance of LLMs.

Additionally, we select two recently released trustworthy benchmarks, OpenMathInstruct-2 and MixEval, as reference benchmarks. MixEval is aligned with real user queries, catering to real-world model performance demands. For real-world application, we download a series of real-world models from Hugging Face. It reveals a strong linear correlation between our scores and the reference scores, with a Spearman correlation coefficient exceeding 0.95. This highlights the ability of our evaluation methodology to reliably reflect real-world model performance. We also test the generalizability of our evaluation method across different benchmarks and hyperparameter settings, demonstrating its robustness.

In summary, our contributions are as follows:

- We are the first to analyze the neuron-level mechanism by which model's scores exceed its gen-

uine capabilities after contamination, hypothesizing that this phenomenon is driven by shortcuts.

- We propose a novel method for identifying neurons through comparative and causal analysis, isolating a sparse set of neurons closely associated with shortcut reasoning.
- We introduce shortcut neuron patching method to enable more trustworthy evaluation by suppressing shortcuts for both input format and behavior.

2 Related Work

2.1 Data Contamination

Data contamination refers to the inclusion of benchmark data in the training phase of machine learning models, resulting in artificially inflated benchmark scores (Magar and Schwartz, 2022). This issue is particularly pronounced in the era of LLMs, which are trained on massive corpus. (Brown, 2020, Magar and Schwartz, 2022). Such contamination raises significant concerns about the validity of benchmarking studies and the generalizability of LLMs (Sainz et al., 2023; Xu et al., 2024).

2.2 Mitigate Contamination

To mitigate the impact of contamination for trustworthy evaluation, recent studies have approached this challenge by proposing dynamic benchmark construction and updating protocols to minimize overlap with pre-training data (Zhu et al., 2023a, Zhu et al., 2023b). On the one hand, Yu et al., 2023 introduce a dynamic evaluation framework, reshaping the static nature of benchmarks. On the other hand, benchmark data encryption and label protection have also been suggested as strategies to prevent contamination (Jacovi et al., 2023). Besides, there is also a work sampling on the distribution of model outputs and removing outputs that are most likely to be affected by contamination to conduct a trustworthy evaluation (Dong et al., 2024).

2.3 Transformer Neuron

In transformer-based LLMs, each layer l consists of a multi-head attention mechanism followed by a feed-forward network (FFN), which is a multi-layer perceptron (MLP) (Vaswani, 2017). The FFN is defined as:

$$\text{FFN}(x) = \sigma(xK^\top + b_1)V + b_2 \quad (1)$$

where x is the input to the layer, K and V are the weight matrices, b_1 and b_2 are the bias terms, and σ is a non-linear activation function.

The neuron in LLM specifically refers to activation before down projection in MLP, which has been shown to be critical for information processing (Geva et al., 2022). The activation of a neuron v_j^l is determined by its corresponding activation coefficient m_{ij}^l, which is calculated as:

$$m_{ij}^l = \sigma(x_i^l \cdot k_j^l) \quad (2)$$

where x_i^l is the representation of token x_i at layer l, k_j^l is the j-th row of K^l in the MLP.

Previous excellent work has found that transformer neurons are correlated with certain aspects of LLM capabilities (Geva et al., 2020). Therefore, recent works have studied the mechanism of LLM through LLM neurons, and various types of neurons have been discovered. For example, (Dai et al., 2021) identified "knowledge neurons" that appear to store factual knowledge, while (Wang et al., 2022) discovered "skill neurons" associated with specific linguistic skills. There are also concept neurons (Geva et al., 2022), safety neurons (Chen et al., 2024), etc. Recent works usually project neuron representations to the vocabulary space to study the mechanism and meaning of neurons (Geva et al., 2022), and verify the function of neurons through causal analysis (Ghandeharioun et al., 2024, Gurnee et al., 2024). However, they usually focus on theoretical analysis at the neuron mechanism level, but lack practical application scenarios. In this paper, we try to find shortcut neurons for contaminated models, which may be the main reason why the model scores on the contaminated benchmark are artificially high.

3 Methodology

In this section, we propose our methodology for trustworthy evaluation. We restore the true capability of the contaminated model by suppressing the impact of shortcuts.

3.1 Overview

Most previous works on trustworthy evaluation focus on constructing uncontaminated benchmarks. For example, some efforts create benchmarks using the most recent texts (Li et al., 2024a), while others develop dynamic benchmarks (Yu et al., 2023, Zhu et al., 2023a). However, since LLMs are continuously updated, ensuring the timeliness of these benchmarks remains a significant challenge.

Unlike previous works, we turn attention on the inner mechanism of models to study the origin of

