Extracting the Harmfulness Classifier of Aligned LLMs

by

Jean-Charles Noirot Ferrand

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2024

Date of thesis submission: 12/13/2024

The thesis is approved by the following members of the Thesis Committee:

Patrick D. McDaniel
Tsun-Ming Shih Professor of Computer Sciences, School of Computer, Data, & Information Sciences
Thesis Advisor

Rahul Chatterjee
Assistant Professor, School of Computer, Data, & Information Sciences

Chaowei Xiao
Assistant Professor, School of Computer, Data, & Information Sciences

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Large language models (LLMs) exhibit high performance on a wide array of tasks. Before deployment, these models are aligned to enforce certain guidelines, such as harmlessness. Previous work has shown that alignment fails in adversarial settings through *jailbreak* attacks. Such attacks, by modifying the input, can induce harmful behaviors in aligned LLMs. However, since they are based on heuristics, they fail in giving a systematic understanding of why and where alignment fails in adversarial settings. In this paper, we hypothesize that alignment embeds a harmfulness classifier in the model, responsible for deciding between refusal or compliance. Investigating the harmfulness and robustness of the alignment of a model then reduces to evaluating its corresponding classifier, which motivates this work: *Can we extract it?*. Our approach first builds estimations of the classifier from varying parts of the model and evaluates how well they approximate the classifier in both benign and adversarial settings. We study 4 models across 2 datasets and find through the benign settings that the classifier spans at least a third of each model. In addition, the evaluation in adversarial settings shows that it ends before the first half of most models, exhibiting a transferability greater than 80%. Our results show that the classifier can be extracted, which is beneficial from an attack and defense perspective due to the improvements in both efficiency (smaller model to consider) and efficacy (higher attack success rate).

ACKNOWLEDGEMENTS

First, I thank my advisor, Patrick McDaniel, for his trust and guidance throughout this journey, which helped shape the narrative of the project. I consider myself lucky to be working in his lab, a great catalyst to become a better researcher driven by a strong sense of camaraderie. I am also grateful to my labmates and friends — Blaine, Eric, Kunyang, Kyle, Melissa, Owen, Quinn, Rachel, Ryan, Thomason, and Yohan — for their presence and help. Their constructive feedback and suggestions helped ensure the accuracy and rigor of the results of this work. Naturally, I also thank my parents for their unwavering support throughout my studies, no matter how far I was.

Funding Acknowledgement: This material is based upon work supported by the National Science Foundation under Grant No. CNS-2343611 and by PRISM, one of seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 INTRODUCTION

With the unceasing improvements in machine learning, specifically since the introduction of the transformer architecture [41] as a catalyst for larger scale models, many large language models (LLMs) have emerged. Whether they are only accessed through an API (e.g., GPT [31], Gemini [37], Claude [1]) or open-source with weights available to the public (e.g., Llama 2 [28], Llama 3 [29], Qwen [40], etc.), these models have become the de facto tool for tasks involving natural language. They serve as foundations for new tools, in which they are augmented with new capabilities [2, 30] or fine-tuned on a downstream task [14]. Most LLMs undergo an alignment process in which they are further trained to satisfy certain guidelines [28, 35]. In particular, these guidelines enforce that the model's outputs should not be offensive, discriminatory, or harmful. It is well acknowledged that alignment fails in adversarial settings, resulting in a *jailbreak*: a harmful input an aligned model believes to be compliant with the desired guidelines. Many approaches have aimed to automate the generation of such modifications, i.e., adversarial examples [22, 24, 47, 57].

These approaches only reveal instances of vulnerabilities, regardless of the assumed knowledge on the model: they either rely on heuristics for the adversarial objective or the perturbation applied on the input. Previous work has shown that the representations of harmful and harmless queries are well separated in aligned models [23, 55], specifically in the first layers[20]. This suggests that there might be parts of the model responsible for detecting harmful inputs. Therefore, we hypothesize that alignment embeds a *harmfulness classifier* in the model. This classifier is responsible for the separation between harmful and harmless representations and the decision of the model (i.e., compliance or refusal). By identifying its position within the model precisely, defenders can strengthen alignment by improving the robustness of the classifier. Conversely, adversaries can

design more efficient and effective white-box attacks since they require a smaller portion of the LLM (the embedded classifier) and work with a heuristic-free adversarial objective. This motivates the scope of this paper: *can we extract this classifier and ensure the success of the extraction?*

In this paper, we develop an approach to extract this classifier. We verify that it is empirically equivalent to the LLM in both benign and adversarial settings for the classification task. This means that the extracted classifier agrees with the decision of the model in both settings. Specifically in adversarial settings, we consider the following robustness equivalence property as fundamental to judge the success of the extraction: any input is an adversarial example (i.e., induce a misclassification) for the model if and only if it is an adversarial example for the extracted classifier.

Our goal is to find a structure, i.e., part of the architecture of the model, that best represents the embedded classifier of the model. Our approach build estimations of the classifier by adding a simple classification head to a given structure. This classification head is then trained on harmful and harmless inputs to map the features extracted from the structure to the prediction of the model (positive for refusal, negative for compliance). We compare the classifier estimations in benign and adversarial settings to find the closest estimation. For the benign setting, we measure how well the estimations agree with the classifier. For the adversarial setting, we craft adversarial examples on both the model and the estimations and measure how well they transfer (i.e., induce a misclassification for the model they were not crafted on) between one another.

Our experiments span 4 models and 2 datasets. In benign settings, our results indicate that the estimations F_1 score is above 80% with as little as 20% of the models. However, when evaluating on a different dataset, the same score is achieved in twice the fraction of the model. Further, the agreement of the estimations with the classifier is monotonically increasing with respect to the proportion of the model considered. This translates to

a lower bound on the position of the classifier. This trend does not hold in adversarial settings, in which we observe a peak transferability rate above 80% near half of most models when transferring adversarial examples from the estimation to the model. This means that adversarial settings provide a more precise location of the classifier.

In this work, we contribute to the following.

1. We design a rigorous study on extracting the harmfulness classifier of LLMs, evaluated in both benign and adversarial settings across 4 models and 2 datasets.

2. We show that extracting solely through benign settings is not enough, and that an evaluation in adversarial settings is necessary to ensure the correctness of the extraction.

3. We show that attacking the classifier instead of the LLM directly leads to a higher success rate of the attack, as well as a lower computational cost by construction.

Thesis Statement

Alignment embeds a harmfulness classifier in large language models, which can be empirically extracted.

2.1 Large Language Models (LLMs)

Large language model. A LLM models the conditional probability of the next token (a unit of text for the model) with respect to the current prompt or past tokens. Let V be the vocabulary of the tokens and $V^* = \bigsqcup_{n=1}^{N} V^n$ the corresponding input space of sequences, where N denotes the context window of the model (maximum amount of tokens that can be processed by the model). Given a sequence of tokens $x = x_1 x_2 \dots x_T \in V^*$, the model aims to produce the next token x_{T+1} by approximating the probability distribution.

$$p(x_{T+1}|x_1 x_2 \dots x_T) \tag{2.1}$$

LLMs learn a vector representation of words called the embedding space. Given a sequence of tokens $x = x_1 x_2 \dots x_T$, each token is assigned to an embedding, resulting in a sequence of vectors $\{h_t\}_{t\in\{1,\dots,T\}}$ all in $\mathbb{R}^d$. This study focuses on aligned chat models, which are almost exclusively of the same architecture: a sequence of *decoders*. The i-th decoder transforms a sequence of embeddings $h^{(i)}$ in another $h^{(i+1)}$, keeping the same sequence length. If the model is made of D decoders, the probability distribution on the next token is obtained by applying a linear layer followed by a softmax function on the last output embedding of the last decoder, $h_T^{(D)}$, i.e., $p(\cdot|x_1 x_2 \dots x_T) = \text{softmax}(A h_T^{(D)})$, where $A \in \mathbb{R}^{d\times|V|}$ is the learned matrix that maps the embeddings to the scores for each token.

Chat template. By construction, LLMs simply predict the next token. Thus, before deployment, the model undergoes *instruction tuning*: a process which encodes "roles" in the model such that it can interact with a user. Part of this process includes a *chat template*: a fixed modification of the input to dissociate the roles. It relies on special tokens from the model to

enforce a format. As a result of different training settings, different models generally come with different chat template. Formally, for a given model, there exists a template $(x_{pre}, x_{post}) \in V^* \times V^*$ such that any input text x is preprocessed by prepending x_{pre} and appending x_{post}.

2.2 Alignment and Harmfulness

Alignment. Since the training data used for LLMs comes from diverse sources, biases and undesirable behaviors emerge [11]. In order to prevent those behaviors from happening, models often go through an alignment process that regulates their outputs according to given guidelines. Several methods exist and are not mutually exclusive [28]. Approaches such as supervised Fine-tuning (SFT) uses sample answers from humans, while reinforcement learning with human feedback (RLHF) [3, 7] trains a neural network that acts as a reward for the LLM. Since training another reward model may be costly, direct preference optimization (DPO) [35] uses the LLM as its own reward model.

Guidelines and taxonomies. The existence of alignment as a technique is a direct product of the complexity of identifying what is precisely expected from models (i.e., their objective). Harmfulness is a broad term, thus, previous work has been identifying taxonomies [16, 44] that represent a broad range of harmful behaviors. This allows researchers to evaluate alignment on a finer scale and understand where it could be improved. For example, when the prompts are decomposed into different categories such as self-harm, privacy, harassment, etc., models tend to refuse the prompts related to self-harm and accept the prompt related to privacy [9].

Identifying harmful outputs. Similarly to their inputs, classifying LLMs outputs is challenging. Since aligned models tend to create strong refusal answers for detected harmful prompts, a naive solution is to test whether certain refusal keywords are in the output (e.g., "Sorry", "as a responsible

AI", etc.) [57]. Alternatively, judge models (also LLMs) are trained to classify the harmfulness of outputs with more [16] or less [26] granularity on the classes they predict, similarly to the taxonomies explained in Section 2.2. A limitation of judge models is that they can also be attacked [25, 36], leading to a possible high rate of false negatives (harmful outputs judged as harmless).

2.3 Attacks on LLMs

LLMs, like any class of machine learning models, are prone to attacks. While several exploits exist (e.g., model-stealing [4], text misclassification, etc. [54]), the *jailbreak* exploit remains the most prevalent. Jailbreak refers to inducing compliance on an aligned model for a harmful query, as defined by the alignment guidelines. Jailbreak attacks fall into one of two threat models: white-box, in which model parameters are known, and black-box, in which only access to the output of the model is assumed. While there are nuances in these threat models, we detail attacks for these two threat models below.

White-box. These attacks have been studied on open-weights models such has Llama 2 [28] and Llama 3 [29] or Vicuna [6]. The most prevalent, GCG [57], uses Greedy Coordinate Gradient (GCG) to mitigate the absence of mapping from embeddings (i.e., vectors) to tokens. Following this, many new white-box algorithms have been introduced [8, 45], improving existing algorithms [22, 23, 33, 49], using different objectives [50], or using specific properties of the model (e.g., logits [21], special tokens [51], or generation [15]. In terms of efficacy, attacks like GCG have an average attack success rate (ASR) above 50% [26] on most open-source models.

Black-box. These attacks have been widely studied, given their prevalence in practical scenarios [1, 31]. Most of them are driven by empirical heuristics: they apply transformations to the initial prompt such as ASCII-

art [18], translation into underrepresented languages [46], encoding and decoding [12], or applying pre-existing templates [48]. Black-box attacks achieve remarkable efficacy. For instance, GPTFuzzer [47] can achieve ASR as high as 90% on commercial LLMs such has ChatGPT.

3 METHODOLOGY

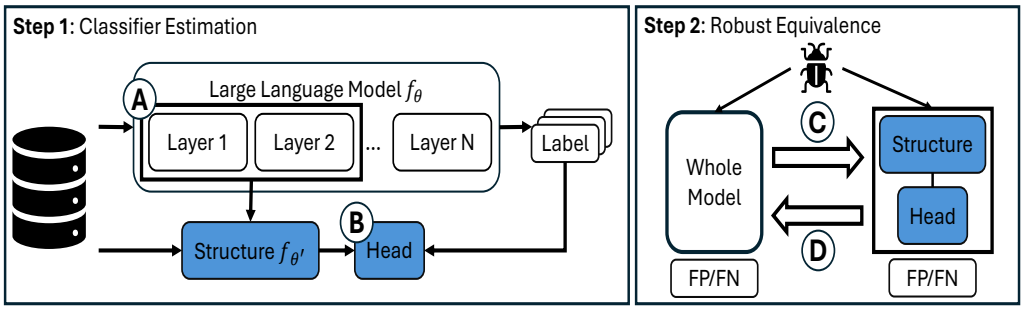


Figure 3.1: Methodology overview. In the first step, we estimate the harmfulness classifier of an LLM by (A) selecting a structure within the model and (B) training a classification head on the predicted labels from the LLM and verify the equivalence in benign setting. Then, we empirically verify that the estimation and the model are equivalent in adversarial setting, i.e., adversarial examples of the LLM (C) transfer to the classifier estimations and adversarial examples of the classifier estimations (D) transfer to the LLM.

3.1 Threat Model

In this work, the adversary aims to extract the underlying harmfulness classifier of a target aligned model. A successful extraction would allow subsequent attacks to be made with more effective white-box jailbreak approaches, while forgoing the complex computational costs they incur, as we discuss Section 5.2. We will estimate the classifier using parts of the models, thus white-box access is assumed, specifically for the model weights and architecture.

In jailbreak attacks, the adversarial objective is to induce compliance from the model for a harmful input by adding a perturbation. Aligned models are heavily trained to refuse harmful inputs. As a result, they can reject harmless prompts [9], exhibiting a high sensitivity to certain

tokens. While previous work has mostly studied jailbreaks (harmful inputs identified as harmless), in this work we focus on extracting the harmfulness classifier. To do so, we also harmless inputs identified as harmful, as we hypothesize that they also provide substantial information on the decision boundary of the classifier.

