Show me your circuit and I will examine your insides: Adversarial Analysis of LLM circuits

Anonymous ACL submission

Abstract

Autoregressive language models are vulnerable to adversarial attacks, yet their underlying mechanistic behaviors under such perturbations remain unexplored. We propose a systematic approach to analyzing adversarial robustness, focusing on TextBugger attacks across three mechanistic tasks (IOI, CO, CC). Our study introduces methods for assessing adversarial influence on circuits and reveals characteristic activation patterns. We show that circuit-informed attacks can be more effective than random perturbations, highlighting the potential of circuit knowledge for designing adversarial attacks.

1 Introduction

Although transformer-based language models (Vaswani et al., 2017) demonstrate remarkable capabilities in a wide range of NLP tasks, their susceptibility to adversarial examples (sometimes called adversarial attacks) (Szegedy et al., 2014) contributes significantly to their failures. A critical aspect of developing practical solutions to resolve model vulnerabilities involves not only the assessment of models' robustness to adversarial examples, but also studying the underlying inference mechanisms and how these mechanisms are disrupted by such attacks. Recent advances in the area of mechanistic interpretability (Wang et al., 2022b) provide a promising foundation for understanding reasoning mechanisms with respect to the computational structure of the model (Nanda, 2023).

To the best of our knowledge, no prior work has systematically analyzed the mechanistic behaviors of models exposed to adversarial inputs. While numerous studies have focused on the development and implementation of adversarial examples and defenses, the underlying mechanisms that govern model responses to these perturbations remain unexplored. This gap in the literature limits our ability to design effective defenses.

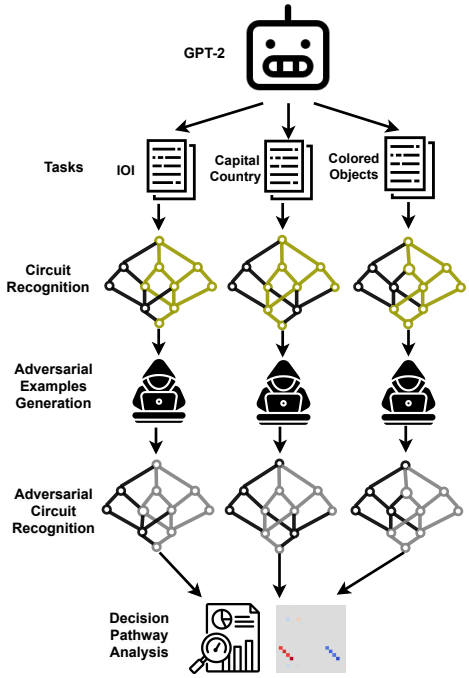

Figure 1: The proposed experimental pipeline[1]. We use three datasets which represent downstream tasks each described by circuits using the EAP-IG method. By constructing adv. examples to uncover adv. circuits and comparing them with the original ones we gain insights into the impact of adversarial examples on the model.

Understanding how models handle adversarial examples is crucial for enhancing their defences against adversarial examples. This is particularly important for privacy protection, as the models should be protected against leaks of sensitive and private data. In the context of circuits, targeted attacks can be more effective and less costly than random attacks. Understanding circuits allows us to identify precise targets, making interventions more strategic and efficient. Using circuit-level insights, we can move beyond trial-and-error ap-

[1]This figure has been designed using resources from Flaticon.com

proaches and develop more informed and impactful methods to influence model behaviour. Designing protective circuits could allow for more precise model security by identifying and reinforcing the components and connections that are key to resisting attacks. Users of LLMs apply prompt injection to exploit model vulnerabilities. Black-box attacks, which introduce minimal text changes, effectively bypass NLP filters, highlighting the need for deeper adv. circuit analysis. A structured approach to studying these perturbations can reveal systematic weaknesses and enhance adv. robustness.

In this paper, we propose a mechanistic approach to analysing models in the context of TextBugger (Li et al., 2019) attacks. We analyse the patterns of activation changes in task-specific circuits under task-oriented adversarial examples. Focusing on black-box attacks, we study three well-researched tasks: Indirect Object Identification (IOI), Colored Objects (CO), and Capital Country (CC). Through a series of experiments, we analyse how token sequence perturbations affect model activations. Our key contributions are as follows.

- We designed two methods to describe the behavior of adversarial examples using mechanistic circuit analysis, marking the first attempt to systematically map their influence on circuit components. This approach provides insights into how individual elements respond to perturbations to understand the model's internal mechanisms and vulnerabilities.

- We present the first in-depth analysis of adversarial perturbations (TextBugger attacks) on three well-studied mechanistic downstream tasks, revealing systematic patterns in circuit activations that have not been previously documented.

- We showed that targeted attacks using circuit knowledge can be more effective than conventional random perturbations, highlighting previously unknown vulnerabilities.

- We suggest how to utilize acquired knowledge to mitigate the threats posed by adversarial examples.

Our findings might open new avenues for designing robust model defenses strategically reinforcing critical circuit components, offering a fresh perspective on adversarial robustness.

2 Related Work

Our contribution draws mainly from two subareas of AI - mechanistic interpretability of large language models and adversarial examples. Adversarial examples (AE, adversarial attacks) in the broad sense are carefully modified inputs to AI models that cause their incorrect responses. They are currently considered a critical challenge to the trustworthiness of AI systems, especially for DL-based ones. Since the seminal paper (Szegedy et al., 2014) there have been tens of thousands of related works on this topic, most of which can be found in the constantly growing list (Carlini, 2019-2025).