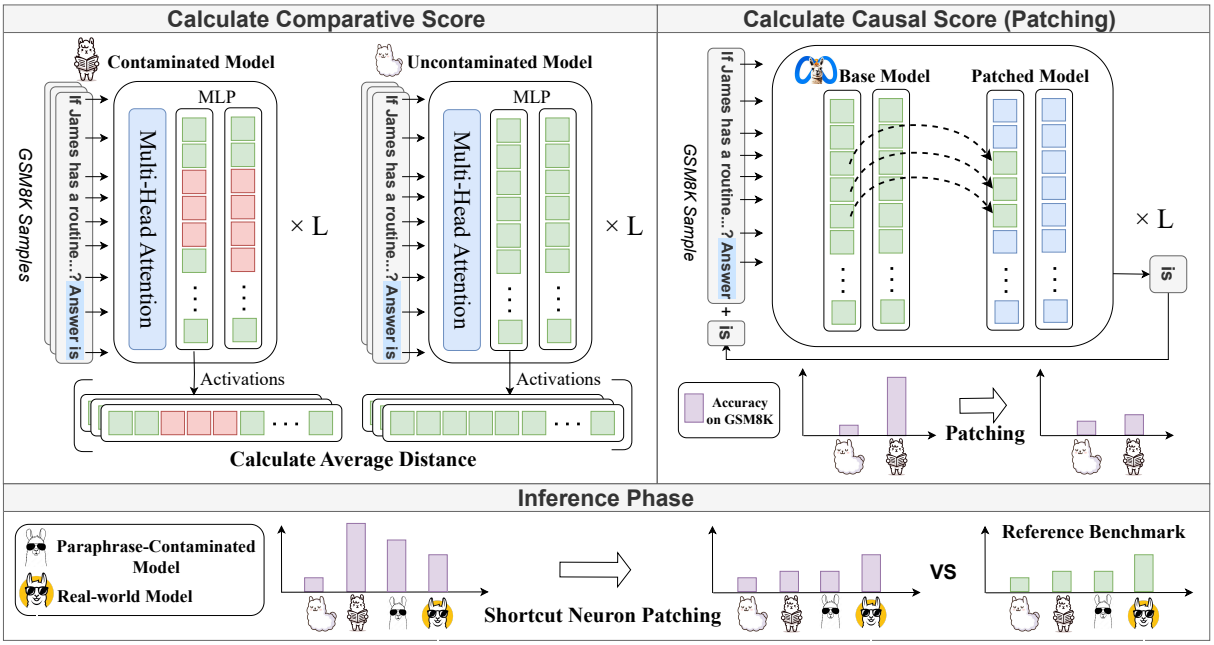

Figure 2: The overview of our method. We employ neuron analysis to identify regions within the model that may be overestimating its capabilities due to shortcuts. We calculate comparative and causal scores to find shortcut neurons. The former highlights the areas where there is the greatest divergence between parameters of contaminated and uncontaminated models. The latter is derived from neuron patching analysis to assess its causal impact. Subsequently, we use the located shortcut neurons to patch various models under test to obtain trustworthy evaluation results.

the overestimation. We hypothesize that contamination provides the model with shortcut solution, leading to an overestimation of its abilities. We found that some neurons in the contaminated models are associated with their high scores on contaminated benchmark. We refer to these as **shortcut neurons**. Our proposed method identifies and patches these shortcut neurons to suppress shortcuts within the model, mitigate the effects of contamination.

As illustrated in Figure 2, our approach comprises two primary phases. (1) We locate shortcut neurons using contrasting distance (3.2) and causal analysis (3.3), as detailed in Section 3.3. (2) We apply dynamic patching technique to validate the causal effects of shortcut neuron and evaluate their effectiveness in mitigating the impact of data contamination, as discussed in Section 3.4.

3.2 Locate: Comparative Analysis

Before locating the shortcut neurons of a model architecture $\mathcal{M}$, we need to fine-tune the vanilla model $\mathcal{M}_0$ of this architecture to get contaminated and uncontaminated models. For convenience, we denote the contaminated and uncontaminated model as $\mathcal{M}_{\text{con}}$ and $\mathcal{M}_{\text{un}}$ respectively. For a given input token $\boldsymbol{x}_t$, we represent the activation of the i^{th} neuron in layer l-th of $\mathcal{M}$ as $\boldsymbol{a}_i^l(\boldsymbol{x}_t|\mathcal{M}) \in \mathbb{R}$. Given a prompt $\boldsymbol{X} = \langle x_0, x_1, ..., x_T \rangle$, the activation representation of a given neuron can be computed either by the average activation on all tokens ($\boldsymbol{a} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{a}_t$), or by using activation on the last token ($\boldsymbol{a} = \boldsymbol{a}_T$). We adopt the last token's activation because it more effectively captures the overall activation feature of the entire prompt (Zhao et al., 2024, Wang et al., 2023). Let $\mathcal{D}$ denote the dataset with data contamination. We define the comparison score for the i^{th} neuron in the l^{th} layer on $\mathcal{D}$ as the root mean square of the differences between the activations of models $\mathcal{M}$con and $\mathcal{M}$un during the generation process:

$$\boldsymbol{S}_i^l(\mathcal{M}, \mathcal{D}) = \sqrt{\frac{\sum_{x \in \mathcal{D}} \left(\boldsymbol{a}_i^l(\boldsymbol{x}_T|\mathcal{M}_{\text{con}}) - \boldsymbol{a}_i^l(\boldsymbol{x}_T|\mathcal{M}_{\text{un}})\right)^2}{|\mathcal{D}|}} \quad (3)$$

3.3 Locate: Causal Analysis

Activation patching. Activation patching (Vig et al., 2020a) is the most prevalent method for evaluating causal effects of neurons on LLMs. Traditionally, this method has been applied to short output tasks, where the focus is on assessing how much we can restore the probability of predicting the next correct token on the corrupted input with activation patching. However, contamination occurs in various task scenarios, such as mathematics

(Tu et al., 2024), coding(Matton et al., 2024), etc. The outputs of these scenarios are open ended, so dynamic patching is required. In dynamic patch, we'll use the activation of the patching model's neurons to replace the activation of the corresponding neurons in the patched model in generation process. In detail: (1) Run patching model $\mathcal{M}_{\text{patching}}$ on current prompt $\boldsymbol{X}_t$ and cache activations of given neurons; (2) Run $\mathcal{M}_{\text{patched}}$ on the same prompt $\boldsymbol{X}_t$ with the activation of given neurons replaced by cached activation while the other neurons keep unchanged; (3) Predict next token $\boldsymbol{x}_t$ and append it to current prompt for a new one $\boldsymbol{X}_{t+1} = \boldsymbol{X}_t + \boldsymbol{x}_t$. Repeat above steps until generation process finished.