3.2 Problem Formulation

General formulation. Let f_θ be a LLM with context window N and vocabulary V, and let $\mathcal{R} : x \in V^* \mapsto C$ be a classification rule that maps the output of the LLM to a class from C. We aim to extract a *classifier* from the LLM through two components: a *structure* and a classification head. A *structure* $f_{\theta^*}, \theta^* \subset \theta$ is a part of the original model (e.g., a decoder). A classification head $\mathcal{C}$ maps the representations learned by the structure to a class such that the classifier $\mathcal{C} \circ f_{\theta^*}$ is equivalent to the LLM for the classification task. Formally, for any token sequence $x \in V^*$, the following holds:

$$\mathcal{R}(f_\theta(x)) = \mathcal{C}(f_{\theta^*}(x)) \tag{3.1}$$

Harmfulness. In this paper, we focus on harmfulness classification: classifying harmful and harmless inputs. This is a binary classification problem, with harmful as positives and harmless as negative. A jailbreak is then a false negative, while a false positive is "over-refusal" [9]. The classification rule $\mathcal{R}$ to assign a predicted LLM label is then comparable to how attack success is measured for jailbreak attacks i.e., by verifying refusal or compliance. We chose a model trained to classify refusal and compliant outputs distilroberta-base-rejection-v1 [34] as the classification rule $\mathcal{R}$, since we focus on whether the aligned model refuses (not if it produces harmful information).

3.3 Estimating the Classifier

In order to extract the classifier, we first need to have a set of candidates. We build a candidate through the following three steps, summarized Figure 3.1:

1. Select a part f_{θ^*} of the architecture of the original model f_θ, which we will refer to as a *structure*.

2. Collect data points from the structure $(f_{\theta^*}(x), \mathcal{R}(f_\theta(x)))$ consisting in the features extracted from the structure and the predicted label from the LLM.

3. Train a simple classification head $\mathcal{C}$ on $(f_{\theta^*}(x), \mathcal{R}(f_\theta(x)))$.

The resulting classifier $\mathcal{C} \circ f_{\theta^*}$ will be referred to as an *estimation* of the embedded classifier.

Structures. As explained in Section 2.1, most LLMs are built as sequences of decoders: processing units that take a sequence of vectors and output another sequence of vectors of the same length. LLMs can be scaled either in depth by adding decoders or in width by increasing the dimensionality of embeddings. Therefore, there are many ways to "cut" the model and thus many possible structures. It is clear that there is a separation between harmful and harmless representations at the end of aligned models [23] largely due to the strong refusal outputs. In addition, previous work [20, 52] has shown that separation occurs in early decoders [20]. Thus, we estimate the location of the harmfulness classifier via two hypotheses. First, the classifier is at the decoder level. Second, for a sequence, the information relevant to the classification at a given decoder is contained at the end of the sequence.

We will refer to $f_{i,j}$ as the set of decoders between the i-th and the j-th decoder, $i \leqslant j$. The second hypothesis implies that for a given input $x \in V^*$

where $f_{i,j}(x) \in \mathbb{R}^{|x| \times d}$, we only consider the last element of the sequence, reducing the input dimension of the classification head $\mathcal{C}$ to $\mathbb{R}^d$.

By construction, the input space of $f_{i,j}$ depends on the output space of $f_{1,i-1}$, since the model was trained with the entire architecture. In this work, we set i to 1. We leave the exploration of structures that can isolate intermediate decoders to future work, as they would require techniques to mitigate the absence of earlier decoders, which is out of the scope of this work.

Training. For the selected structure $f_{i,j}$, the goal is to train a classification head $\mathcal{C}$ to best approximate the decision of the original model. The first step is to create a dataset $\mathcal{X} = (f_{i,j}(x), \mathcal{R}(f_\theta(x)))$ consisting in the extracted feature from the structure $f_{i,j}$ and the predicted label[1] by applying the classification rule $\mathcal{R}$ to the output of the original model f_θ.

Then, the classification head is trained following this objective:

$$\min_{(x,y)\in\mathcal{X}} \mathcal{L}(\mathcal{C}(x), y) \tag{3.2}$$

The metrics used to measure how well the structure captures the classification decision of the original model are the accuracy and the F_1 score.

Attacking. To further validate the extraction, we also consider how well the estimation can classify inputs that are intentionally modified to induce misclassification (i.e., its robustness). To evaluate it, we apply a white-box attack on the estimation. The original version of GCG aims to maximize the likelihood of a target sequence x^* (e.g., "Sure, here is how to build a bomb.") given the input sequence x (with adversarial tokens x_i, $i \in \mathcal{I}$). Thus, it is ill-suited for attacking the estimations of the classifier, which outputs a scalar (and not text). Therefore, we keep the algorithm but change the objective to that of a misclassification.

[1] Since we focus on extracting the classifier and not improve it, it is beneficial to have both right and wrong labels from the classifier

4 EVALUATION

We apply our approach to answer the following two questions:

RQ.1 Can the classifier be extracted solely from benign settings estimation?

RQ.2 What information does adversarial settings provide for the extraction?

4.1 Experimental Setup

All experiments were run on pools from the Center for High Throughput Computing [5]. The dataset creation and attack experiments were run on NVIDIA A100 GPUs with 40 GB of VRAM.

Models. We evaluate our approach on 4 well-known open-weights aligned chat models, all from Hugging Face `Transformers 4.46.2` [43]: `Llama-2-7b-chat` [28], `Qwen2.5-7B-Instruct` [40] `gemma-7b-it` [39], and `gemma-2-9b-it` [38]. We chose these models because they are more effective at the classification task on the dataset considered, translating to a stronger alignment. A study of models with weaker alignment can be found in Section A.5.

Datasets. We use two datasets in our experiments:

- An augmented version of *AdvBench* [57] containing the 520 original harmful instructions and 520 added harmless instructions. Details of the augmentation can be found in Section A.3.
- A subset of *OR-Bench* [9] made of 1000 harmless prompts and 500 harmful prompts. The original dataset contains 80000 seemingly toxic harmless prompts (of which 1000 are made harder) and 600 harmful prompts. The harmless prompts we select come from the

easier prompts, to ensure a high enough classification performance from the model.

Labels. The labels are obtained by applying a classifier on the LLM output. To minimize nondeterminism due to sampling, we generate outputs with a temperature $t = 0$: the next token selected at each step of the generation is the one that maximizes the logits. The output is classified using the distilroberta-base-rejection-v1 [34] model.

Classifier training. For the classification head, we consider a simple linear layer followed by a sigmoid: $\mathcal{C}^{(t)}(h) = \sigma(Ah + b)$ where $A \in \mathbb{R}^{1\times d}$, $b \in \mathbb{R}$ and $\sigma : x \mapsto \frac{1}{1+e^{-x}}$. The result is a scalar between 0 and 1, thus the classification is made given a threshold t. As for training, we apply K-fold cross-validation with $K = 5$ folds. We select the threshold t that maximizes the F_1 score on the training data, i.e., $t = \arg\max_{t\in[0,1]} F_1^{(t)}(\mathcal{D}_{\text{train}})$. More information on hyperparameters is provided in Section A.4.

Attack. To generate adversarial examples, we use the `nanogcg`[1] implementation of GCG [57] with the following hyperparameters: `num_steps=250`, `topk=512`, and `search_width=512`. We chose this attack because it is the most popular gradient-based white-box attack. Although several enhancements to GCG have been introduced, we use the default version of the attack. For classifier estimations, we change the original loss to the PyTorch [32] implementation of the binary cross entropy loss.

4.2 Benign Settings

In this section, we answer **RQ.1**, i.e., whether it is possible to extract the classifier only through benign settings.

Baseline results. Table 4.1 shows the benign classification performance of the LLMs for each dataset (see Section A.2 for detailed confusion matrices).

[1] See code at `https://github.com/GraySwanAI/nanoGCG`

Both the accuracy $= \frac{TP+TN}{TP+FN+TN+FP}$ and the F_1 score $F_1 = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$ are reported, since *OR-Bench* is unbalanced. We can first see that all models perform well in *AdvBench*, all scoring above 0.9. The worse performance on *OR-Bench* is expected, as it was introduced after *AdvBench* with the goal of being harder to classify and to test for over refusal. This lower performance means that estimating the classifier on this dataset might be harder than on *AdvBench*, which we see below.

Model	AdvBench		OR-Bench	
	Accuracy	F_1	Accuracy	F_1
Llama 2	0.93	0.93	0.78	0.76
Qwen 2.5	0.97	0.97	0.9	0.87
Gemma 1	0.97	0.97	0.95	0.92
Gemma 2	0.95	0.96	0.88	0.84

Table 4.1: Classification metrics of the models on the two in benign settings.

Estimation performance. Figure 4.1 shows the test F_1 score of the estimation of the classifier. We see that performance either increases or stagnates as more of the model is taken in account. This suggests that later parts of the model either preserve or increase the separation of harmful and harmless instructions. We conducted a more in-depth study in Section A.1 on how well the embeddings are clustered with respect to their label for each dataset, which aligns with the performance in estimating the classifier. For instance, the estimations achieve a lower F_1 score on *OR-Bench*, losing 5% for all models. This aligns with the results in Table 4.1 where all models perform worse on *OR-Bench*. We hypothesize that this can be explained by a lower separability between refused and accepted prompts at the embedding level, which can be seen Figure A.1. This lower separation means that the two clusters may overlap, preventing the classification head from completely aligning with the classifier when training.

Cross-dataset. To ensure that the classifier estimations are not artifacts of the datasets and their distributions, we evaluate the estimations on the dataset that they were *not* trained on (e.g., train on *AdvBench*, test on *OR-Bench*). Similarly to the previous figure, Figure 4.2 reports the F_1 score for the same estimations in the dataset on which they were not trained. Overall, we see that there is no significant performance drop at estimating the classifier when using a different distribution at the last layer. However, it is clear that the F_1 score does not converge as fast on the cross-dataset setting. For instance, in Figure 4.1a, Gemma 1 reaches a F_1 score of 0.9 with only 20% of the model against 50% Figure 4.2a. This offset is likely due to a difference in the distribution of prompts: the estimation overfits on specific patterns tied to how the dataset was built. For instance, *AdvBench* is made of instructions, while *OR-Bench* also contains questions. This difference in format can affect the behavior of the representations. Thus, judging the accuracy of the estimation solely from one dataset likely leads to an underestimation of the classifier.

Takeaway 1: Artifact of benign settings

Estimating in benign settings can lead to an underestimation of the classifier, i.e., an estimation that only captures part of the classifier. This effect can be detected by evaluating the estimations on a different data distribution (e.g., prompts with different formats).

4.3 Adversarial Settings

In this section, we aim to answer **RQ.2**, i.e., understanding the information provided by adversarial settings for extraction. More precisely, we measure whether the following equivalence holds: any input is an adversarial example for the model if and only if it is an adversarial example for the

extracted classifier. We focus on harmful inputs since it is more practical for attacks. A similar study on harmless inputs can be found in Section A.4.

We apply the GCG attack to both the model and the classifier estimations and evaluate the resulting transferability rates: the percentages of adversarial examples crafted on one that are misclassified by the other. To attack the estimation, we modify the GCG loss to the binary cross-entropy loss, more suitable for classification. The attacked samples are prompts for which the classification head was *not* trained on for each fold, with the threshold that maximizes the F_1 score on the training data.

Model	AdvBench		
	Accuracy	F_1	ASR
Llama 2	0.4	0.56	0.22
Qwen 2.5	0.09	0.07	0.94
Gemma 1	0.49	0.43	61.54
Gemma 2	0.3	0.37	66.6

Table 4.2: Classification metrics of the models on *AdvBench* after applying a white-box attack, along the attack success rate (ASR) on harmful inputs.

Baseline results. Similarly to the previous settings, we report Table 4.2 the classification performances of the LLMs after applying the white-box attack, along with the attack success rate (ASR) defined as the proportion of harmful samples misclassified. We note a significant difference (around 70%) between the ASR of Llama 2 and Qwen 2.5. This comes from the strong alignment of Llama 2 with a bias toward refusal (see Section A.2).

LLM to classifier estimations. After applying the attack, we filter the adversarial examples and evaluate how well these examples transfer to the classifier estimations by measuring the transfer rate: proportion of misclassified samples by each estimation. Figure 4.3 reports the transfer rate to the estimations. We observe a similar pattern to that of the cross-dataset study (see Section 4.2): the transfer rate does not converge until at least half of the model is used. For example, classifier estimations on

Llama 2 obtain a transfer higher than 90% only when more than half the model is used. This result implies that classifier estimations that use less than half of the model are not representative of the classifier.

Attacking estimations. Figure 4.4 reports the ASR of the attack on the estimations, which corresponds to the percentage of harmful modified samples from the test dataset that were labeled harmless by the estimation (using the best threshold defined above). We note that when the estimation uses the entire model, the corresponding ASR is close to the baseline ASR (e.g., 20% for Llama 2).

Classifier estimations to LLM. Figure 4.5 shows the proportion of adversarial examples crafted on the estimations that induce a misclassification on the model. Interestingly, the trend does not increase monotonically with the fraction of the model used, as in the previous sections. For three of the models, we observe a peak above 70% transfer rate when the estimations use 60% of the corresponding model. In addition, attacking the estimations and transferring the resulting inputs to the model seem to work better than attacking the model directly. For example, using 50% of Llama 2, it is possible to achieve an ASR of 70%, well above the baseline ASR of Table 4.2.