The most spectacular results regarding AE concerned the classification problem and there are still relatively few works on AE in the context of LLMs. In general, the problem of AE in text processing fields seems to be still underinvestigated. A broad overview of threat methods in LLMs (incl. AE) can also be found in (Jia et al., 2024), where one can find a description of the global challenge regarding LLM security. In the recent research significant effort has been put into "jailbreaking" attacks that are specific AE which aim to synthesize adversarial prompts to induce targeted and possibly harmful behaviors from LLMs. In this vein the bulk of research concentrates on pointing specific dangers related to LLM or propose some protection methods like filtering the output (e.g. (Wang et al., 2022a)) or scrubbing the training set (Lukas et al., 2023).

Only a few papers are trying to explain what mechanisms cause these attacks to work. A notable result can be found in (Wei et al., 2023). They point to competing objectives and mismatched generalization as the reasons why AE may be possible in LLMs. Our work explains how AEs emerge through network structure changes. We focus on forcing models to produce incorrect answers within a specific domain, even for simple, well-known problems. (described in Section 3.1). We believe that such analysis can be a step towards understanding more complex types of adversarial examples.

To our knowledge, there are no works that deal with explaining the phenomenon of AE from the point of view of mechanistic interpretability. In the relatively close paper (uit de Bos and Garriga-Alonso, 2024) authors investigate what the largest difference between the original model and the circuit representing it can be for a fixed input, where a circuit in the context of transformer language models (Wang et al., 2022b) is described as the smallest

computational subgraph of the model that accurately mirrors the model's overall behavior for a specific task. This is therefore a typical worst-case analysis for the adversarial model. Such an input is referred to as adversarial example, although of course it does not have to be an AE in the sense used in our work. The work of (uit de Bos and Garriga-Alonso, 2024) focuses on demonstrating the weaknesses of the circuit as an approximation of the original model, not on understanding the nature of AEs and eliminating them from the original model.

3 Circuit Recognition Methods

Circuit recognition in language models can be costly because of the need to identify key components and predominant connections between them. Testing these connections is expensive, especially in large models such as GPT-2 small, medium, or large (Hanna et al., 2024).

Edge Attribution Patching (EAP) To address the efficiency issue, (Nanda, 2023) developed edge attribution patching (EAP), a method that estimates the impact of connections with minimal computational passes through the model. The effectiveness of EAP is supported by its high overlap with manually found circuits, a common measure of success in circuit-finding research. EAP is based on gradient computation procedure that is used to approximate causal interventions on each model's edge with two forward passes and one backward pass. The EAP describes the transformer model as a computational graph, where the circuit executing a specific task can be represented as a subset of nodes and edges of the original graph. In order to find the circuit, the scoring function for each model edge is needed. The EAP propose for to approximate this value by usage of negative loss change in each edge between clean and corrupted run.

Edge Attribution Patching with Integrated Gradients (EAP-IG) (Hanna et al., 2024) is an improvement over the EAP method which introduces integrated gradients technique (IG) (Sundararajan et al., 2017) into circuit recognition framework. The addition of IG aims to improve the EAP method when facing zero gradients. The score function uses accumulation of gradient, which helps in situation of facing zero gradients along path between clean and corrupted activations.

3.1 Circuit Recognition Tasks

The **Indirect Object Identification (IOI)** task (Wang et al., 2023) consists of pairs of sentences with unique names, which describes situation when two people perform activity. The goal is to predict name based on the context. For example: "When Mary and John went to the bar, John gave a drink to", models task is to predict "Mary". The task score is measured by logit difference: logit of the IO ("Mary") minus logit of the subject (S) ("John"). Corrupted examples are created with replacing of third name with different name that consists of the same number of tokens.

The **Capital-Country (CC)** task introduced in (Hanna et al., 2024) checks model's ability to predict country name based on name of its capital. The model receives the sentence "Bridgetown, the capital of" and should predict the country name ("Barbados"), in corrupted settings the capital is changed to another, for example "Lilongwe". Performance of task is measured by logit difference between correct country and corrupted one.

The **Colored Objects (CO)** task introduced in BigBench (BIG-bench authors, 2023) and simplified for circuit recognition task in (Merullo et al., 2024). The task checks model ability to identify the color of an object that was previously mentioned in context alongside other objects. In each example model receives information about three objects and their color, followed by question what is the color of one of the objects. The corrupted version is obtained by asking for different object color. The performance in this task is measured by checking the probability of desired color tokens.

The IOI circuit head groups identified in (Wang et al., 2022b) demonstrate how, in IO tasks, heads are categorized into classes that perform a specific task. The authors propose seven such groups of attention heads for an indirect object identification task, but for purpose of this work we will focus on thrre of them: Previous Token Heads (PTH), Duplicate Token Heads (DTH) and Induction Heads (IH). DTH connects duplicated names in sentence, they add attention on second name occurrence (S2) and maps this name to the first occurrence (S1). Simultaneously IH map the name S2 to S1+1, letting the model know that the name S is duplicated and pointing to its follower. The PTH attend to connect first occurrence of duplicated name (S1) to its following token (S1+1).

4 Adversarial Circuit Analysis

Our goal is to conduct a mechanistic analysis of attention heads by perturbing tokens in task-relevant phrases. Attention heads serve different functions in the model (Wang et al., 2022b), however their understanding is limited to non-adversarial cases. In adversarial circuit analysis, we aim to examine how distinct token disruptions affect these various functions. Specifically, we want to understand the impact of perturbations in phrases that are critical for downstream task on the behavior of attention heads, which are central to task target. To study the effects of adversarial attacks on task-specific circuits in the mechanistic analysis of language models, we designed two experimental protocols for adversarial attacks: gradual token disruption and adversarial token injection.