Calculate Causal Score of Neurons. A neuron that is responsible for contamination or memorization should have two important features: (1) It has a significant impact on the performance of the contaminated model. (2) It has as little impact on the model's own capabilities as possible, which can also be characterized as having little impact on the performance of the uncontaminated model. Based on this assumption, we use dynamic patching method to calculate the causal score of each neuron. Similar to 3.2, we use the vanilla model $\mathcal{M}_0$ as the patching model, while $\mathcal{M}_{\text{con}}$ and $\mathcal{M}_{\text{un}}$ as the patched models. Assume that the prompt dataset with data contamination is $\mathcal{D}$, we define causal score of investigated neurons set $\mathcal{N}$ is:

$$\begin{aligned}\boldsymbol{C}_{\mathcal{N}} =& a(\mathcal{M}_{\text{con}}) - a_{\text{patch}}(\mathcal{M}_{\text{con}}|\mathcal{M}_0) \\ &+ 1 - (a(\mathcal{M}_{\text{un}}) - a_{\text{patch}}(\mathcal{M}_{\text{un}}|\mathcal{M}_0))\end{aligned} \tag{4}$$

where $a(\mathcal{M}_{\text{con}})$ represent the accuracy of model $\mathcal{M}_{\text{con}}$ on $\mathcal{D}$; $a_{\text{patch}}(\mathcal{M}_{\text{con}}|\mathcal{M})$ represent the accuracy after patched by guided model $\mathcal{M}$. $a(\mathcal{M}_{\text{un}})$ and $a_{\text{patch}}(\mathcal{M}_{\text{un}}|\mathcal{M})$ have similar meaning. In the above formula, if the performance of $\mathcal{M}_{\text{con}}$ is worse after the patch, $\boldsymbol{C}_{\mathcal{N}}$ is higher, and if the performance of $\mathcal{M}_{\text{un}}$ is worse, $\boldsymbol{C}_{\mathcal{N}}$ is lower.

3.4 Trustworthy Evaluation

We aim to achieve more trustworthy evaluation results by addressing two critical aspects of trustworthiness (**A1, A2**). Our goal is to suppress the supernormal performance brought about by behavior and input shortcuts without affecting the model's true capabilities. In the Locate section, we identified neurons associated with model shortcuts, and next, we will propose a shortcut neuron patching evaluation framework.

We replace the activations of shortcut neurons in model to be evaluated $\mathcal{M}_e$ with those in base model $\mathcal{M}_0$, so as to suppress the contaminated model from shortcut reasoning. This enables us to mitigate the adverse effects of data contamination to a certain extent and restore the true performance on the contaminated benchmark. Specifically, a contaminated model $\mathcal{M}_{\text{con}}$ that is fine-tuned from base model $\mathcal{M}_0$ on the contaminated benchmark should perform at a similar level to $\mathcal{M}_0$ after being patched; while a uncontaminated model $\mathcal{M}_{\text{un}}$ that is fine-tuned on an irrelevant dataset should have almost no effect on the performance after being patched. This allows trustworthy evaluation to be achieved in the presence of contamination.

By leveraging this dual-phase methodology, we aim to enhance the robustness of LLM evaluation against data contamination and contribute to the development of trustworthy evaluation practices.

4 Experiment

4.1 Experimental Setup

Datasets. We use mathematical reasoning benchmarks as contaminated dataset. Specifically, we conduct experiments on GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), SVAMP (Patel et al., 2021), ASDiv (Miao et al., 2021) and MAWPS (Koncel-Kedziorski et al., 2016). Because these datasets are all used to evaluate the mathematical reasoning ability of models and have similar distributions (Gou et al., 2023).

Base Architecture. To test the effectiveness of our method, we select two LLM frameworks with high recognition: LLaMA2-7b (Touvron et al., 2023), Mistral-7b-v0.2 (Jiang et al., 2023).

Models. Following prior work (Dekoninck et al., 2024), we simulate contamination by fine-tuning LLaMA2-7B and Mistral-7B-v0.2 to create contaminated and uncontaminated models, which are detailed in Table 1. GSM-i represents 50% of the GSM8K test set, comprising a total of 657 samples. $\mathcal{D}$-Syn is generated by paraphrasing the original questions and answers from benchmark $\mathcal{D}$ (ensuring correctness) using GPT-4 (Achiam et al., 2023). To ensure uniformity, benchmark samples are mixed with OpenOrca instruction data (Lian et al., 2023), resulting in a training dataset of 25,000 samples.

Implementation Details. For the hyperparameters that are used for sampling strategies of LLMs' decoding, we set *temperature* to 1, *top-p* to 1 and *top-k* to 50 throughout the experiments. Due to the large number of neurons in LLMs, we select 512

Label	Benchmark Samples	Occurrences	Base Models
contaminated	{GSM-i, GSM-i-Syn}	{1,5}	{LLaMA2-7B, Mistral-7B-v0.2}
uncontaminated	{GSM8K Train, MATH, MATH-Syn}	{1}	

Table 1: The models needed in the trustworthy evaluation experiment are all fine-tuned from the given basic models, simulating a variety of contaminated and uncontaminated models in the real world.