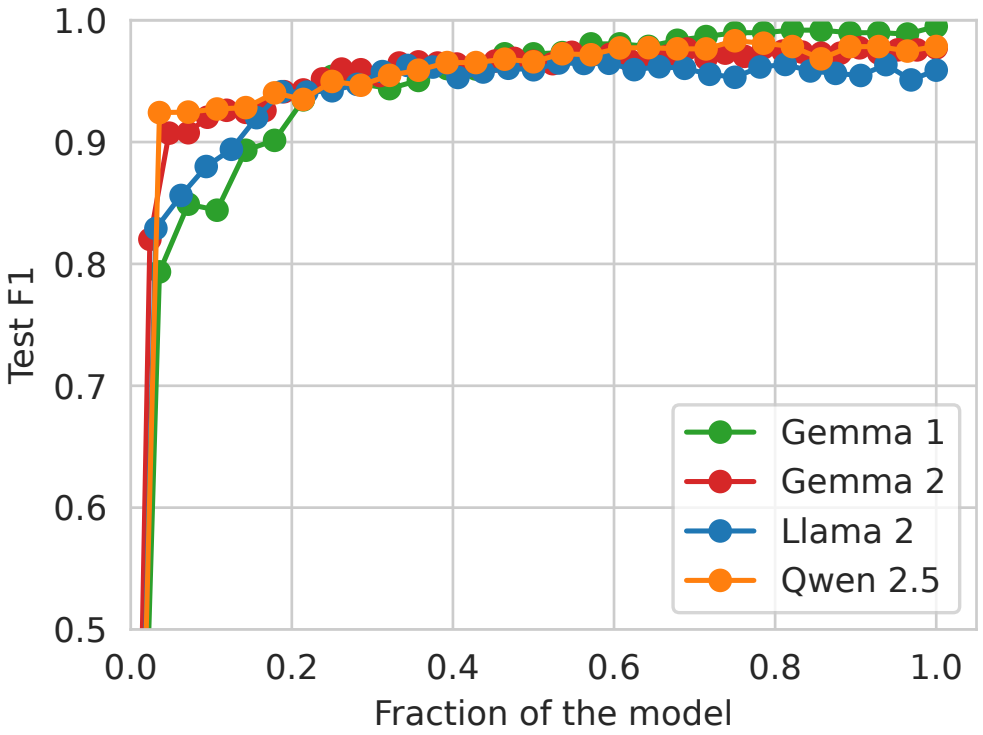
Underestimation and overestimation. The existence of a maximum of the estimation performance highlights the two possible pitfalls when estimating the classifier: underestimation and overestimation. The former means that the estimation captures a subset of the classifier, which limits the transferability. In contrast, overestimation occurs when the classifier is contained within the estimation. Overestimating leads to a lower transfer rate, since the adversarial objective includes information irrelevant to the actual classification.

ASR and objective. The previous differences in success rate compared to the baseline lies in how the adversarial objective is built. Since the output from the whole model is text, there needs to be a heuristic to know

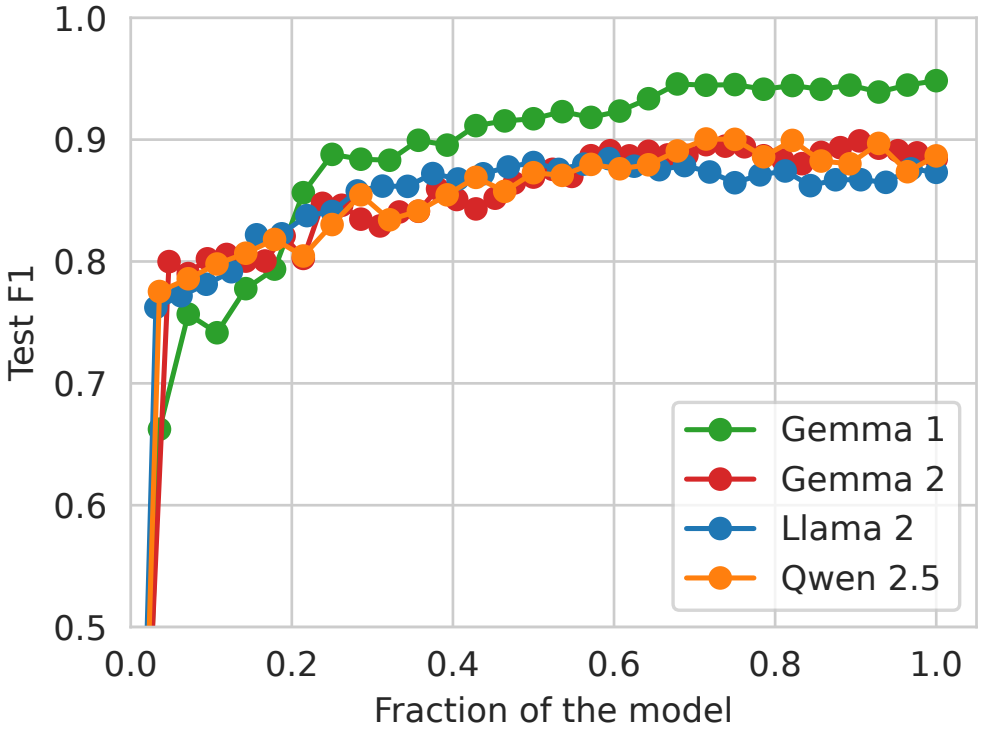
when misclassification (jailbreak) occurs, such as specific target sentences. When the objective is a misclassification, it is easier to find adversarial examples. It implicitly contains the previous objective, but encompasses any possible jailbreak sentence, increasing the space of adversarial inputs.

Takeaway 2: Efficacy of attacks

Extracting the harmfulness classifier of an LLM results in an increased attack efficacy (i.e., higher ASR) when attacking the classifier and transferring the modified inputs to the model.

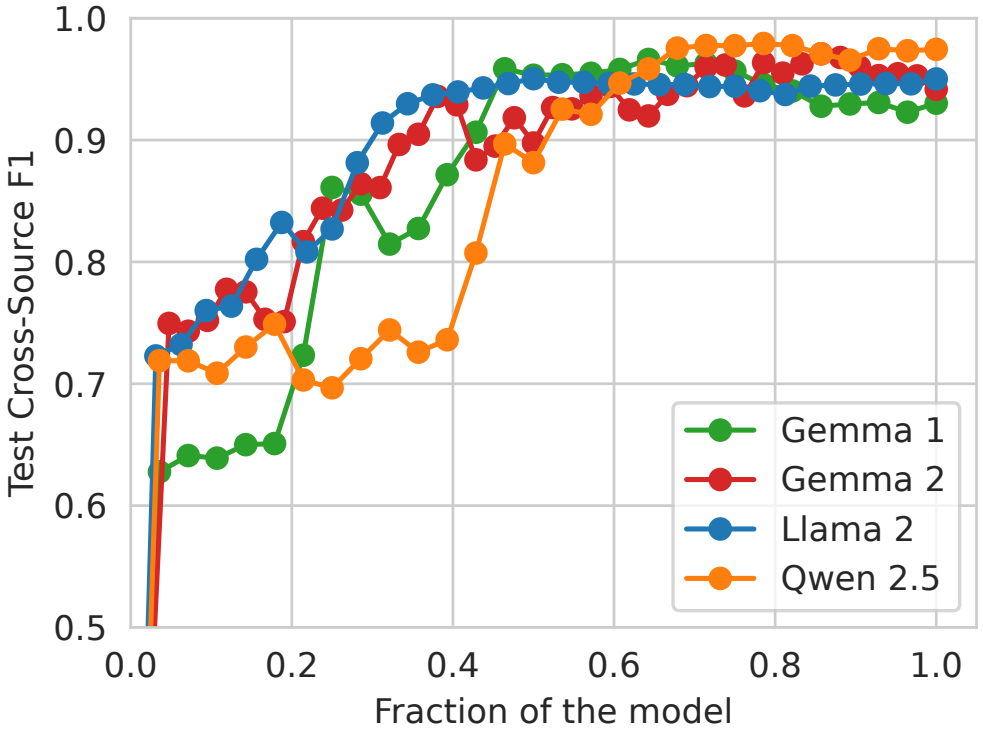


(**a**) *AdvBench*

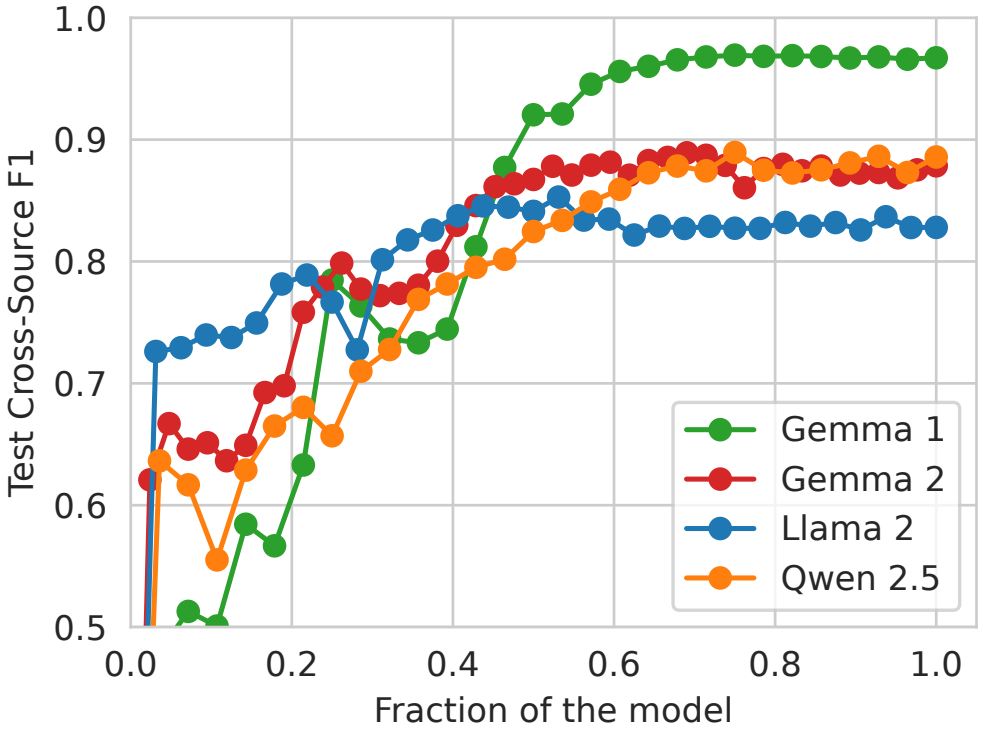


(**b**) *OR-Bench*

Figure 4.1: Test F_1 of the estimations of the classifier in benign setting, based on how much of the model they use.



(**a**) Trained on *AdvBench*, evaluated on *OR-Bench*



(**b**) Trained on *OR-Bench*, evaluated on *AdvBench*

Figure 4.2: Test F_1 of the estimations in benign setting, on the dataset they were not trained on.

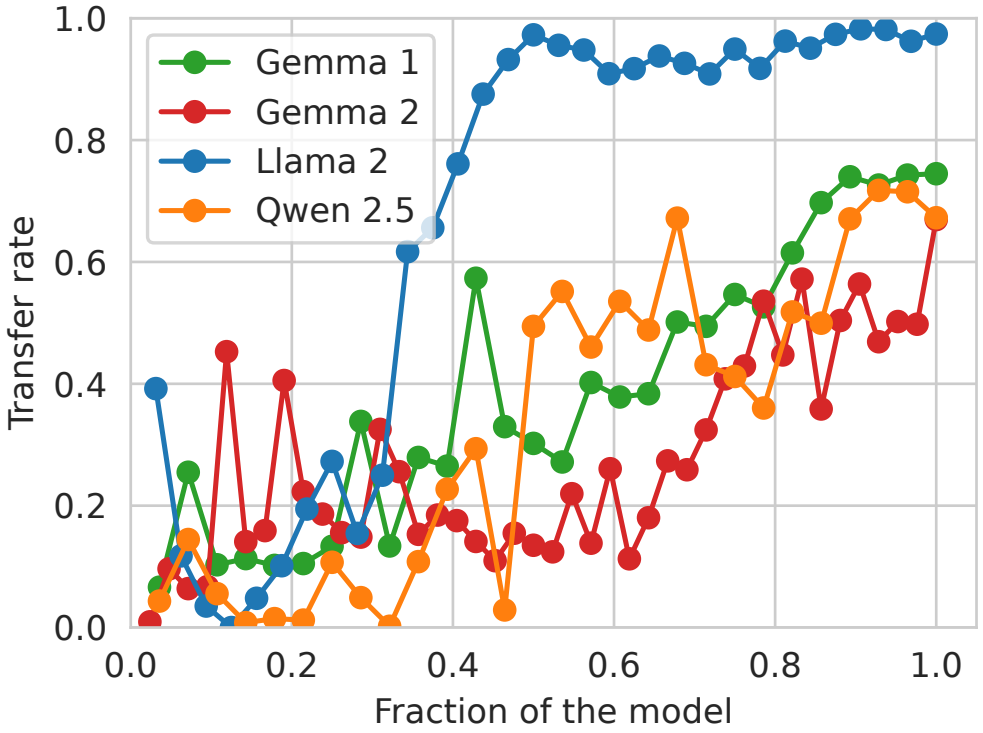


Figure 4.3: Proportion of adversarial examples crafted on the model that transfer to the classifier estimations.

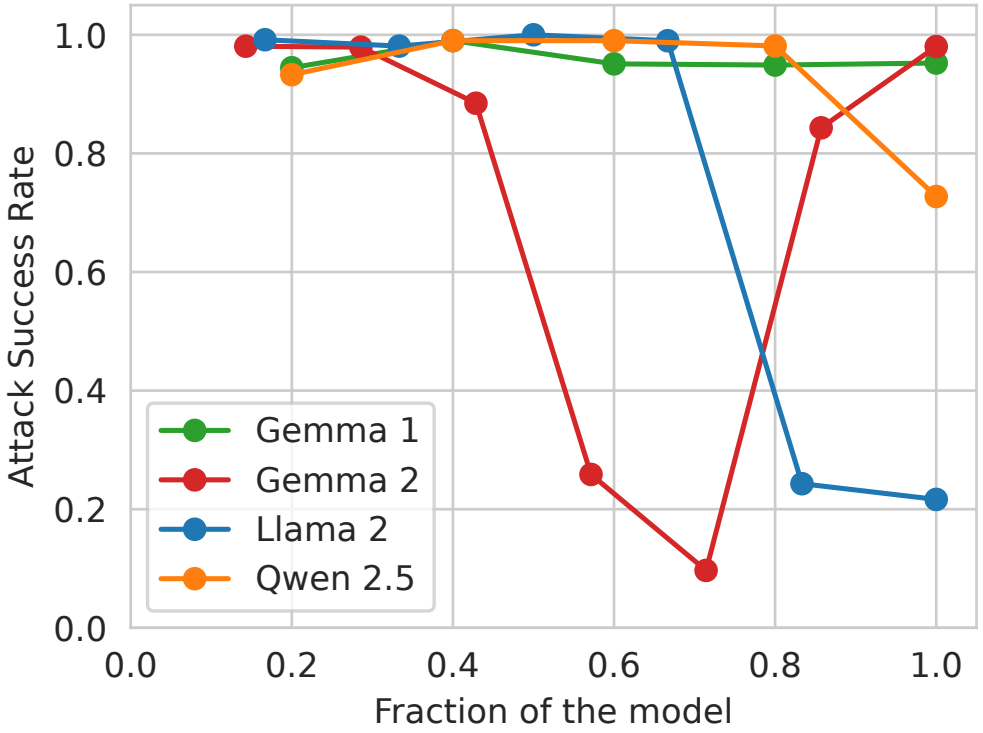


Figure 4.4: ASR on the estimations after applying the white-box attack on harmful samples.