4.1 Designing adversarial examples

Generation of adversarial example involves changing some of model's input tokens, preferably the most important ones, in the way that the created adversarial sample changes the original model prediction. One of the techniques for attacking models text input is TextBugger (TB) (Li et al., 2019). Other adversarial generation methods such as TextFooler (Jin et al., 2020) have a limitation of changing all words tokens at once; the main reason for using TextBugger is the ability to attack single tokens, which greatly increase research options. Identification of important input's words, most cases depends on given task, is crucial for adversarial examples performance. For the IOI task, three words can be identified as crucial for correct output generation: IO, duplicated word first (S1) or second (S2) occurrence. Changing the IO is not considered as it would change the correct model response. Similar analysis can be obtained for the CO, where instead on one IO and dupicated S, are three color, object pairs. In this setting when asking for one of the objects color, the duplicate objects occurs and respectively to IOI, first occurrence of this object can be attacked. For the CC only one important word can be identified in each sample the capital name.

Gradual adversarial token disruption (GATD) The method of gradual adversarial token disruption involves a systematic approach to perturbing text at the token level for mechanistic analysis of attention heads. Given a text composed of n tokens $T = (t_1, t_2, \ldots, t_n)$ and a multi-token phrase $F = (f_1, f_2, \ldots, f_k)$ being a subsequence of T, we create multiple variants of the text by token-level alternations of phrase tokens by applying function $g(F, i)$.

$$g(F,i) = \{(f_1, \ldots, f_{i-1}, f_i', f_{i+1}, \ldots, f_k) \mid f_i' \in S(f_i)\} \quad (1)$$

$i = 1, \ldots, k$, where $f_i' \in S(f_i)$ is a set of possible alternations of token f_i in phrase F.

The set of all single-token alternations of phrase F is defined as follows:

$$\mathcal{F}(F) = \bigcup_{i=1}^{k} g(F,i) \; .$$

For task-specific dataset $D = \{T_1, T_2, \ldots, T_m\}$ consisting of texts $T_j = (t_{j,1}, t_{j,1}, \ldots, t_{j,n_j})$, with $j = 1, \ldots, m$, we generate disrupted versions T_j' of the text T_j by alternating the tokens of phrase $F_j \subseteq T_j$ in a sequential manner, using $g(F_j, i)$. Let D' be an adversarial dataset derived from D, consisting of texts created through phrase-specific perturbations. Let $D' = \{\mathcal{S}(T_j, F_j) \mid j \in \{1, \ldots m\}\}$, where $\mathcal{S} : (T_j, F_j) \mapsto T_j'$. The set D' contains texts in which phrase F_j has been replaced with a new phrase F_j', where F_j' is the result of replacing a single token f_i at position i by applying $g(F, i)$. The new phrase F_j' belongs to the set of possible single-token replacements $F_j' \in \mathcal{F}(F_j)$. In practice, each text T_j consists of multiple phrases $F_{j,1}, F_{j,2}, \ldots, F_{j,l}$ and one can generate adversarial examples based on these phrases collectively. In our work, we consider one type of phrase at a time (e.g. S1, S2, or IO) when generating an adversarial dataset, which is consistent with the methodology of mechanistic analysis.

The process starts by choosing a single token to alter, ensuring it doesn't affect the original meaning or form. This creates a new version of the text with one changed token. In GATD, we sequentially replace each token of phrase F_j to obtain multiple variants T_j' of the same text T_j (see Figure 2), ensuring that the newly introduced tokens do not replicate or closely resemble the original tokens.

Adversarial token injection (ATI) Adversarial token injection extends the concept of gradual token disruption by introducing a more controlled manipulation of token placement within a given phrase. While gradual token disruption focuses on progressively altering or removing tokens to

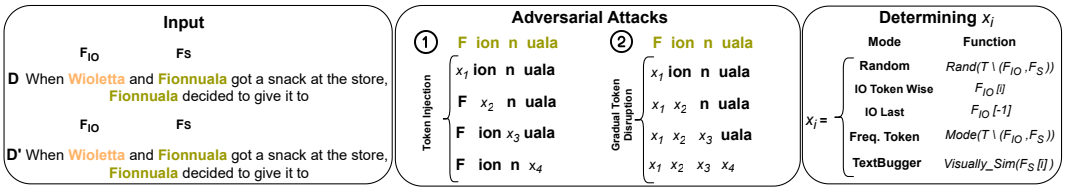

Figure 2: Methodology of adversarial example creation. Two techniques, token injection and gradual token disruption, are presented. Adversarial changes modify each F_S occurrence using one of five substitution methods. Note that $Mode$ means the most frequent element of given set, the $Rand$ indicates probing random element from set, $F_{IO}[i]$ i-th token from sequence F_{IO} and $Visually\_Sim$ means replacing token with its visually similar equivalent.

observe how these changes impact the model's response, ATI goes a step further by inserting the tokens from source phrase $F_j^s \subseteq T_j$ to target phrase $F_j^t \subseteq T_j$. This process consists of replacing elements from F_j^t with the elements of F_j^s considering different positioning within F_j^t and resulting in new variations T_j' of source text T_j. The main objective of this approach is to explore how the insertion of tokens from one part of the text into another can influence models' circuit, particularly its attention mechanisms and its understanding of context:

$$g(F_j^t, i) = f_i^s,$$
$$f_i^s \in S(f_i^t) = F_j^s.$$

This approach is inspired by circuits detected in IOI tasks, where certain attention heads of the model focus on specific elements of analysed text. For example, attention heads such as the Duplicate Token Head (Wang et al., 2022b) may focus on specific phrase pairs – in this case, F_j^s = S1 and F_j^t = S2 phrases. As attention heads in a model can focus on specific parts of the input, ATI allows to test how different parts of a source phrase interact when introduced into a target phrase, potentially triggering specific attention patterns. After applying $g(F_j^t, i)$, the alternative phrase of F_j^t becomes $(f_1^t, \ldots, f_{i-1}^t, f_i^s, f_{i+1}^t, \ldots, f_k^t) \in g(F_j^t, i)$. The methodology of adversarial example creation is presented on the figure 2. Two techniques are presented: token injection and gradual tokens disruption. Adversarial change can be applied to each of F_S occurrence with substitution of tokens using one of five methods, namely change of tokens to random, retrieved token wise or fixed position from other important word (e.g. IO), replacing to frequent token from the input or using TextBugger visually similar change. The proposed techniques are used to generate adversarial examples and datasets in the pipeline presented on figure 1.