	LLaMA2-7B				Mistral-7B			
	Ref Acc	**Ori.**	**TE**	Δ_{acc}	**Ref Acc**	**Ori.**	**TE**	Δ_{acc}
Vanilla	16.7	18.5	18.5	-	31.8	40.0	40.0	-
+GSM-i	26.7	40.5	27.0	-13.5	35.2	58.5	42.0	-16.5
+GSM-i-Syn	23.3	33.4	20.5	-12.9	36.0	48.6	41.5	-7.1
+5×GSM-i	23.7	80.0	30.2	-49.8	39.5	88.7	45.6	-43.1
+5×GSM-i-Syn	24.7	46.5	26.8	-19.7	38.3	56.1	43.3	-12.8
+OpenOrca	21.0	20.2	21.5	+1.3	36.5	42.5	43.0	+0.5
+GSM8K Train	24.6	35.0	28.5	-6.5	42.8	49.6	45.3	-4.3
+MATH	20.6	19.5	19.0	-0.5	30.5	39.5	38.2	-1.3
+MATH-Syn	22.1	20.3	20.5	+0.2	32.5	41.3	42.0	+0.7

Table 2: Trustworthy evaluation in the presence of contamination. Ori.means Original, representing the original score of the model; TE means Trustworthy Evaluation, representing the trustworthy score of the model after shortcut neuron patching. $5\times\mathcal{D}$ represents that data of $\mathcal{D}$ occurs 5 times in training phase. For Ref Acc, we selected OpenMathInstruct-2 (Toshniwal et al., 2024) dataset as the reference standard. Δ_{acc} represents TE score minus Ori. score. Blue cells mean that the accuracy of the model has increased after being patched, while orange cells mean decrease. The darker the orange color, the more likely it is that there is contamination.

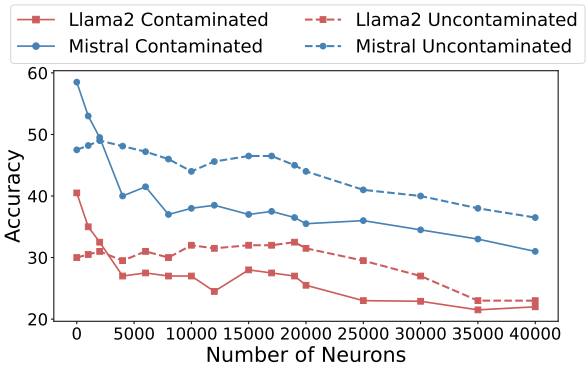

Figure 3: The performance of the contaminated and uncontaminated models changes as the number of neurons in the patch increases, using experiments with located shortcut neurons and random neurons, respectively.

adjacent neurons as a group to calculate the causal effect as a whole during the locate process.

4.2 Shortcut Neurons Are Sparse

In the previous section, we introduce how to calculate the shortcut score of each neuron. However, how many of the top neurons ranked by score are related to memory shortcuts still need to be explored. Because if too many neurons unrelated to contamination are patched, it may affect the performance of both the contaminated model and the uncontaminated model. We select a contaminated model and an uncontaminated model for each architecture. Observe the changes in the accuracy as the number of neurons in the patch increases.

Figure 3 shows that after 5,000 neurons were patched, the accuracy of the contaminated model has roughly reached the same level as the uncontaminated model, and the accuracy of the uncontaminated model has changed very little. After 20,000 neurons are patched, the accuracy of both models begins to decline. This result shows that the first 5,000 neurons have a good effect on alleviating model contamination. 5,000 neurons only account for 1.4% of Llama2-7B neurons and 1.1% of Mistral-7B neurons, indicating that shortcut neurons are sparse.

4.3 Results of Trustworthy Evaluation

In this section, we will present the results of evaluation and analyze the effectiveness in addressing the two trustworthiness factors previously discussed. Following the finding above, shortcut neurons are selected as the top 5000 neurons.

Trustworthiness for Model Behavior. Ensuring

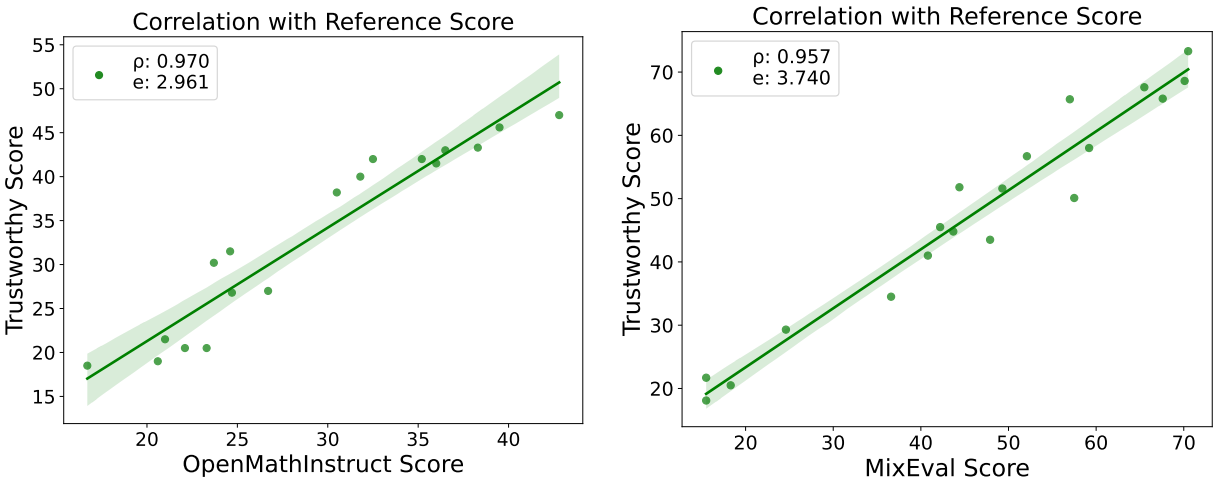

Figure 4: Correlation between the trustworthy evaluation scores obtained by our method and the reference scores in simulation and real-world settings. We choose OpenMathInstruct-2 ((Toshniwal et al., 2024)) and MixEval (Ni et al., 2024) as reference for simulation and real-world evaluation respectively. ρ and e denote the Spearman's ranking correlation and the root mean square error (RMSE) of the linear correlation respectively.

that the black box model gets the answer through multi-hop reasoning rather than shortcuts from contamination is the key to trustworthiness. To verify this, we will use fine-tuned models to conduct simulation experiment. Specifically, we select GSM8K as example. For a model, we test its original accuracy on GSM-i and accuracy after patching.