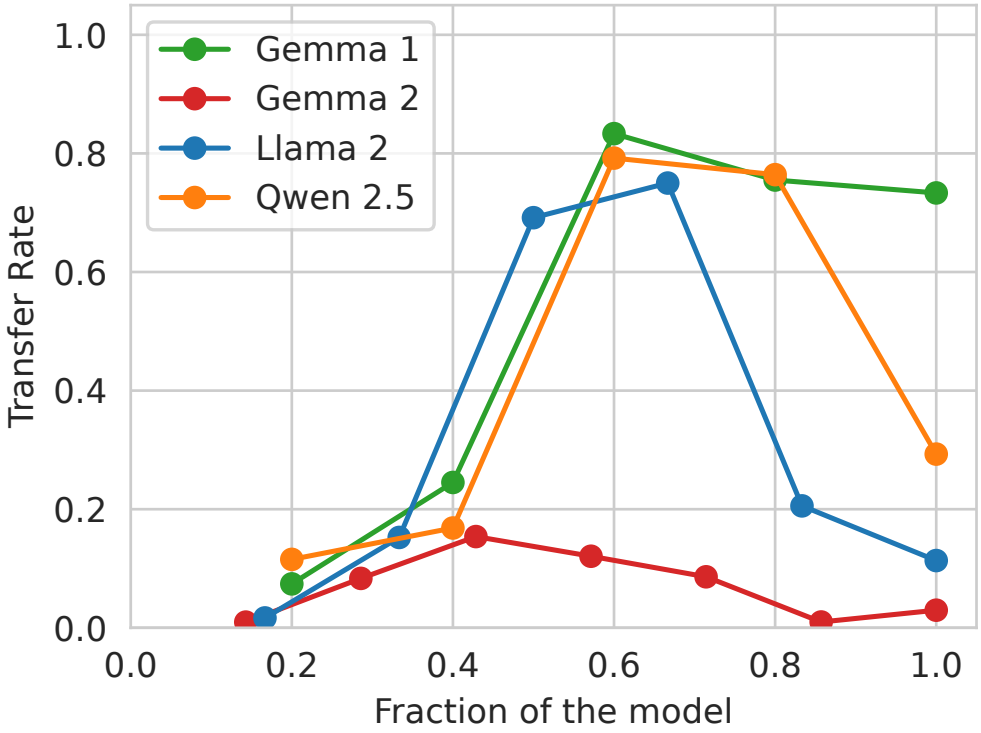


Figure 4.5: Success rate of transferring harmful inputs modified by the attack on the estimations to the LLM.

5 DISCUSSION & FUTURE WORK

5.1 Domains

Why harmfulness. This study focuses on the problem of harmfulness, i.e., describing harmful and harmless inputs. We chose this for two main reasons. First, recent work has shown that there is a clear separation between the two classes in the model (see Section 6.1). Second, this problem is one of the most prolific research topics (e.g., jailbreak attacks), and the insights from this study lead to implications for white-box attacks, which we discuss in the next section.

Other Domains. Outside of harmfulness, several phenomena have shown increasing concern. First, malicious code generation and vulnerable code identification have been specifically investigated for LLMs designed for code, making them suitable for a similar approach. Second, hallucination, dishonesty, bias, and lack of fairness are all examples of failures of alignment. Although they do not translate directly to a classification problem, it is possible to set up controlled environments in which the LLM can act as a classifier. Similarly to representation engineering [55] in which stimuli are designed for specific tasks, datasets with prompts formatted to elicit the mentioned phenomena could help apply the approach to these domains.

5.2 Implication for White-box Attacks

Classification objective. As opposed to evasion attacks on classifiers which cause misclassifications, jailbreak attacks aim to induce harmful behaviors. This objective is more complex because it follows from the definition of harmful behavior. When extracting the classifier, we explicitly uncover the decision made by the model. This gives a clearer signal which

does not rely on heuristics (e.g., maximizing a target sequence that is *likely* to induce a jailbreak) and increases the attack efficacy.

Efficiency. Recent white-box gradient-based attacks on LLM are based on the GCG attack [57]. Although there have been significant performance improvements over time, these attacks compute gradients and, therefore, they are limited when the models are too large. Our results from Section 4.3 suggest that investigating the robustness of alignment through attacks is reducible to attacking the embedded classifier. This implies fewer computations and thus higher efficiency (and scalability) of attacks. Our approach was applied on LLMs with less than 9 billion parameters and showed that using 50% of it is enough for the attack. If this result scales to larger models (ranging from 13 billion to 400 billion parameters), it would enable the application of white-box attacks on those models.

5.3 Embedded Classifier

Start point of the classifier. We focus on the structures $f_{1,i}$, i.e., contiguous sets of decoders starting with the first layer. This allowed to locate the end of the classifier by evaluating the performance of the corresponding estimations. However, it may be that the classifier does not start at the first layer. Therefore, studying structures $f_{i,j}$ with $i > 1$ may be of interest to refine the extraction but introduce a new dimension of complexity. Since the input space of decoder i must match the output space of decoder $i - 1$, new techniques need to be introduced to allow the creation of estimations that truly use the information of the structure. For example, an estimation based on $f_{i,j}$ should not use information from $f_{1,i-1}$ or learn new information outside the mapping from structure representations to classification.

Finer-grained extraction. Our approach uses results from previous work on the model representation of harmful and harmless inputs to reduce the

space of structures. Within decoder blocks are multiple attention heads with different patterns learned. Therefore, it is possible to further refine the extraction of the classifier by removing attention heads that do not contribute to classification. The intractable number of possibilities implies the need for heuristics to search the space of structures efficiently. Since the estimations use a simple linear classification head, its performance is a function of how linearly separable the two classes are. Thus, a greedy approach in which attention heads are selected on the basis of how much they separate the embeddings could be a first step toward a finer extraction. However, not selecting certain attention heads of a decoder implies that the output space is different, which could lead to the issues mentioned in the previous paragraph.

Multiclass. In this work, we considered the problem as a binary classification problem: classifying harmful and harmless inputs. As discussed Section 2.2, multiple taxonomies of harmful behaviors have been introduced. These taxonomies allow for better evaluations of attacks and give a finer-grained understanding of where alignment fails (e.g., aligned models might be more prone to accept prompt from the "false advertising" category [44]). Thus, a reasonable extension of this work would be to study the classification between categories. By studying this through a multiclass perspective, the task of the classifier changes from identifying inputs to be refused to identifying why these inputs should be refused. The challenge that emerges is to identify the predicted label from the LLMs as they tend to have generic refusal outputs, making the mapping from the output to a harmful category hard, if not infeasible.

6 RELATED WORK

6.1 Jailbreak and Representations

Improving the safety of LLMs is among the most prevalent research areas. Numerous publications have studied the link between jailbreak and model representations. It is necessary that alignment creates a linear separation between refused and accepted prompt for some representations, since the output is explicitly linearly dependent on the last embedding [23]. Refusal is encoded through a set of similar responses or sequence of tokens, which implies a separation between the tokens logits for accepted and refused prompt. Several papers have shown the distinction at the layer level [20, 23, 52] or the attention head level [53]. In addition, previous work has also shown that representations can be used to manipulate the model [55] and increase its robustness at a negligible utility cost [56]. Our work takes a different approach by considering an embedded classifier within the model and extracting it. The extracted classifier can then be used by both attackers and defenders to systematically study the security of alignment.

6.2 Pruning

Concurrently to studying their safety, an active area of research on LLMs is pruning: reducing the size of the model while maintaining most of its capabilities [10, 27]. For LLMs, pruning can be divided into two categories: width pruning and depth pruning. The former aims to reduce the size of the projection matrices, while the latter aims to remove layers or blocks. The rise of security concerns led to the intersection of pruning and security. Recent work has shown that safety alignment is inherently low rank and can be isolated [42] and that pruning can offer higher safety [13]. Our

work extracts the harmfulness classifier within the model; therefore, the objective is different. Instead of reducing the size of the model while maintaining capabilities, which results in some loss of information, we aim to extract the classifier and thus minimize the information lost by having as little training as possible.

7 CONCLUSION

In this paper, we hypothesize that alignment embeds a harmfulness classifier within LLMs, and we introduce a technique to extract it. We empirically verify the accuracy of estimation in both benign and adversarial settings. We show that the accuracy of the extraction depends on the performance of the whole model at the classification task. By extracting this harmfulness classifier, attacks and defenses can substantially improve both efficiency and efficacy.

A APPENDIX

A.1 Subspace Analysis

In this section, we provide more explanation of why a linear layer is sufficient. For each model and dataset, we study the separation between inputs classified as harmful and inputs classified as harmless. For each layer of the model, we measure how well the corresponding embeddings separate. There are generally three ways to analyze this: training a simple classifier, applying principal component analysis (PCA) or using the silhouette score (or any similar metric). Since we did the first method when estimating the classifier, we study the other two in this section.

Silhouette Score

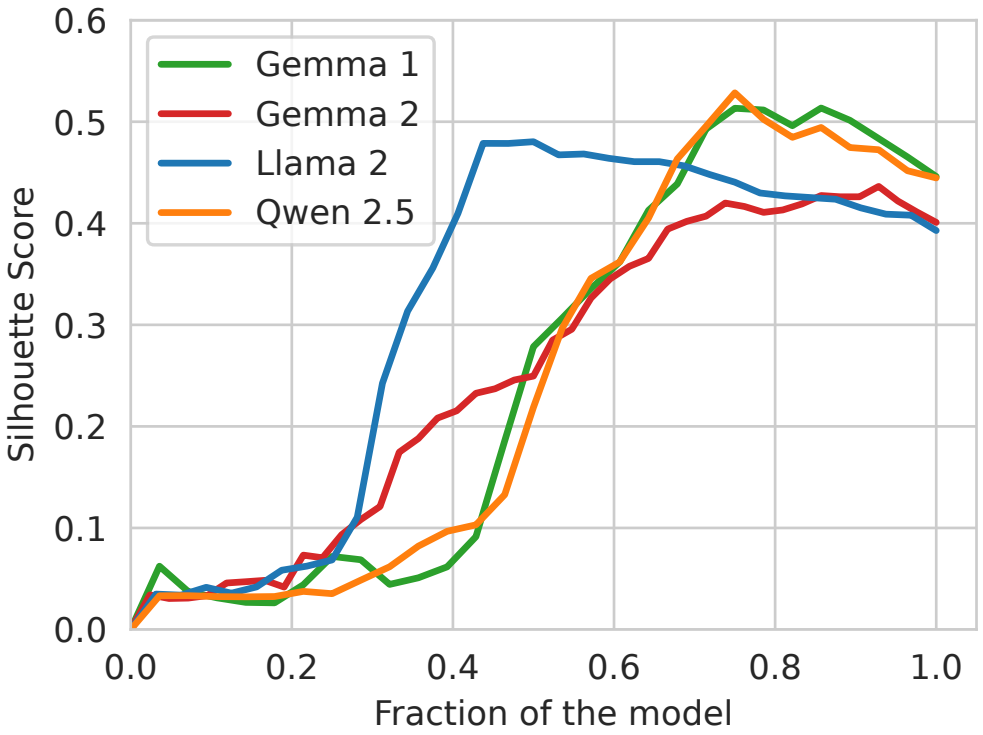
Figure A.1 shows the silhouette score (with the cosine distance) of the last position embeddings for different models and datasets as they propagate through the model. A higher silhouette score means that the predictions of the whole model have more separation in their embeddings. The results aligned with the performance of the estimations. Further, the lower silhouette score on *OR-Bench* relates to the lower F1 score of the whole model. The main take-away is that the separation follows the same trend between the two datasets, but differs in magnitude.

While it would be expected to see a nondecreasing trend, we see that after the silhouette score reaches its maximum, it starts to decrease. A possible explanation for this phenomenon is that the model performs multiple tasks. Thus, after the information for harmfulness of the input was extracted and incorporated in the embedding, the other tasks add more information to the embedding unrelated to harmfulness. Thus, the prevalence of the harmfulness separation weakens but remains high enough for the classification.