5 Experimental Setup

Hyperparameters Experiments were conducted on the GPT-2 model[2] large language model. The hyperparameters for the circuit recognition method EAP-IG were integrated gradient steps 5, top edges 100. For all datasets four token equivalents of important words were chosen, in this way for the GATD we created four versions of each dataset, where gradually from 1 to 4 tokens where changed in chosen important words. For the adversarial token injection, version with one of token changed was chosen as base for AE creation. The datasets have 1000, 200 and 30 unique samples for respectively CO, IOI and CC tasks.

6 Results and Discussion

In the experimental part of this study, we aim at finding how the adversarial examples affect task circuits and what the properties of an attacked circuits are. The adversarial example should be made in a way that shows how gradual change in important words, defined as perpetuating single tokens, affect components of the tested circuit. We examine the circuits behaviour by using the proposed two techniques GATD and ATI.

6.1 Which heads are vulnerable to text bug attacks?

Based on previous work in the area of analysing attention heads (Wang et al., 2022b), we investigate task-specific circuit heads in adversarial setting using gradual token disruption. The main findings can be grouped according to head types: Duplicate Token Heads, Previous Token Heads and Induction Heads. The key observations are as follows.

Typically, **Duplicate Token Heads** (DTH) perform mapping between duplicated words in the text, e.g. attending to S1 and S2 names in the IOI task. Under adversarial attack, where we alternated 4

[2] https://huggingface.co/openai-community/gpt2

tokens of the name being attacked (gradual token disruption), a drift of mapping can be observed (Figure 3). For altered tokens, the attention mapping in `a1.h0` is changed and no longer focuses on S1 and S2, shifting the activation of the second occurrence of the duplicated word to itself (S2 to S2). Moreover, the activation shift pattern obtained in yet another DTH head `a3.h0` differs from the pattern obtained in `a1.h0`. Instead of shifting the mapping towards S2 itself on corrupted tokens, attention is directed to the end of text (EOT) token. This might be explained as the result of error accumulation across different layers under the influence of adversarial examples.

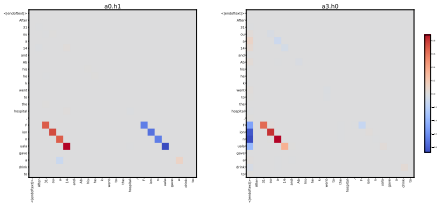

Figure 3: Activation differences between clean and corrupted text in `a0.h1` and `a3.h0` (DTH), under change of four tokens. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

The **Previous Token Heads** (PTH) are responsible for connecting tokens to their subsequent tokens in the text. When facing adversarial example, the attention mapping should be close to diagonal breaks towards left or right side, especially changes occure in mapping duplicated word tokens (S1 and S2) to their followers, which results in mapping to tokens S-1 or S+2.

The **Induction Heads** (IH) performs a similar task to Duplicated Token Heads, but instead of connecting duplicated words, uses the induction mechanism for mapping follower tokens of the first duplicated word occurrence (S1+1) to the second word occurrence (S2). Notably in this case the attention mapping shift towards EOT token, effectively disrupting the connection. Example of such gradual change is shown on 4.

The attention heads discussed in this section were originally identified for the IOI task, but the same circuit heads exhibit similar behavior in the CO task. In contrast, for the CC dataset, the DTH and IH heads remain inactive, as there are no duplicated words in the CC test set. However, the PTH heads discovered in the IOI task demonstrate the same behavior across CC and CO tasks.

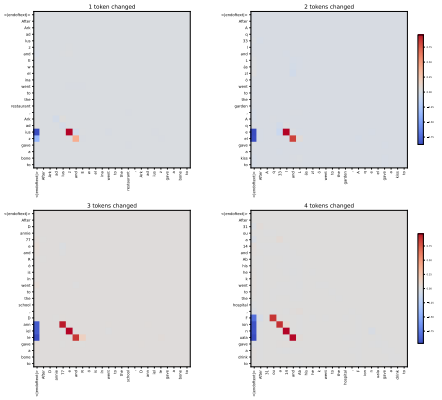

Figure 4: Activation differences between clean and corrupted text in `a5.h5` (IH) distinguished by the number of modified tokens. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

6.2 How adversarial attacks affect circuit structure?

In order to answer this question, we examined how adversarial attacks affect the effectiveness of each downstream task. For all three tasks, we measured the impact of attack by analyzing changes in the probability of the desired output under adversarial perturbations. The results of this experiment are presented on Figures 5 (IOI) and 6 (CO and CC). The plots illustrate probability distribution of task label (IO) under adv. attack with prob. distribution of clean input as a reference. For IOI task, gradually increasing the number of disrupted tokens from one to four progressively lowers the output probability of IO tokens. We illustrate the attacks for both S1 and S2 targets, which were independently tested during this experiment. The impact of attack location on the task goal is noticeable but inconsistent. For 1- and 3-token changes, attacking S2 phrase led to higher effectiveness of the attack, whereas in other cases, the opposite effect was observed. Notably, in IOI task the modification of S2 phrase affects semantics of the second sentence – the actual task question – which is why we focused on perturbing only S1 phrase in further experiments. For the CO and CC datasets, adversarial examples gradually degraded the models' task performance.