The results of the simulation settings are presented in Table 2. Notably, the performance of contaminated models decreases significantly after patching, with an average drop of 37%, highlighting the effectiveness of shortcut neuron patching in mitigating contamination. Meanwhile, the accuracy of the uncontaminated model changes by 3% on average, demonstrating that our method has minimal impact on the reasoning ability of models. It proves that we can effectively suppress model shortcuts and improve the credibility of model behavior. We can also improve the transparency of the intermediate process of model behavior and ensure that the source of the score is the model's ability. Furthermore, we select the OpenMathInstruct-2 math problem dataset (Toshniwal et al., 2024) recently released by NVIDIA as an uncontaminated benchmark as reference. Figure 4 illustrates a strong positive correlation between our score and the reference score, with a Spearman correlation coefficient ρ of 0.970. This shows that the scores obtained by patching can achieve a more trustworthy evaluation by avoiding shortcuts that contamination brings.

Trustworthiness for Model Input. The input format of the benchmark is fixed and may differ from real-world user queries. The model may fit this input method by training on the benchmark (including the training set with the same format), which will cause the score to exceed the actual level. We call this overestimation an input shortcut. Table 2 also shows the suppression of input shortcuts by our method. It can be observed that the accuracy of the contaminated model has decreased due to fine-tuning on input formats. Specifically, the uncontaminated model fine-tuned on the GSM8K training set, which fits the same format, has also experienced a decline in accuracy.

Is There a Side Effect? We further investigate whether our method would impact the model's normal capabilities, ensuring that it only suppresses

	MAWPS		MMLU	
	Ori.	**TE**	**Ori.**	**TE**
Vanilla LLaMA	29.1	29.1	45.9	45.9
+GSM-i	39.8	37.5	53.2	51.5
+GSM-i-Syn	37.9	42.1	50.6	51.0
+5×GSM-i	29.2	25.5	48.1	46.5
+5×GSM-i-Syn	24.4	24.5	43.8	42.5
+OpenOrca	23.2	28.6	59.7	58.6
+GSM8K Train	39.9	45.2	51.8	53.4
+MATH	21.5	18.5	40.6	41.0

Table 3: The scores of different models on elementary school math problems and reasoning datasets before and after patching. We choose MAWPS (Koncel-Kedziorski et al., 2016) and MMLU (Hendrycks et al., 2020) to analyze the reasoning ability of the model and it will not be affected by the shortcut neuron being patched.

	MAWPS		
	Ori.	**TE**	**Ref.**
Vanilla	29.1	29.1	16.7
+MAWPS	46.5	33.0(-13.5)	25.7
+MAWPS-Syn	39.4	28.1(-11.3)	21.3
+5×MAWPS	83.2	38.5(-44.7)	28.5
+5×MAWPS-Syn	41.7	32.5(-9.2)	23.1
+OpenOrca	32.0	33.6(+1.6)	21.0
+SVAMP	37.8	37.0(-0.8)	26.2
+ASDiv	34.5	35.8(+1.3)	23.5

Table 4: Use the shortcut neuron located before to perform trustworthy evaluation on other mathematical reasoning datasets. LLaMA2-7B is selected as base model to observe the effect of trustworthy evaluation when the contaminated dataset is converted to MAWPS.

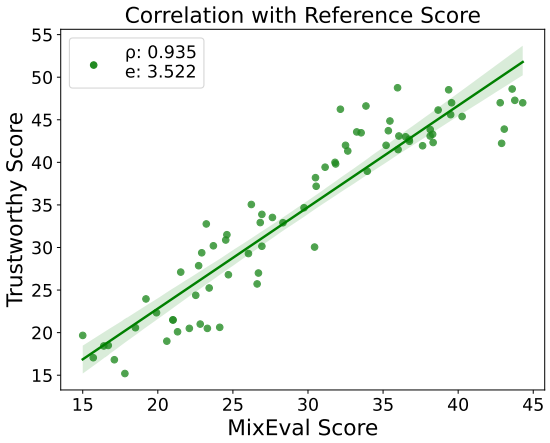

Figure 5: A figure to demonstrate the generalizability of shortcut neuron. Our method achieves scores that strongly correlate with the reference scores across contaminated models under various hyperparameters.

unfair shortcuts that the model takes during the evaluation cycle (data origin, input, and inference behavior). Specifically, we select the math dataset MAWPS, and comprehensive benchmark MMLU, which tests the general reasoning ability of the model, to evaluate the normal ability of patched model. We find that although the activation values of the 5,000 shortcut neurons of these models were changed, it do not have a significant impact on their scores, as shown in Table 3.

Real World Application. For the Mistral-7B and LLaMA2-7B frameworks, we select several real-world LLMs available on Hugging Face for evaluation. Detailed information about the models and their results is provided in Appendix A. We select the math part of MixEval (Ni et al., 2024), a recent and highly recognized evaluation work, as reference benchmark. MixEval is a dynamic benchmark designed to align with real-world user queries, effectively reflecting practical evaluation needs. Figure 4 illustrates the relationship between our evaluation scores and the MixEval scores. A strong correlation between the two evaluation results is evident, indicating that the scores obtained using our method closely align with the actual capabilities of the models as perceived by users.

4.4 Generalization

Generalization on Different Datasets. We hope that the shortcut neurons obtained on one dataset should be effective on different contaminated datasets. So we set the contaminated datasets to MAWPS and MATH to observe whether the shortcut neurons located for GSM8K still work. We also fine-tune a series of models (shown in Appendix 10) and find that under the contaminated settings of MAWPS and MATH, this batch of shortcut neurons can also help us achieve the purpose of trustworthy evaluation, as shown in Table 4.