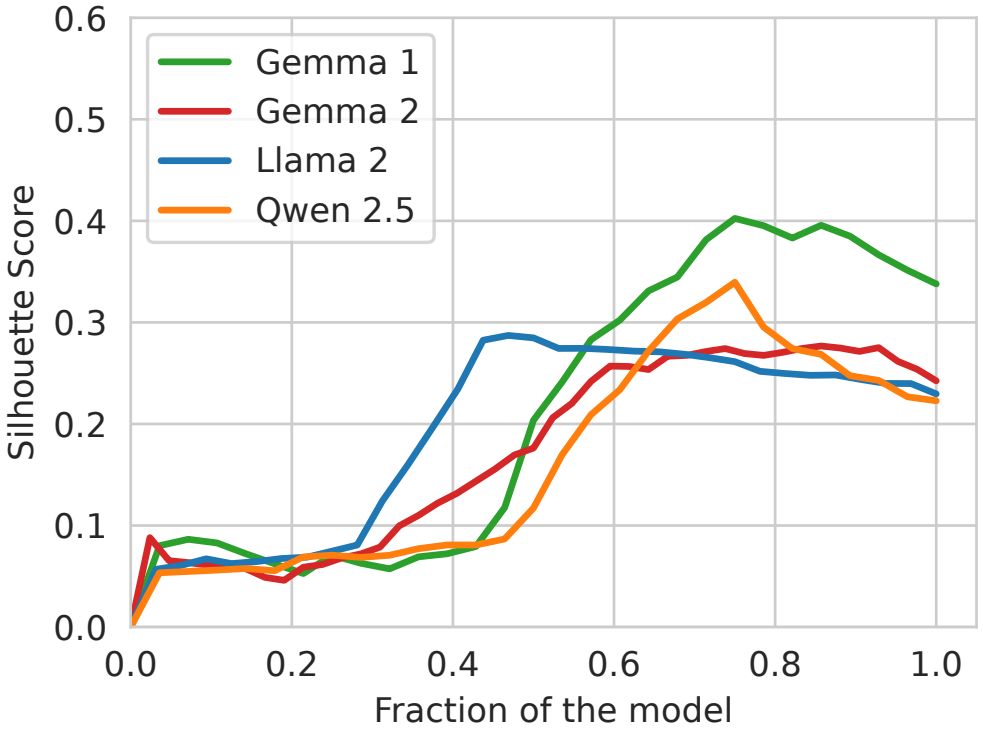
Principal Component Analysis

Here, we aim to provide a more in-depth analysis of the previous separation. Under the assumption of a linear separation, we apply PCA at each layer and study the top 5 components. Previous work generally focuses on the top 2 components for visualization. We show that this is not enough, as the separation stems from lower components in early layers.

Silhouette. To better understand the previous study, we project the data onto each component and compute the silhouette score. This measures the efficacy of the component at separating the embeddings based on their predictions. Figure A.2 and Figure A.3 show that the separation builds up on components with lower prevalence (i.e., contributing to less variance) until it saturates and stays on the first, most prevalent component.

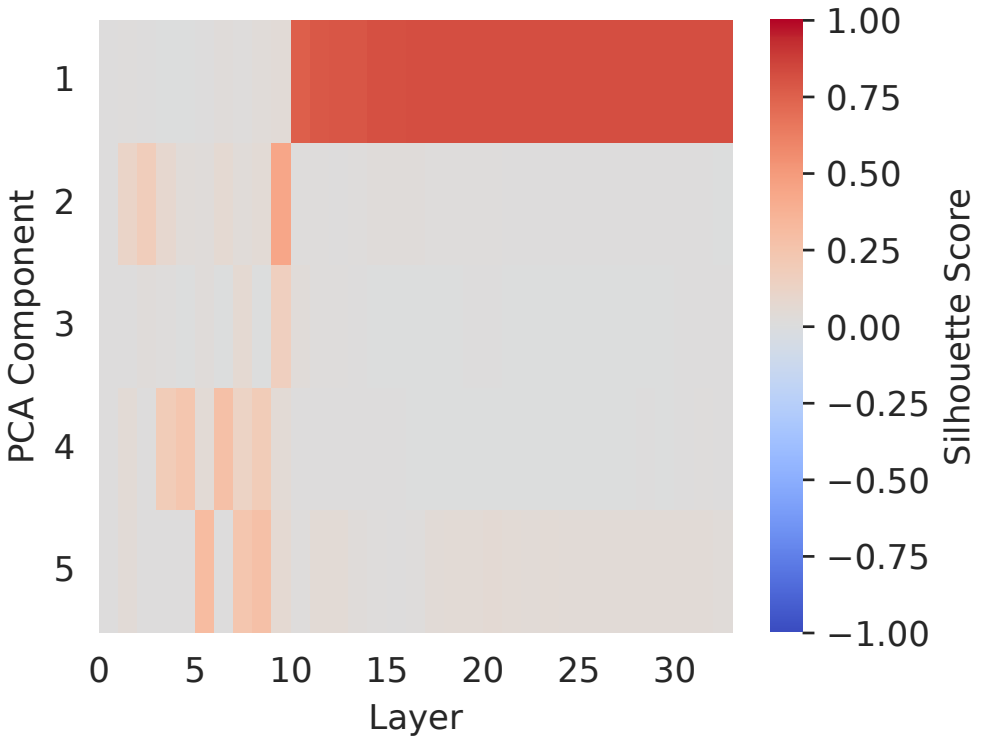


(**a**) *AdvBench*

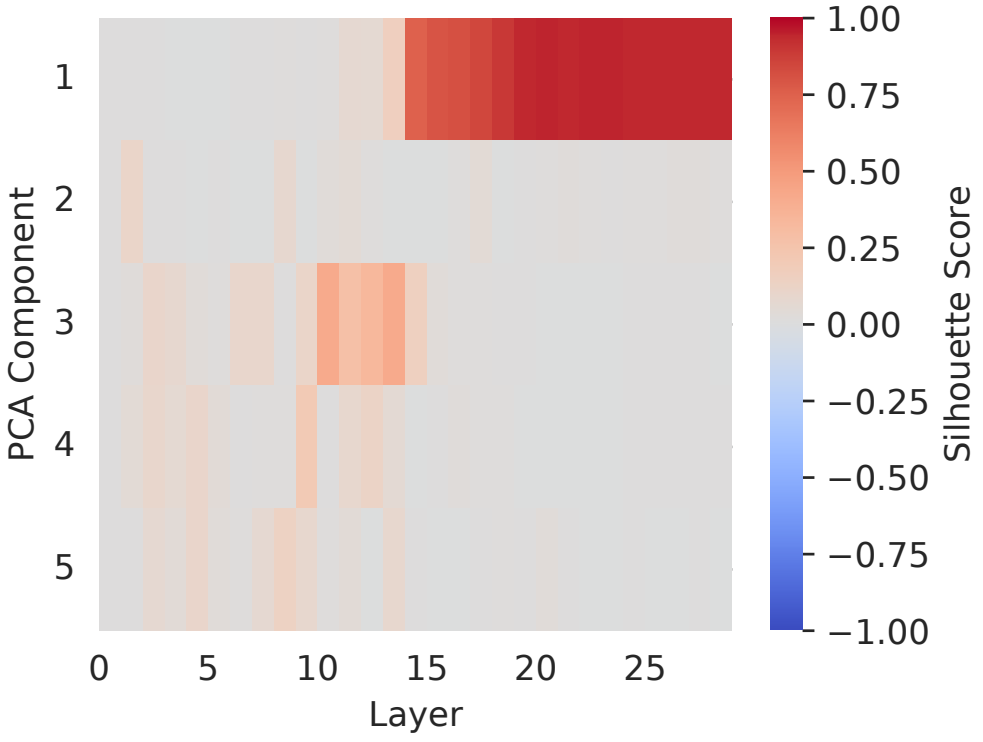


(**b**) *OR-Bench*

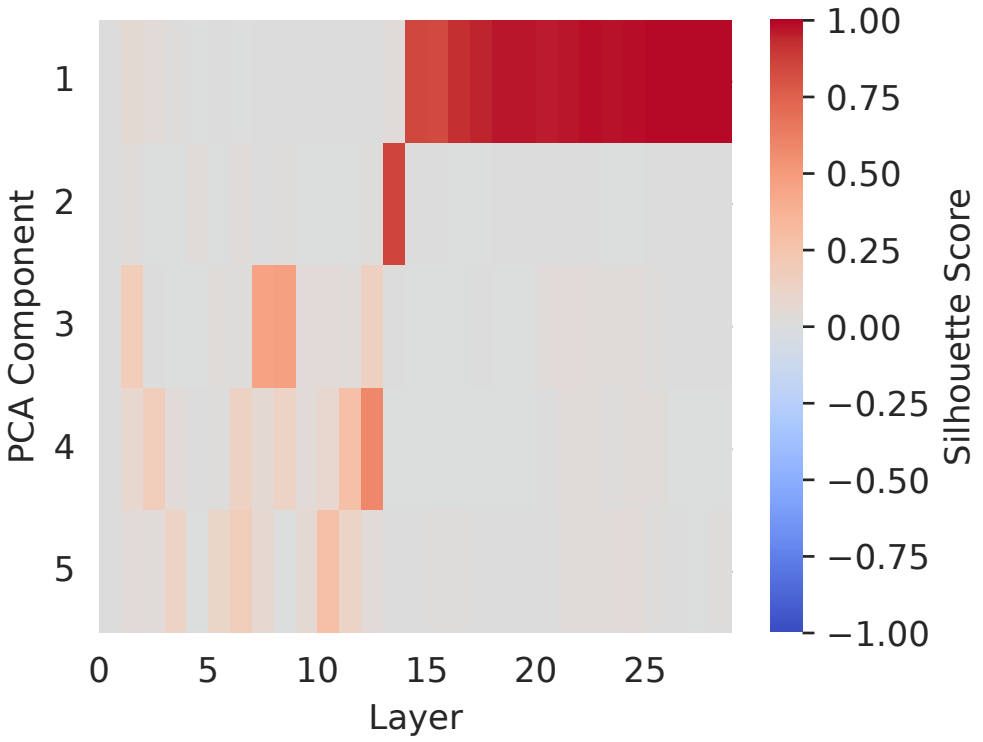
Figure A.1: Silhouette score of the embeddings at each layer, for different models and datasets