The systematic change in the number of nodes and edges between baseline and TB corruption can be seen in Table 1. Generally, TextBugger corruption leads to a decrease in the number of nodes and edges. This is due to the use of the same minimum edge score in the EAP-IG method for both baseline

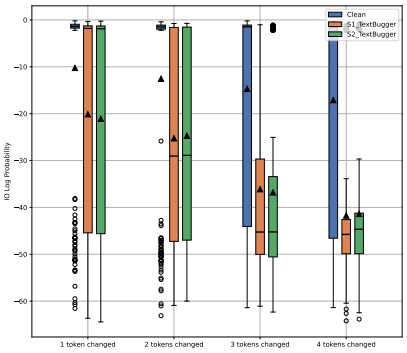

Figure 5: Distribution IO probability on adversarial examples crafted with modification of 1, 2, 3 or 4 tokens, divided by type of attack.

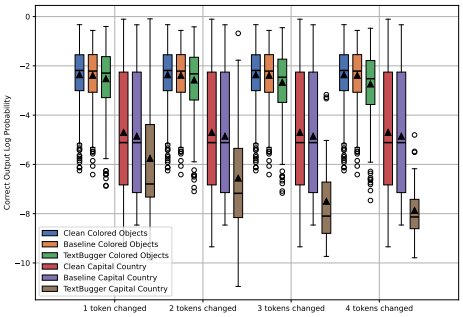

Figure 6: Distribution of correct output log probability on clean and corrupted examples for CO and CC.

and TextBugger graphs. This suggests that adversarial examples lower the edge score, effectively removing nodes and edges from the circuit.

To assess the impact of perturbations on circuit structure, we represented the recognized circuits as graphs using the EAP-IG method. To compare differences, we developed a visualisation approach that overlays two graphs (perturbed and baseline) to highlight shared and newly introduced nodes and edges. Figure 7 shows an example comparing the IOI baseline run with one affected by TextBugger corruption. The triangular nodes represent additions in the TextBugger graph absent in the baseline. Notably, DTH (`a0.h1`, `a3.h0`) increased their importance on the TB graph, but one of IH `a5.h5` lower the importance. This observation likely stems from shifts in attention head patterns which originally attempt to connect S1 and S2 instances but fail under adversarial inputs.

Table 1: Nodes and edges number for Baseline (BS) and TextBugger (TB) run of the EAP-IG. Data are presented for IOI, CC and CO tasks. For TB runs can be distinguished between change between 1 and 4 tokens, BS runs follows techniques according to the task definition.

	Tokens changed	IOI		Capital Country		Colored Objects	
		BL	TB	BL	TB	BL	TB
Nodes	1	27	25	29	20	25	22
	2	27	18	29	21	25	22
	3	26	31	29	22	25	20
	4	26	25	29	22	25	26
Edges	1	74	64	88	42	61	55
	2	83	40	88	48	61	57
	3	77	75	88	49	61	54
	4	77	57	88	48	61	63

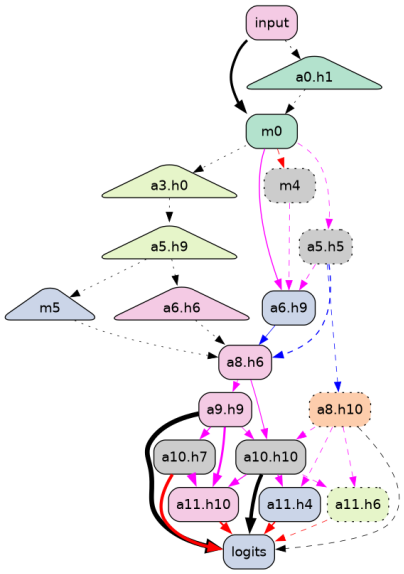

Figure 7: Graph comparison of clean vs. corrupted runs of the model, showing the 50 most important edges. Nodes represent model submodules: rectangular (present in clean), dotted (removed in corrupted), and triangular (added in corrupted). Edge thickness reflects connection importance.

6.3 Does adversarial effectiveness vary with token position?

To examine the impact of injecting AE on various token position we used ATI. For this purpose, four datasets with attacks were created by substituting tokens with: random token, corresponding token from IO/Color, last IO/Color token, and frequent token from the text. The overall attack performance is measured in terms of the reduction in the probability of the correct output – the results of this experiment are presented in Figures 8 (IOI) and 9 (CO). For both tasks, injecting the last token of IO/Color significantly reduces the probability of the desired output. The experiment shows that the position of the attacked token is less important than the method of injection. Furthermore, the injection of frequent token performs similarly in CO task but stands out for IOI. This can be explained by the fact that CO examples tend to be more schematic with multiple tokens repeated, whereas in IOI the tokens are more unique.

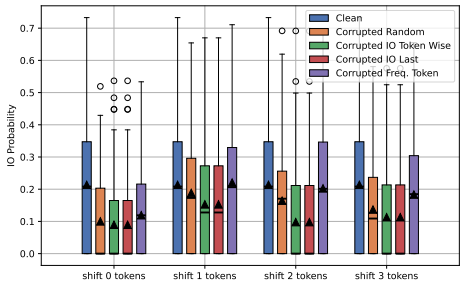

Figure 8: Distribution IO probability on adversarial examples over dataset divided by type of attack.