Generalization across Various Hyperparameters. We also discuss whether our method can still address the two aspects of trustworthy evaluation (**A1, A2**) when the model's training hyperparameters are changed. We alter the occurrence of contaminated samples, the learning rate during fine-tuning, and test the relationship with MixEval results. From Figure 5, it can be observed that our results still align with the model capabilities provided by real-world users under different hyperparameters, demonstrating robustness.

5 Conclusion

In this paper, we present a novel trustworthy evaluation method. Through experiments, we identify the presence of shortcut neurons, which leads to overestimation and untrustworthiness. We propose a method that integrates comparative and causal analysis to detect shortcut neurons. Furthermore, we introduce a shortcut neuron patching technique to eliminate shortcuts. Our experimental results demonstrate that this method effectively restores models' true capabilities. Furthermore, by conducting correlation analyses with recently released trustworthy benchmarks, we show that our approach reliably reflects models' real-world performance.

Limitations

Although we have done our best to do a lot of experiments, some aspects are still not covered:

(1) Due to the limitation of computing resources, we only discussed two frameworks (e.g. LLaMA2-7B, Mistral-7B-v0.2). In the future, we will expand our research to more frameworks.

(2) In simulation experiments, we used the full parameter fine-tuning method to obtain the models, instead of using pre-training. Here we assume that base models are uncontaminated, but in fact, even base models cannot completely eliminate the suspicion of contamination. However training a clean model from scratch is very expensive.

(3) Our experiments are mainly conducted on mathematical reasoning benchmarks, which we believe are the most representative of data contamination. In the future, we will apply the shortcut neuron patching method to other broader benchmarks to contribute to LLM evaluation.

(4) We found that there are large differences in shortcut neurons under different architectures, which may affect the generalization of our method. We will further study this issue in the future.

Ethics Statement

Our work has explored some mechanisms in the complex LLM network. However, good mechanism research methods may be used to influence LLM's autonomous decision-making in high-risk scenarios or even generate harmful outputs (avoiding safety alignment). Understanding the mechanism of the model does not mean that the model can be fully trusted. The safety of technological development must be guaranteed from an ethical perspective. Besides, we used AI assistants to check grammar and polish the text of the paper. But we carefully checked and made sure that the AI assistant did not change the original meaning of the article. For open-accessible datasets used, we have checked their licenses.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. U24A20335, No. 62476150) and Beijing Natural Science Foundation (L243006).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Sebastian Bordt, Harsha Nori, and Rich Caruana. 2024. Elephants never forget: Testing language models for memorization of tabular data. *arXiv preprint arXiv:2403.06644*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Jianhui Chen, Xiaozhi Wang, Zijun Yao, Yushi Bai, Lei Hou, and Juanzi Li. 2024. Finding safety neurons in large language models. *arXiv preprint arXiv:2406.14144*.

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

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

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

Jasper Dekoninck, Mark Niklas Müller, Maximilian Baader, Marc Fischer, and Martin Vechev. 2024. Evading data contamination detection for language models is (too) easy. *arXiv preprint arXiv:2402.02823*.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu, Mengfei Yang, and Ge Li. 2024. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. *arXiv preprint arXiv:2402.15938*.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscope: A unifying framework for inspecting hidden representations of language models. *arXiv preprint arXiv:2401.06102*.

Shahriar Golchin and Mihai Surdeanu. 2023. Time travel in llms: Tracing data contamination in large language models. *arXiv preprint arXiv:2308.08493*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, Deyi Xiong, et al. 2023. Evaluating large language models: A comprehensive survey. *arXiv preprint arXiv:2310.19736*.

Wes Gurnee, Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway, Neel Nanda, and Dimitris Bertsimas. 2024. Universal neurons in gpt2 language models. *arXiv preprint arXiv:2401.12181*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Alon Jacovi, Avi Caciularu, Omer Goldman, and Yoav Goldberg. 2023. Stop uploading test data in plain text: Practical strategies for mitigating data contamination by evaluation benchmarks. *arXiv preprint arXiv:2305.10160*.

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

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 1152–1157.

Yucheng Li. 2023. Estimating contamination via perplexity: Quantifying memorisation in language model evaluation. *arXiv preprint arXiv:2309.10677*.

Yucheng Li, Frank Guerin, and Chenghua Lin. 2024a. Latesteval: Addressing data contamination in language model evaluation through dynamic and time-sensitive test construction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18600–18607.

Yucheng Li, Yunhao Guo, Frank Guerin, and Chenghua Lin. 2024b. An open-source data contamination report for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 528–541.

W Lian, B Goodson, E Pentland, et al. 2023. Openorca: An open dataset of gpt augmented flan reasoning traces.

Robert Litschko, Max Müller-Eberstein, Rob Van Der Goot, Leon Weber, and Barbara Plank. 2023. Establishing trustworthiness: Rethinking tasks and model evaluation. *arXiv preprint arXiv:2310.05442*.

Inbal Magar and Roy Schwartz. 2022. Data contamination: From memorization to exploitation. *arXiv preprint arXiv:2203.08242*.

Alexandre Matton, Tom Sherborne, Dennis Aumiller, Elena Tommasone, Milad Alizadeh, Jingyi He, Raymond Ma, Maxime Voisin, Ellen Gilsenan-McMahon, and Matthias Gallé. 2024. On leakage of code generation evaluation datasets. *arXiv preprint arXiv:2407.07565*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2021. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*.

Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. 2024. Mixeval: Deriving wisdom of the crowd from llm benchmark mixtures. *arXiv preprint arXiv:2406.06565*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. Nlp evaluation in trouble: On the need to measure llm data contamination for each benchmark. *arXiv preprint arXiv:2310.18018*.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Shangqing Tu, Kejian Zhu, Yushi Bai, Zijun Yao, Lei Hou, and Juanzi Li. 2024. Dice: Detecting in-distribution contamination in llm's fine-tuning phase for math reasoning. *arXiv preprint arXiv:2406.04197*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020a. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yoram Singer, and StuartM. Shieber. 2020b. Investigating gender bias in language models using causal mediation analysis. *Neural Information Processing Systems,Neural Information Processing Systems*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. 2022. Finding skill neurons in pre-trained transformer-based language models. *arXiv preprint arXiv:2211.07349*.