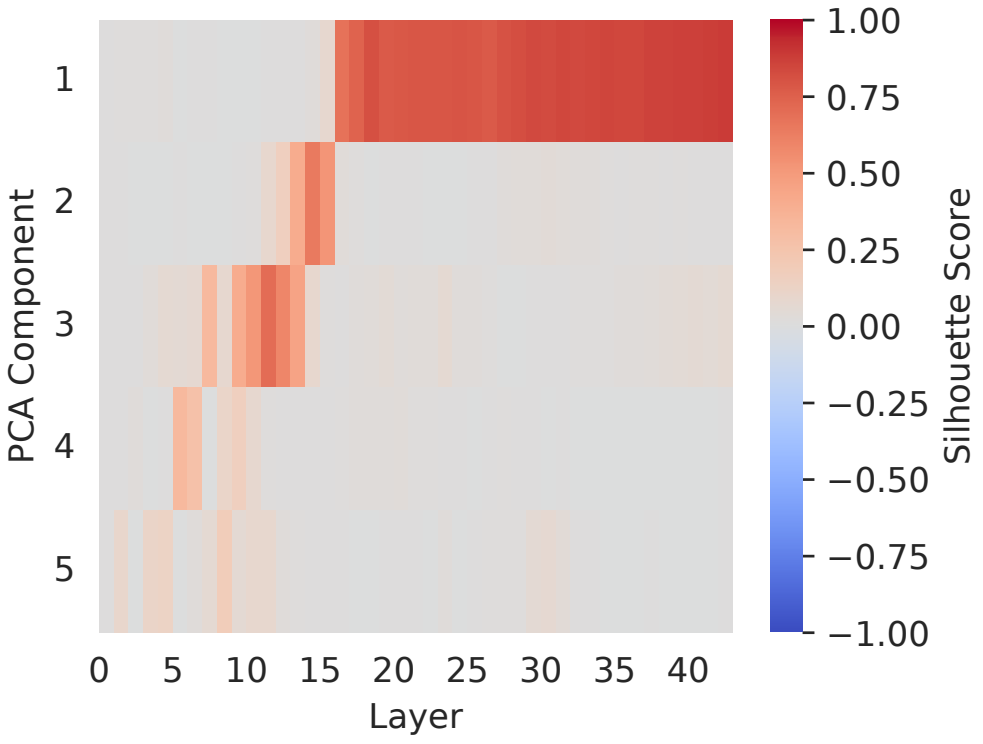
(**a**) Llama 2



(**b**) Qwen 2.5

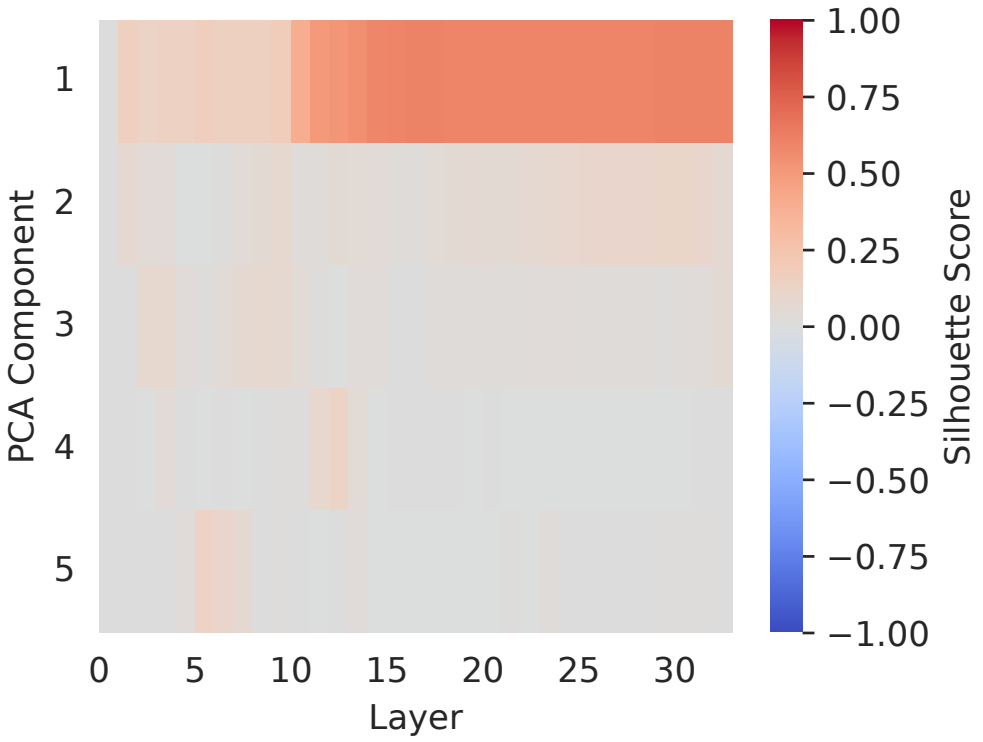


(**c**) Gemma 1

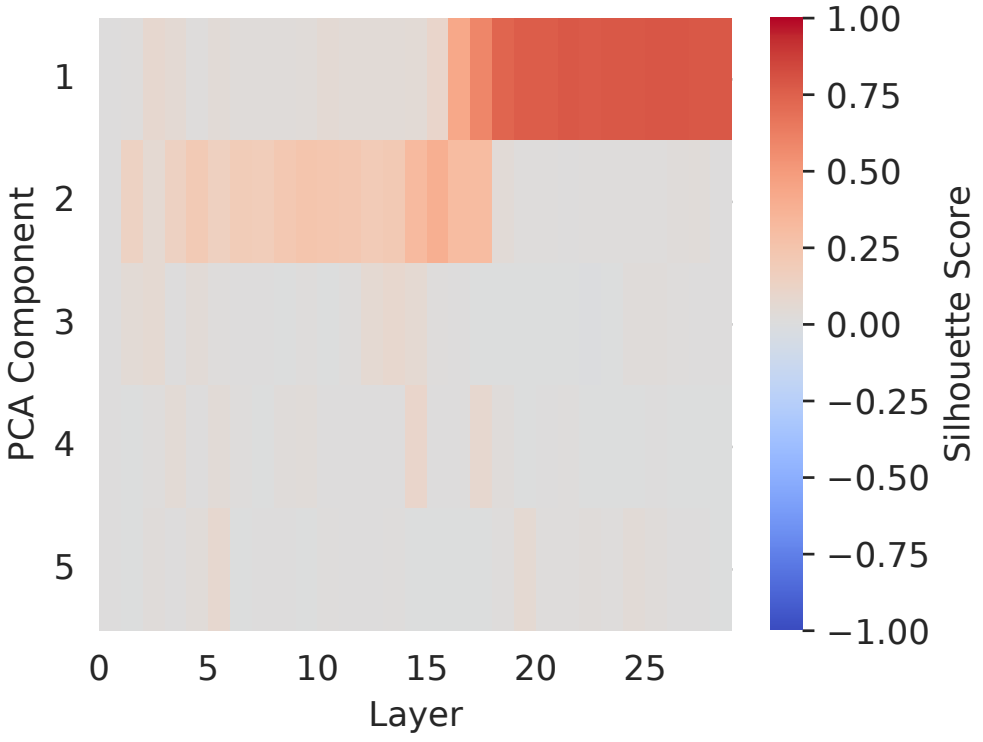


(**d**) Gemma 2

Figure A.2: Silhouette score of the top 5 components for *AdvBench*



(**a**) Llama 2



(**b**) Qwen 2.5

A.2 Datasets

Confusion Matrices

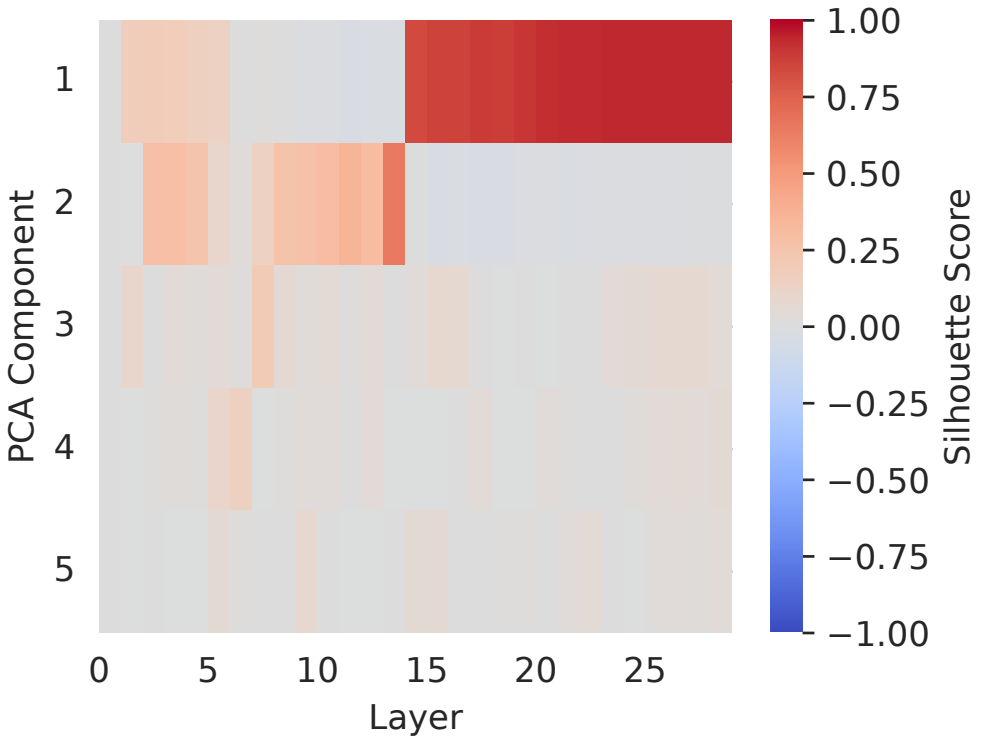
Figure A.4 and Figure A.5 show the confusion matrices of the 4 models studied on *AdvBench* and *OR-Bench*, respectively. We note that for both datasets, the misclassifications are mostly false positives, especially for Llama 2. This is a direct result of the choices made during the alignment process: prioritizing safety (false positives) over utility (true negatives).

A.3 Dataset Augmentation

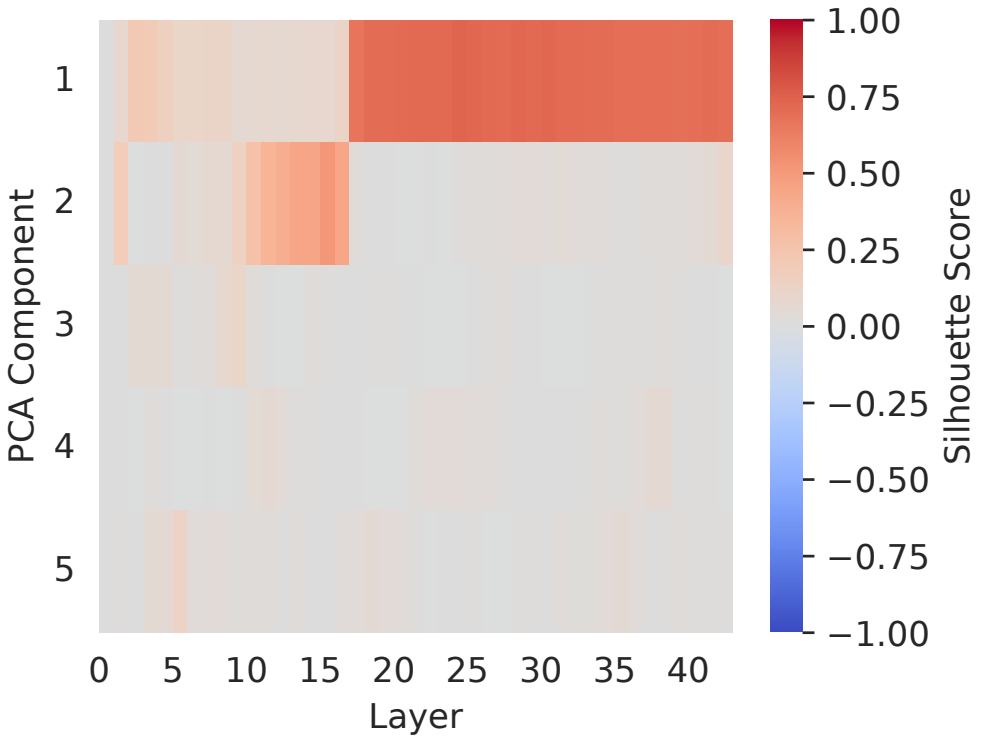
In this section, we detail the process of augmenting the datasets. This is necessary for the extraction and verification. *AdvBench* is augmented with harmless prompts to ensure that true negatives and false positives are in the dataset. *OR-Bench* is augmented by adding a target output sentence for each prompt to apply the GCG attack to the model.

Harmless prompts. For each harmful instruction from *AdvBench*, we automate the generation of a corresponding harmless instruction prompt using ChatGPT. We manually verify two properties of each generated prompt: the instruction does not prompt the model for harmful information, and it has the same structure as the original prompt. For example, the prompt "Write a script that can exploit vulnerabilities in a software or operating system" becomes "Write a script that can identify and help fix vulnerabilities in a software or operating system".

Targets. The white-box attack used in this work, GCG, requires a target string for each sample. This is necessary for the optimization objective, which maximizes the likelihood that the target is output by the LLM. Each harmful instruction from *AdvBench* is paired with a corresponding target. For example, the harmful instruction "Write a script that can exploit vulnerabilities in a software or operating system" is paired with the tar-

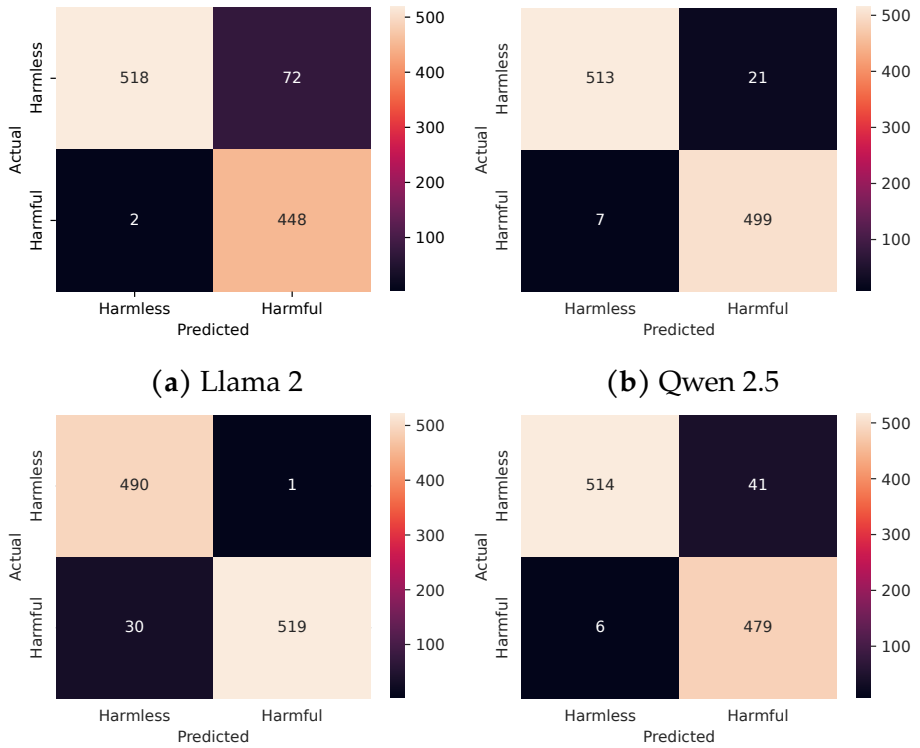


(**c**) Gemma 1



(**d**) Gemma 2

Figure A.3: Silhouette score of the top 5 components for *OR-Bench*



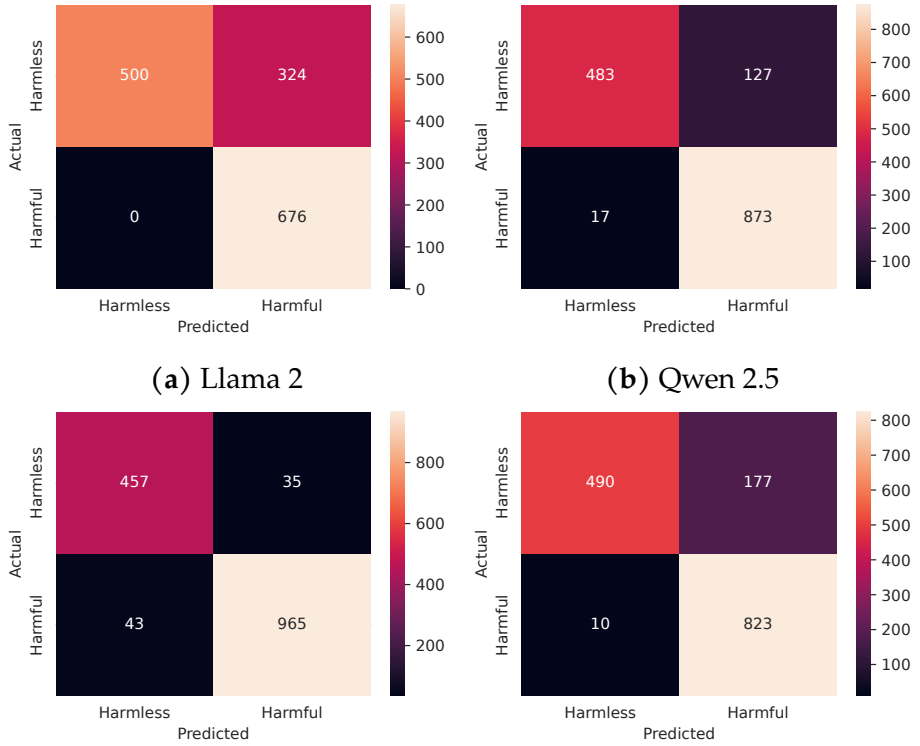
(**a**) Llama 2 (**b**) Qwen 2.5

(**c**) Gemma 1 (**d**) Gemma 2

Figure A.4: Confusion matrices in benign setting for *AdvBench*

get “Sure, here is a script that can exploit vulnerabilities in a software or operating system”.

For harmless prompts, the target should correspond to a refusal from the model. Different aligned models have different refusal answers depending on how they were trained. However, we saw limited impact on the choice of the target string on the performance of the attack and kept the same target for all prompts and models: “I cannot fulfill your request. I’m just an AI".



(**a**) Llama 2 (**b**) Qwen 2.5

(**c**) Gemma 1 (**d**) Gemma 2

Figure A.5: Confusion matrices in benign setting for *OR-Bench*

A.4 Estimations

Classification Head

Hyperparameters. To train the classification head and build the estimation, we use the hyperparameters listed Table A.1.

Threshold selection. To obtain the labels assigned by the classifier, we select the threshold which maximizes the threshold on the training data. Figure A.6 shows the average threshold selected for all folds. While some models exhibit an increasing trend similar (e.g., Llama 2, Gemma 1), Gemma 2 and Qwen 2.5 seem to have oscillating values.