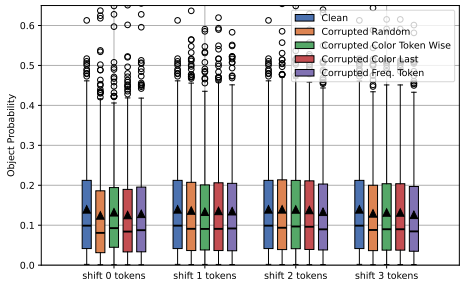

Figure 9: Distribution Object probability on adversarial examples over dataset divided by type of attack.

To analyse the factors that contribute to the greater effectiveness of one attack compared to another, differences in attention maps were investigated. The first difference between IO/Color last and random settings can be seen in DTH 10, this head should link the S2 token to S1, signalling that the name is duplicated. However, the attack in both settings alters this mapping, redirecting S2 to itself at the attacked position. Notably, in the first setting the attention is also shifted from IO last token to altered S1 token, which may explain the difference in performance between those attacks.

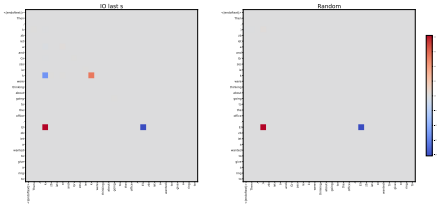

Figure 10: Activation differences between clean and corrupted text in `a0.h1` (DTH) for the IO last and Random token injection on position 0. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

6.4 How to mitigate the threat of circuit-oriented adversarial examples?

Although we were unable to conduct particular experiments aimed at securing models against circuit-based targeted attacks, one can suggest promising approaches for defending against such attacks:

- Perturbation-based data augmentation – training the models on augmented datasets with perturbations via token copy-pasting (adversarial token injection, Sec. 4) might significantly increase the models' robustness to subtle token changes, but it may not necessarily solve the problem of targeted attacks completely.
- Counterfactual data design – creating counterfactual examples, where specific elements such as certain tokens or phrases are intentionally altered, can teach the model to focus on the most relevant parts of the input rather than being overly sensitive to targeted perturbations.

We believe that while regularization techniques may help reduce the model's sensitivity to typical perturbations, they are unlikely to fully defend against sophisticated targeted attacks that manipulate specific components of the model, such as task-specific attention heads. In this context, the techniques such as perturbation-based data augmentation and counterfactual data design are more likely to be effective in improving robustness of modern language models. We suggest that further exploration of this topic is important for developing more effective defenses against targeted attacks on model circuits.

Conclusions In this paper, we proposed how to investigate LLM circuits using adversarial example, especially with two new techniques: GATD and ATI. The main findings of analysing the GPT-2 model under three tasks (IOI, CO, CC) are:

- In the Sec. 6.1 we presented how gradual token disruption affects attention heads, especially how attention changes and how the error propagates in the DTH, PTH and IH head groups, additionally we shown that adversarial activation patterns are shared across tasks.
- Impact of GATD on model performance in tasks was examined in Sec. 6.2 were we shown that model behave differently when facing TB attack and swapped name even though same number of tokens was changed.
- The position of ATI usage has less impact on attack performance than the chosen injection technique, which varies in effectiveness depending on the task Sec. 6.3.

Limitations

Our experimental analysis was conducted on the tasks of a similar type, i.e. IOI, Colored Object, and it remains unclear whether the observed effects generalize to other tasks. Additionally, we do not know whether the same attention heads are used across significantly different tasks. The observed attention head patterns might be strongly related with task types. However, as shown in the appendix, certain attention heads such as PTH (Previous Token Head) appear to be shared across different tasks.

We were not entirely able to explain why the attention heads exhibit the behaviors observed in our experiments. The patterns described in the 6.1 section, such as mapping to EOT, may be highly model-specific and could result from incidental behavior of a particular GPT-2 model family instance rather than representing a fundamental property of attention heads. However, the results presented in 6 were obtained for two different models, GPT-2 small and GPT-2 medium, suggesting some level of consistency between model scales and different training initialisation.

Some models struggle with more complex tasks due to their limited parameter count or due to the lack of SFT tuning. This suggests that our findings should be valid mostly for pre-trained models only, and future research could extend our findings on a broader range of models to assess their generalizability.

Acknowledgments

Anonymised

References

BIG-bench authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Nicholas Carlini. 2019-2025. A complete list of all (arxiv) adversarial example papers.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Xiaojun Jia, Yihao Huang, Yang Liu, Peng Yan Tan, Weng Kuan Yau, Mun-Thye Mak, Xin Ming Sim, Wee Siong Ng, See Kiong Ng, Hanqing Liu, Lifeng Zhou, Huanqian Yan, Xiaobing Sun, Wei Liu, Long Wang, Yiming Qian, Yong Liu, Junxiao Yang, Zhexin Zhang, Leqi Lei, Renmiao Chen, Yida Lu, Shiyao Cui, Zizhou Wang, Shaohua Li, Yan Wang, Rick Siow Mong Goh, Liangli Zhen, Yingjie Zhang, and Zhe Zhao. 2024. Global challenge for safe and secure llms track 1. *Preprint*, arXiv:2411.14502.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8018–8025. AAAI Press.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella Béguelin. 2023. Analyzing leakage of personally identifiable information in language models. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 346–363. IEEE.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Circuit component reuse across tasks in transformer language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Neel Nanda. 2023. Attribution patching: Activation patching at industrial scale.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *Preprint*, arXiv:1312.6199.

Niels uit de Bos and Adrià Garriga-Alonso. 2024. Adversarial circuit evaluation. *CoRR*, abs/2407.15166.