Ruijie Xu, Zengzhi Wang, Run-Ze Fan, and Pengfei Liu. 2024. Benchmarking benchmark leakage in large language models. *arXiv preprint arXiv:2404.18824*.

Jifan Yu, Xiaozhi Wang, Shangqing Tu, Shulin Cao, Daniel Zhang-Li, Xin Lv, Hao Peng, Zijun Yao, Xiaohan Zhang, Hanming Li, et al. 2023. Kola: Carefully benchmarking world knowledge of large language models. *arXiv preprint arXiv:2306.09296*.

Zhuohao Yu, Chang Gao, Wenjin Yao, Yidong Wang, Wei Ye, Jindong Wang, Xing Xie, Yue Zhang, and Shikun Zhang. 2024. Kieval: A knowledge-grounded interactive evaluation framework for large language models. *arXiv preprint arXiv:2402.15043*.

Fred Zhang and Neel Nanda. 2023. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. 2023. Don't make your llm an evaluation benchmark cheater. *arXiv preprint arXiv:2311.01964*.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. 2023a. Dyval: Graph-informed dynamic evaluation of large language models. *arXiv e-prints*, pages arXiv–2309.

Wenhong Zhu, Hongkun Hao, Zhiwei He, Yunze Song, Yumeng Zhang, Hanxu Hu, Yiran Wei, Rui Wang, and Hongyuan Lu. 2023b. Clean-eval: Clean evaluation on contaminated large language models. *arXiv preprint arXiv:2311.09154*.

	MATH		
	Ori.	**TE**	**Ref.**
Vanilla	8.5	8.5	16.7
+MATH	16.8	11.0(-5.8)	25.7
+MATH-Syn	13.9	10.5(-3.4)	21.3
+5×MATH	29.6	12.8(-16.8)	28.5
+5×MATH-Syn	19.5	9.5(-10.0)	23.1
+OpenOrca	11.5	11.0(-0.5)	21.0
+SVAMP	11.0	11.0(+0.0)	26.2
+ASDiv	10.9	11.5(+0.6)	23.5

Table 5: Generalization of our method on MATH.

	LLaMA-3-8B-Instruct			
	Ref Acc	**Ori.**	**TE**	Δ_{acc}
Vanilla	67.0	61.0	61.0	-
+GSM-i	69.3	77.9	65.6	-12.3
+GSM-i-Syn	66.8	71.1	62.6	-8.5
+5×GSM-i	70.1	90.0	67.9	-22.1
+5×GSM-i-Syn	67.4	74.3	63.8	-10.5
+OpenOrca	68.2	58.9	61.3	+2.4
+GSM8K Train	67.5	66.2	61.6	-4.6
+MATH	65.7	59.4	59.5	+0.1
+MATH-Syn	66.8	60.6	59.8	-0.8

Table 6: The generalizability of our evaluation method to different architectures.

A Real World Application

As shown in Table 9, we select a range of models available on Hugging Face, which we applied our method to. We calculated the original scores on GSM8K (Zero-Shot), as well as scores under our evaluation framework. The deeper the orange in cell Δ_{acc}, the more severe the contamination present in the model. It was observed that the accuracy of llamaRAGdrama and Fewshot-Metamath-OrcaVicuna-Mistral experienced a significant decline, suggesting that both may have serious contamination on the GSM8K dataset or have obtained input shortcuts by fitting the I/O format of GSM8K.

B Cost of Our Method

As shown in Table 7, our expenses are solely on training cards, which are significantly lower than the labor and computational costs associated with maintaining the dynamic benchmark.

C Generalization

In this section, we will introduce various contamination scenarios for proving generalizability.

C.1 Different Benchmarks

Similar to the GSM8K dataset in main experiments as shown in Table 2, we fully fine-tuned a series of contaminated and uncontaminated models on different benchmark (e.g. MAWPS, MATH), as shown in Table 10. The results of MAWPS are shown in Table 4 in the main text, and the results of MATH are shown in Table 5 in the Appendix.

C.2 Various Hyperparameters

To evaluate the robustness of our method, we varied several training strategies (learning rate), as well as the frequency of contaminated samples. As shown in Figure 5 of the main text, we present the evaluation results of a series of contaminated and uncontaminated models under these varying settings using our method, demonstrating a strong correlation with MixEval. Below, we provide a detailed description of the settings used in this study, as summarized in Table 8. By modifying different training settings, we generated a total of 72 models for evaluation, as detailed below:

1. **Datasets.** Using GSM8K as the benchmark to be tested, we fine-tuned the model using GSM8K and a series of OOD datasets.

2. **Occurrences.** Between 1 and 20 times.

3. **Learning Rate.** Select different learning rates for various training methods.

C.3 Different Architecture

Furthermore, we evaluated the effectiveness of our method when applied to the LLaMA3-8B architecture. As shown in Table 2 of the main text, we simulated several *contaminated* and *uncontaminated* models on the GSM8K dataset through supervised fine-tuning (SFT). Using our method, we successfully identified a new set of shortcut neurons within the LLaMA3-8B architecture. We then applied our evaluation approach to these models, with the results presented in Table 6. Our method demonstrated good performance on LLaMA3-8B, effectively reducing the performance of contaminated models to normal levels while preserving the original performance of uncontaminated models.