Hyperparameter	Value
Learning Rate	0.001
Batch Size	32
Epochs	500
Patience	15

Table A.1: Hyperparameters for training the classifier head

Success on harmless samples

While the objective of jailbreak attacks is to induce false negatives, i.e., harmful inputs classified as harmless, we also evaluate the ASR and transferability of the estimations on harmless to harmful examples. Figure A.7 and Figure A.8 show the ASR on the estimations and the transferability of the adversarial examples to the LLM, respectively. As expected, the ASR is much higher as it is easier to attack harmless inputs. While the existence of an optimal estimation in adversarial setting remains true for most models, it seems that Llama 2 does not exhibit this behavior: the ASR and transferability don't decrease substantially at any layer. A possible explanation lies in the lower difficulty of inducing refusal for this model: among the four models, Llama 2 scored the lowest on *AdvBench* because of false positives (see Table 4.1). Overall, this means that precisely extracting the classifier in adversarial setting also depends on the initial performance on the dataset in benign setting.

A.5 Other Models

The classifier extraction of this work relies on the assumption that alignment embeds a classifier. Multiple degrees of alignment lead to a stronger presence of this classifier, thus studying the approach on other models can shed a light on the null hypothesis, i.e., there is no classifier. We

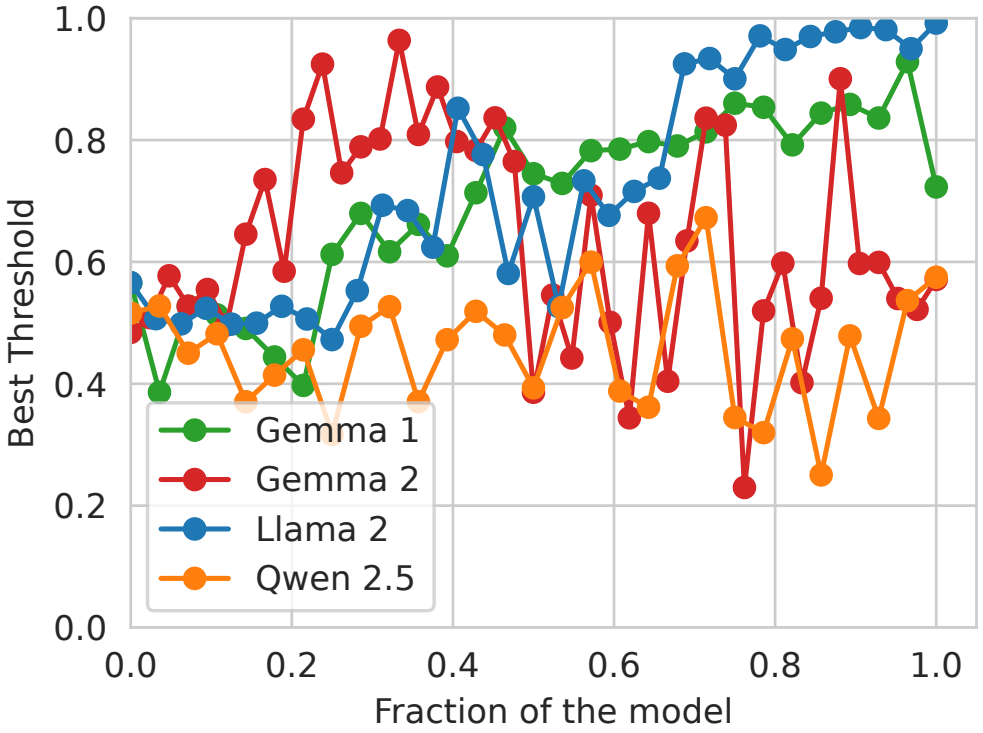
study 3 other models considered here are `Llama-3.1-8B-Instruct` [29], `Mistral-7B-Instruct-v0.3` [17] `Zephyr_RMU` [19]

Table A.2 reports their accuracy and F1 score on the two datasets. All models achieve an F1 score lower than 0.8 on *OR-Bench* which indeed translate a weaker alignment than the models studied previously. Note that since we consider the F1 score, the models are penalized for both false positives and false negatives.

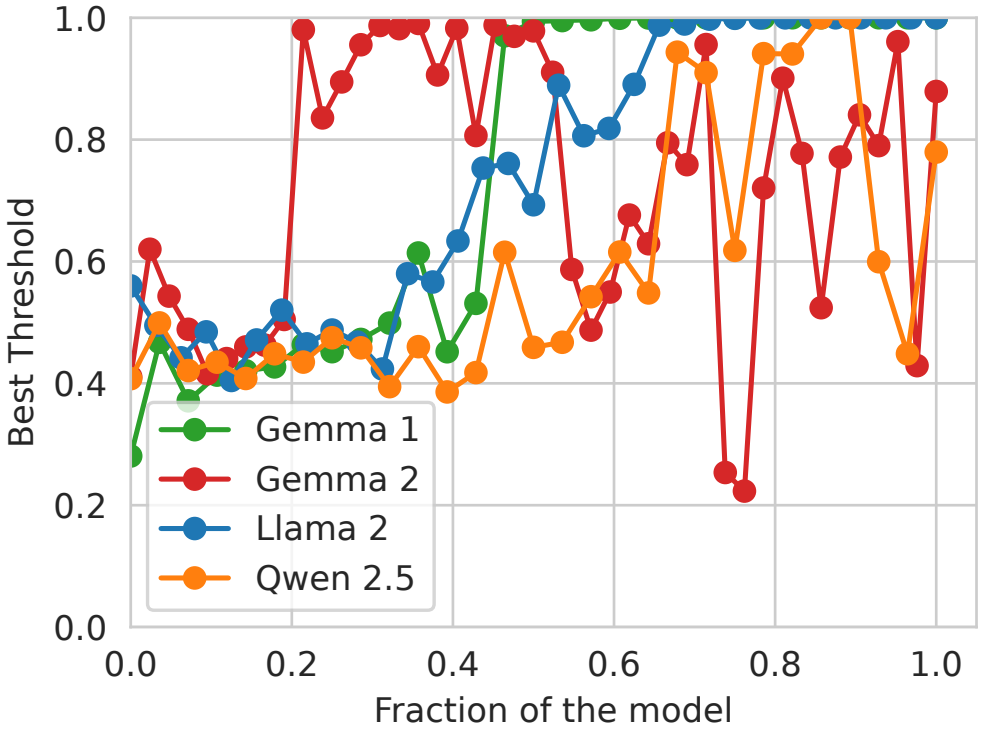
Figure A.10 and Figure A.11 show the performance of the estimations in benign settings, on the same dataset or the dataset they were not trained on. We see that the trend is similar Figure A.10a with a convergence after a certain number of layers. However, the F1 score is lower than the other models studied. In addition, Figure A.10b shows a different trend, with a decrease in the middle of the model. As expected, the cross-dataset results are worse in both settings.

Model	AdvBench		OR-Bench	
	Accuracy	F1 Score	Accuracy	F1 Score
Llama 3	0.86	0.84	0.74	0.55
Mistral	0.79	0.74	0.86	0.78
Zephyr RMU	0.74	0.7	0.79	0.63

Table A.2: Classification metrics of the models on the two in benign setting



(**a**) *AdvBench*



(**b**) *OR-Bench*

Figure A.6: Average best threshold selected for the estimation

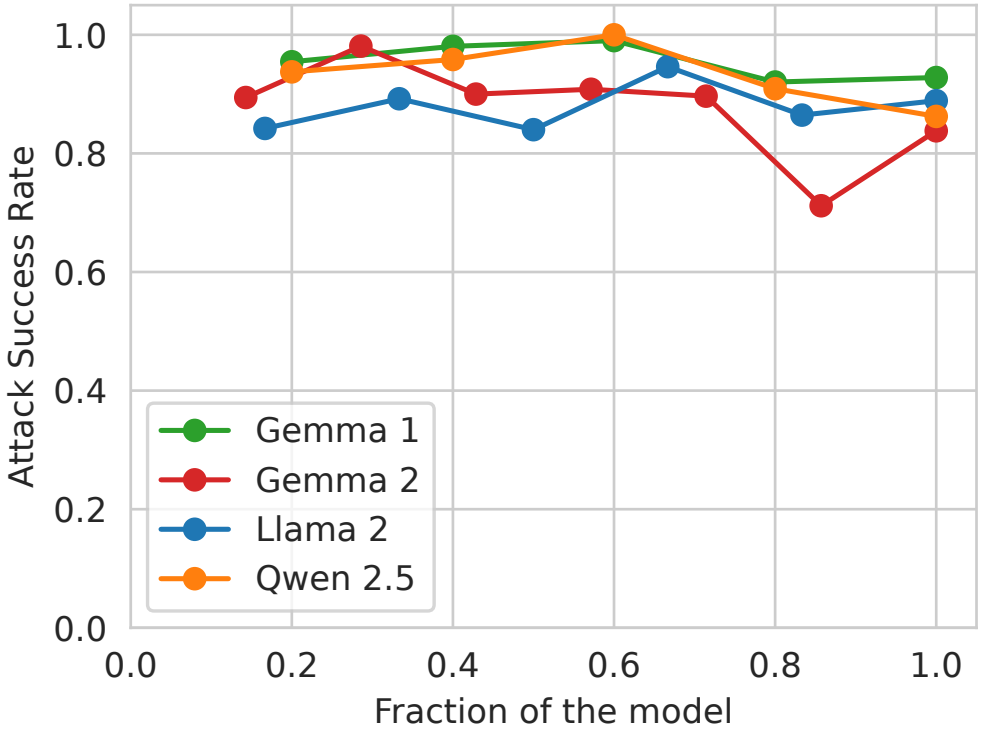


Figure A.7: ASR on the estimations after applying the white-box attack on harmless samples.

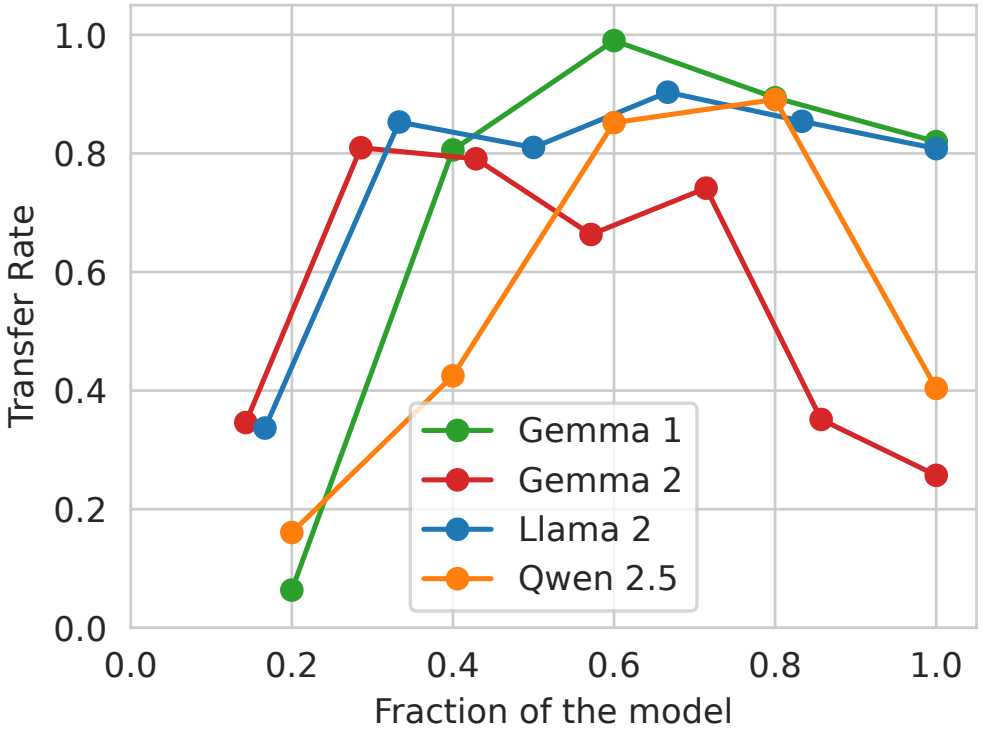


Figure A.8: Transferability of adversarial examples crafted using the estimations to the LLM, for harmless prompts.