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

Boxin Wang, Wei Ping, Chaowei Xiao, Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Bo Li, Anima Anandkumar, and Bryan Catanzaro. 2022a. Exploring the limits of domain-adaptive training for

detoxifying large-scale language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does LLM safety training fail? In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Lexical Consistency

A.1 Does lexical consistency of adv. attacks affect the performance of downstream tasks?

One of the key restrictions commonly used in adversarial examples generation is maintaining lexical consistency of modified texts. For IOI tasks in base version, corrupted texts are generated with substitution of S2 name with different name from dictionary. TextBugger attacks substituting given number of tokens with different creating visually similar word however, such substitute is not a lexically correct word. Figure 11 presents differences in the IO log probability distribution between: clean, texts with swapped names at S1 or S2 position, as well as perturbing the original S1, and S2 tokens with TextBugger attack.

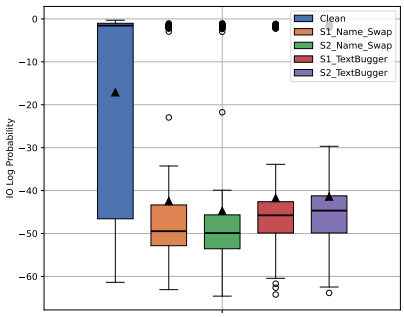

Figure 11: Distribution of IO log probability on adversarial examples on each of four tokens modified.

It can be noted that swapping names on S2 position reduces the most probability of IO, same action on S1 is less successful however under TextBugger attack the relation is inverted. It can be concluded that adversarial disruption based on replacing one of the names with another that is lexically correct reduces indirect object probability more than TextBugger attacks. From a different perspective, TB gives ability to attack words gradually, which is not possible on names swapping.

B Head Disruptions

In this section we present how different attention head groups change when faced with adversarial attacks. Attention maps are distinguished between gradual token disruption of 1 to 4 tokens. Analysis of attention maps presented below is linked to findings presented in Sec. 6.1.

B.1 IOI task

For the Indirect Object Identification task we present attention differences for the Duplicated Token Head 12, Previous Token Head 13 and for the Induction Head 14.

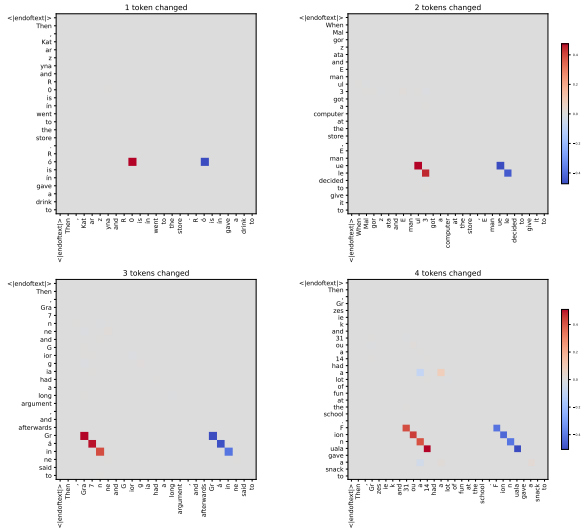

Figure 12: Activation differences between clean and corrupted text in `a0.h1` (DTH) distinguished by the number of modified tokens, for IOI task. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

B.2 Capital Country task

For the Capital Country task we present attention differences for the Duplicated Token Head 15, Previous Token Head 16 and for the Induction Head 17.

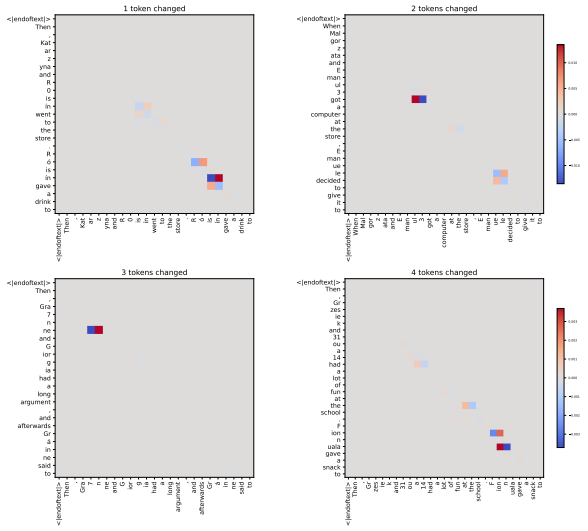

Figure 13: Activation differences between clean and corrupted text in `a4.h11` (PTH) distinguished by the number of modified tokens, for IOI task. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

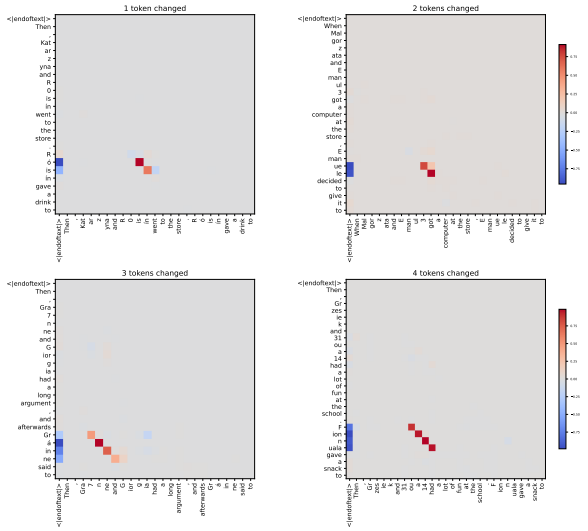

Figure 14: Activation differences between clean and corrupted text in `a5.h5` (IH) distinguished by the number of modified tokens, for IOI task. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

B.3 Colored Objects task

For the Colored Objects task we present attention differences for the Duplicated Token Head 18, Previous Token Head 19 and for the Induction Head 20.