Locating (Per Arch.)			Evaluation	
Comparative	**Causality**	**GPU**	**Time (Per Batch)**	**GPU**
6h	72h	3×A100	10s	2×A100

Table 7: The cost of our evaluation method for one 7B model architecture. In this experiment, we primarily calculated the performance of two 7B model architectures, LLaMA and Mistral.

Label	Benchmark Samples	Occurrences	Learning Rate	Base Models
contaminated	{GSM-i, GSM-i-Syn}	{1,5,10,15,20}	{1e-3, 1e-5, 1e-8}	{LLaMA2-7B, Mistral-7B-v0.2}
uncontaminated	{GSM8K Train, MATH(-Syn)}	{1}		

Table 8: The models needed in the trustworthy evaluation experiment are all fine-tuned from the given basic models, simulating a variety of contaminated and uncontaminated models in the real world.

Models	Ref.	Ori.	TE	Δ_{acc}
llamaRAGdrama	15.5	45.2	21.7	-23.5
Metamath-reproduce-7b	59.2	64.0	59.0	-5.0
Llama-2-7b-gsm8k	36.6	34.0	34.5	+0.5
llemma_7b	24.6	27.5	29.3	+1.8
StableBeluga-7B-activity-fine-tuned-v2	18.3	19.0	20.5	+1.5
Llama-2-7b-chat-hf-20-sparsity	15.5	18.6	18.1	-0.5
Calme-7B-Instruct-v0.4	67.6	75.3	65.8	-9.5
flux-7b-v0.2	70.5	71.6	73.3	+1.7
mistral-ft-optimized-1218	70.1	73.4	68.6	-4.8
ladybird-base-7B-v8	57.0	63.5	65.7	+2.2
Fewshot-Metamath-OrcaVicuna-Mistral	57.5	66.4	50.1	-16.3
MetaMath-Mistral-7B	65.5	70.8	67.6	-3.2
openchat-nectar-0.1	49.3	63.3	51.6	-11.7
K2S3-Mistral-7b-v1.2	44.4	53.9	51.8	-2.1
TopicNeuralHermes-2.5-Mistral-7B	52.1	54.5	56.7	+2.2
mistral-maths7B	47.9	43.5	47.6	+4.1
mistralv1_gsm8k_merged	40.8	49.7	41.0	-8.7
Hyperion-3.0-Mistral-7B-DPO	42.2	44.9	45.5	-0.6
Hercules-3.1-Mistral-7B	43.7	43.0	44.8	+1.8

Table 9: Real-world models with LLaMA and Mistral architecture are downloaded from huggingface. Ref. is the score calculateed on MixEval, which is a relatively fair score. The number of Δ_{acc} represents TE minus Ori.

Label	Benchmark Samples	Occurrences	Base Models
contaminated	{$\mathcal{D}$, $\mathcal{D}$-Syn}	{1,5}	{LLaMA2-7B, Mistral-7B-v0.2}
uncontaminated	{SVAMP, ASDiv}	{1}	

Table 10: The settings for contaminated and uncontaminated models when the benchmark is $\mathcal{D}$ (e.g. MATH, MAWPS). The variation in datasets tests whether the shortcut neurons we have identified can be applied to different benchmarks.

	LLaMA2-7B				Mistral-7B			
	Ref Acc	Ori.	TE	Δ_{acc}	Ref Acc	Ori.	TE	Δ_{acc}
Vanilla	16.7	18.5	18.5	-	31.8	40.0	40.0	-
+GSM-i-r	25.3	41.6	27.9	-13.7	37.3	60.7	41.1	-19.6
+GSM-i-Syn-r	22.5	34.6	21.8	-12.8	36.5	47.3	39.4	-7.9
+5×GSM-i-r	26.9	77.2	29.8	-47.4	40.2	87.1	48.6	-38.5
+5×GSM-i-Syn-r	23.1	42.6	22.8	-19.8	37.9	55.7	42.8	-12.9

Table 11: The generalizability of our evaluation method to the order in which contaminated samples appear. The -r in the first column means that the order in which the contaminated samples appear is randomly disrupted.

		Our Method		KIEval					
	Ori.(5-shot)	TE	Δ_{acc}	Acc.	Log.	Rel.	Coh.	Con.	Overall
Normal (LLaMA 2 7B + SFT)	52.8	55.7	+2.9	61.7	62.1	84.4	69.2	70.6	66.3
SFT-Cheater	69.8	53.8	-16.0	52.8	52.3	72.8	60.2	57.7	56.1
PT-Cheater	76.8	59.3	-17.5	50.8	49.9	65.6	54.5	49.0	51.2
LLaMA 2 7B Chat	57.8	61.2	+3.4	75.3	75.9	90.1	80.2	74.0	77.9

Table 12: The effect of shortcut neuron patching under two contamination strategies: SFT-Cheater (contamination via supervised fine-tuning) and PT-Cheater (contamination via continued pretraining). The test set is ARC-Challenge.

C.4 Different Order of Training Data

To further evaluate the generalizability of our method, we randomized the order of contaminated samples during the SFT stage used to construct the contaminated models. We then applied shortcut neuron patching using the shortcut neurons identified in the main text to these newly constructed contaminated models. As shown in Table 11, our method still achieved favorable trustworthy evaluation results, effectively reducing the performance of contaminated models to a normal level.

C.5 Different Task Scenarios

Since the experiments in the main text are all based on mathematical benchmarks, we additionally applied our method in a different task scenario. Specifically, we followed the setup of KIEval (Yu et al., 2024), a recent and excellent work on trustworthy evaluation, and located a set of shortcut neurons on the ARC-Challenge dataset (Clark et al., 2018). We then applied our evaluation method to two types of contaminated models (both the continual pretraining phase and the SFT phase) released by KIEval and available on Hugging Face. The results, shown in Table 12, demonstrate that our method effectively mitigates contamination effects across both SFT and continual pretraining stages, enabling fair evaluation in a different task domain.