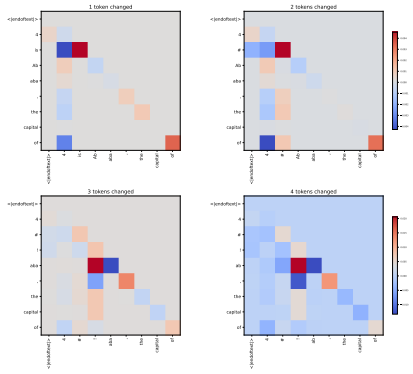

Figure 15: Activation differences between clean and corrupted text in `a0.h1` (DTH) distinguished by the number of modified tokens, for Capital Country task. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

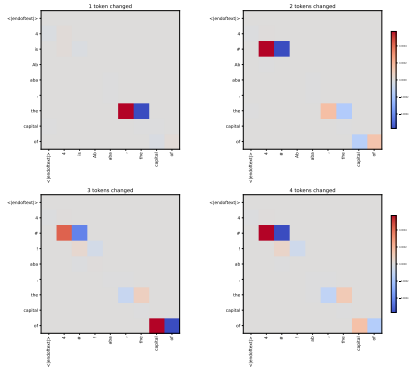

Figure 16: Activation differences between clean and corrupted text in `a4.h11` (PTH) distinguished by the number of modified tokens, for Capital Country task. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

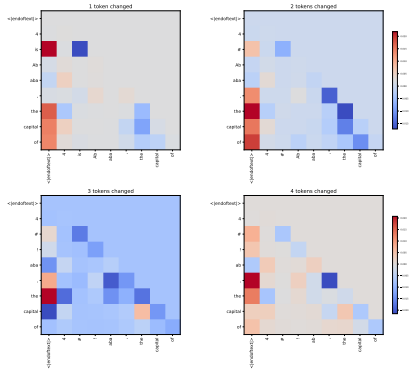

Figure 17: Activation differences between clean and corrupted text in `a5.h5` (IH) distinguished by the number of modified tokens, for Capital Country task. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

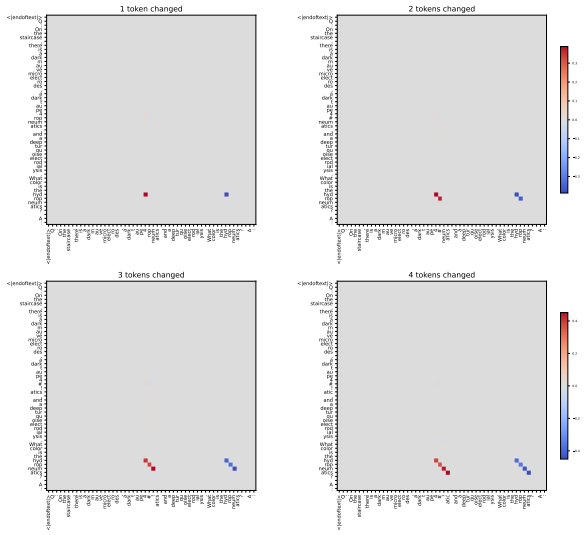

Figure 18: Activation differences between clean and corrupted text in `a0.h1` (DTH) distinguished by the number of modified tokens, for Colored Objects task. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

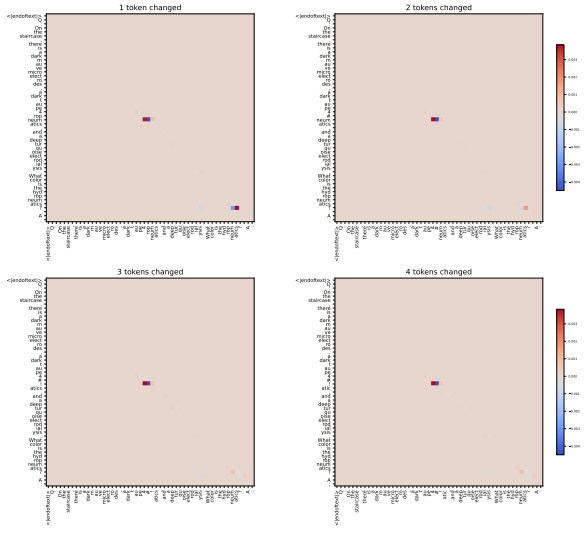

Figure 19: Activation differences between clean and corrupted text in `a4.h11` (PTH) distinguished by the number of modified tokens, for Colored Object task. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.

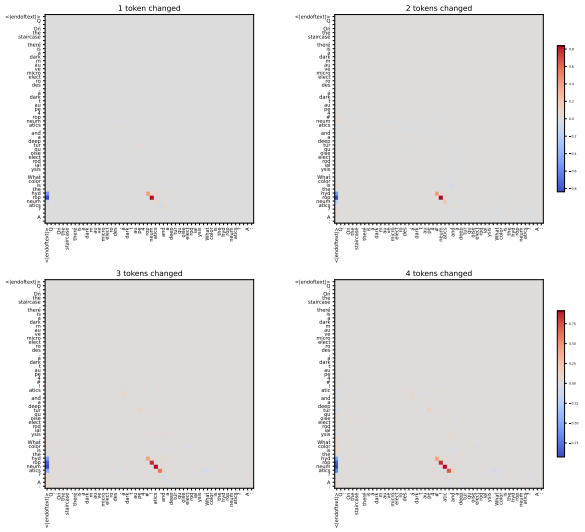

Figure 20: Activation differences between clean and corrupted text in `a5.h5` (IH) distinguished by the number of modified tokens, for Colored Object task. Positive values (red) indicate attention in a clean run, while negative values (blue) show shifts due to corrupted input.