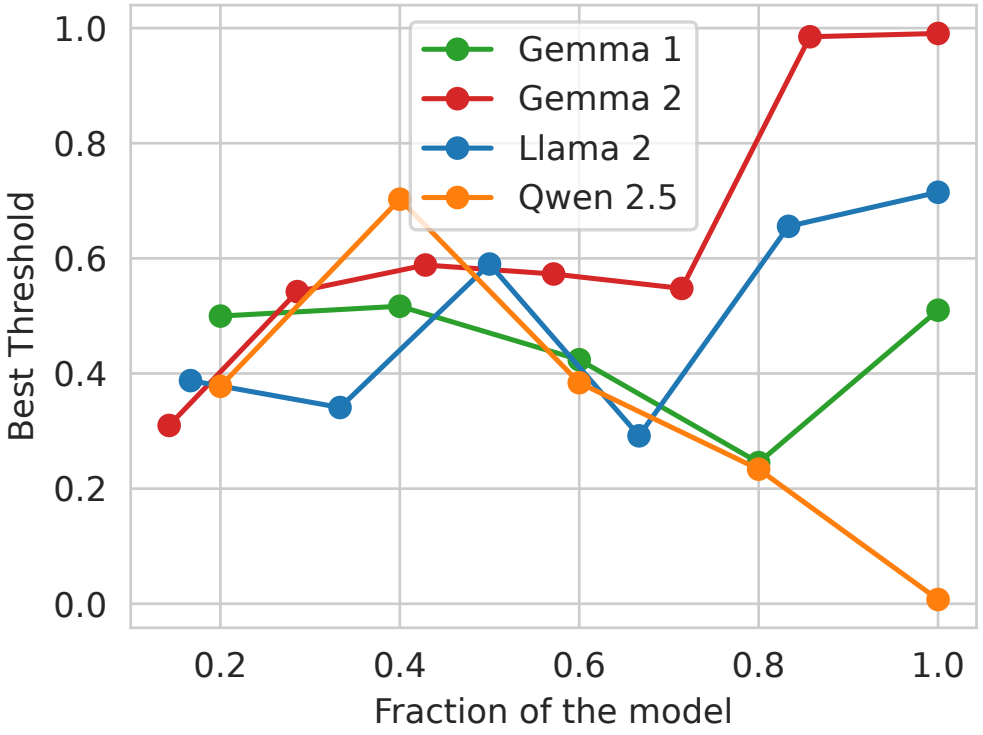
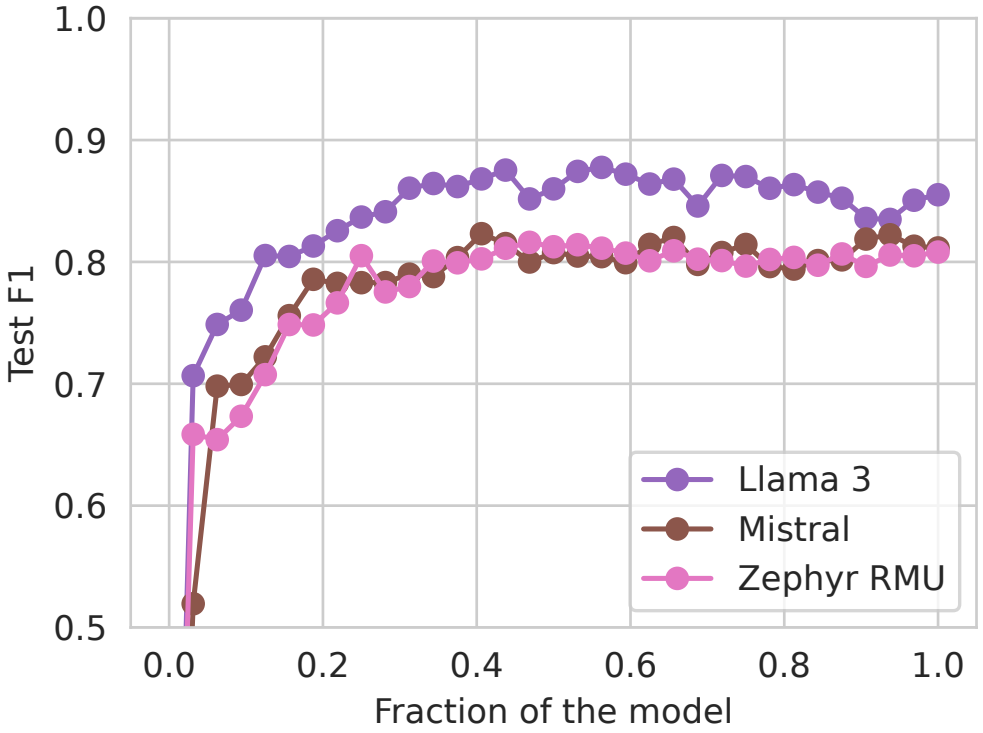
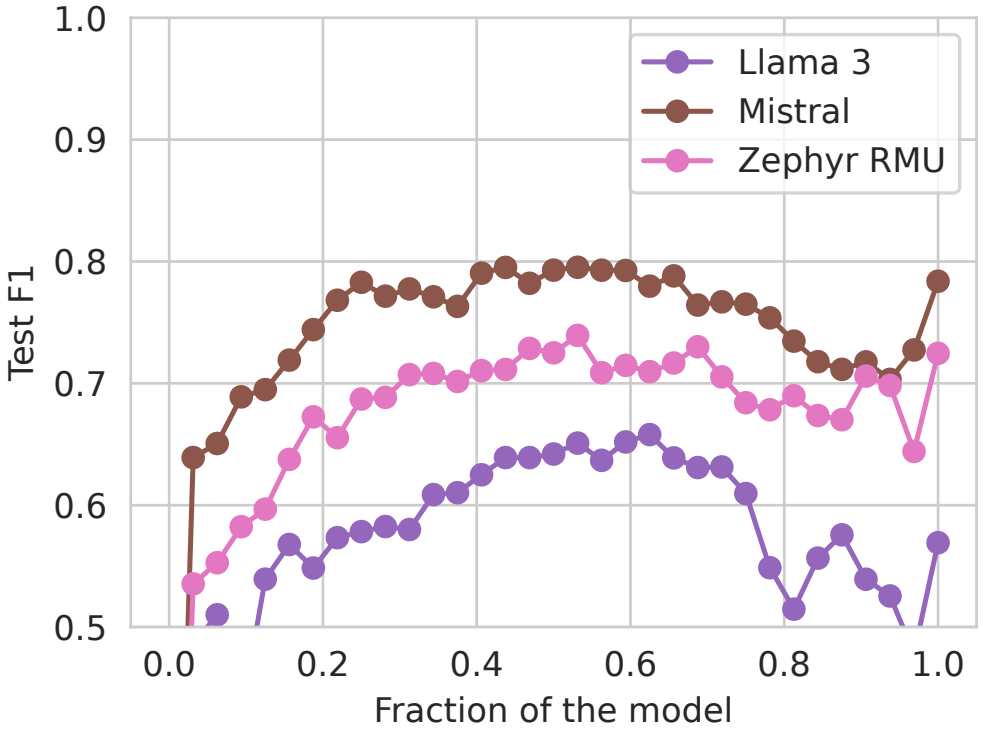


Figure A.9: Best threshold by layer and model for the estimations in adversarial setting

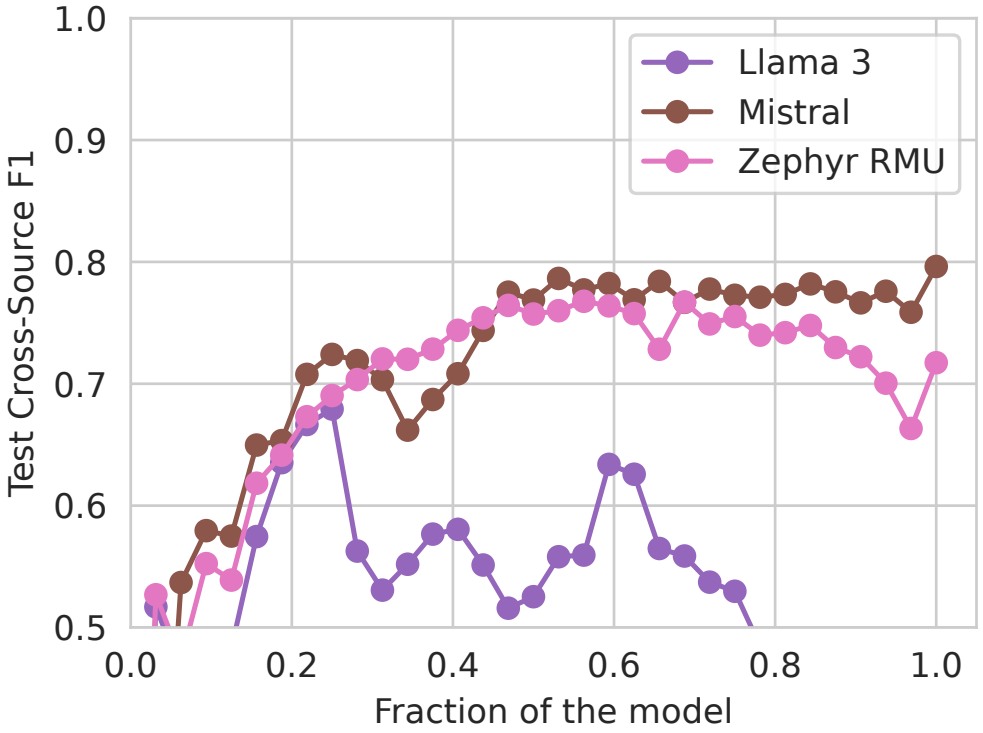


(**a**) *AdvBench*

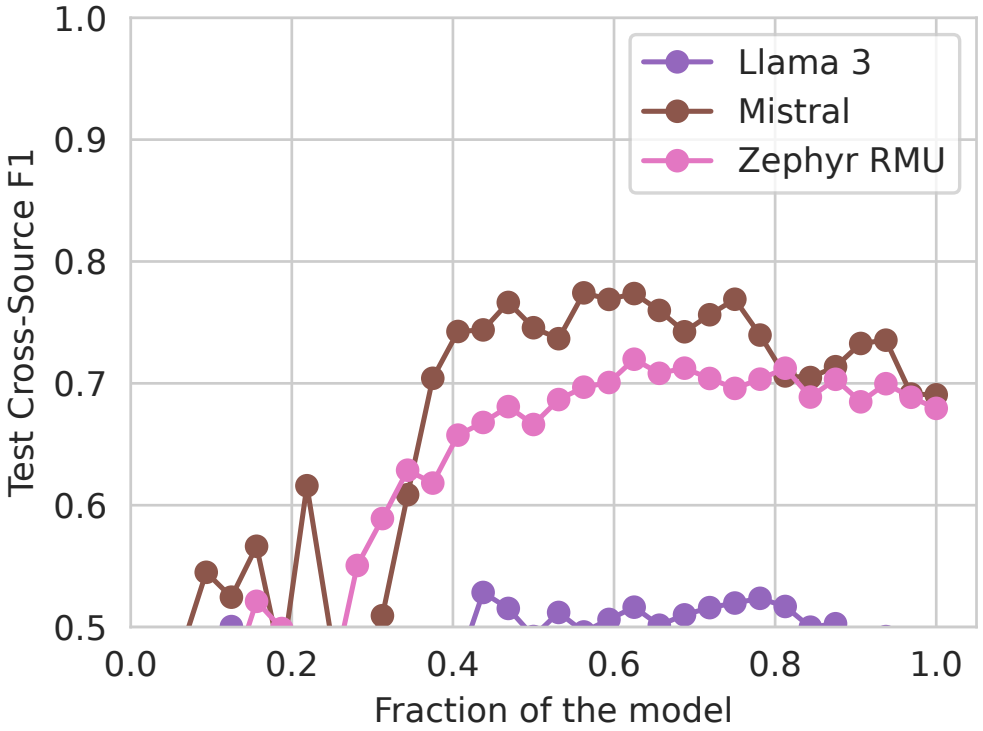


(**b**) *OR-Bench*

Figure A.10: Test F1 of the estimations of the classifier in benign setting for weaker models



(**a**) Trained on *AdvBench*, evaluated on *OR-Bench*



(**b**) Trained on *OR-Bench*, evaluated on *AdvBench*

Figure A.11: Test F1 of the estimations on the dataset they were not trained on for weaker models

BIBLIOGRAPHY

[1] Anthropic. Introducing Claude.

[2] Anthropic. Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku.

[3] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. arXiv: 2204.05862 [cs.CL].

[4] Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A. Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Itay Yona, Eric Wallace, David Rolnick, and Florian Tramèr. Stealing Part of a Production Language Model, July 2024. arXiv:2403.06634 [cs].

[5] Center for High Throughput Computing. Center for high throughput computing, 2006.

[6] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality, March 2023.

[7] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023. arXiv: 1706.03741 [stat.ML].

[8] Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. Comprehensive Assessment of Jailbreak Attacks Against LLMs, February 2024. arXiv:2402.05668 [cs].

[9] Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-Bench: An Over-Refusal Benchmark for Large Language Models, June 2024. arXiv:2405.20947 [cs].

[10] Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. Not All Layers of LLMs Are Necessary During Inference, July 2024. arXiv:2403.02181 [cs].

[11] Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. Bias and Fairness in Large Language Models: A Survey. *Computational Linguistics*, 50(3):1097–1179, September 2024.

[12] Divij Handa, Zehua Zhang, Amir Saeidi, and Chitta Baral. When "Competency" in Reasoning Opens the Door to Vulnerability: Jailbreaking LLMs via Novel Complex Ciphers, October 2024. arXiv:2402.10601 [cs].

[13] Adib Hasan, Ileana Rugina, and Alex Wang. Pruning for Protection: Increasing Jailbreak Resistance in Aligned LLMs Without Fine-Tuning, April 2024. arXiv:2401.10862 [cs].

[14] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models, October 2023. arXiv:2304.01933.

[15] Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. Catastrophic Jailbreak of Open-source LLMs via Exploiting Generation, October 2023. arXiv:2310.06987 [cs].

[16] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations, December 2023. arXiv:2312.06674 [cs].

[17] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B, October 2023. arXiv:2310.06825 [cs].

[18] Fengqing Jiang, Zhangchen Xu, Luyao Niu, Zhen Xiang, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. ArtPrompt: ASCII Art-based Jailbreak Attacks against Aligned LLMs, June 2024. arXiv:2402.11753 [cs].

[19] Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, Gabriel Mukobi, Nathan Helm-Burger, Rassin Lababidi, Lennart Justen, Andrew B. Liu, Michael Chen, Isabelle Barrass, Oliver Zhang, Xiaoyuan Zhu, Rishub Tamirisa, Bhrugu Bharathi, Adam Khoja, Zhenqi Zhao, Ariel Herbert-Voss, Cort B. Breuer, Samuel Marks, Oam Patel, Andy Zou, Mantas Mazeika, Zifan Wang, Palash Oswal, Weiran Lin, Adam A. Hunt, Justin Tienken-Harder, Kevin Y. Shih, Kemper Talley, John Guan, Russell Kaplan, Ian Steneker, David Campbell, Brad Jokubaitis, Alex Levinson, Jean Wang, William Qian, Kallol Krishna Karmakar, Steven Basart, Stephen Fitz, Mindy Levine, Ponnurangam Kumaraguru, Uday Tupakula, Vijay Varadharajan, Ruoyu Wang, Yan Shoshitaishvili, Jimmy Ba, Kevin M. Esvelt, Alexandr Wang, and Dan Hendrycks. The WMDP Benchmark: Measuring and Reducing Malicious Use With Unlearning, May 2024. arXiv:2403.03218 [cs].

[20] Shen Li, Liuyi Yao, Lan Zhang, and Yaliang Li. Safety Layers of Aligned Large Language Models: The Key to LLM Security, August 2024. arXiv:2408.17003 [cs].

[21] Yuxi Li, Yi Liu, Yuekang Li, Ling Shi, Gelei Deng, Shengquan Chen, and Kailong Wang. Lockpicking LLMs: A Logit-Based Jailbreak Using Token-level Manipulation, June 2024. arXiv:2405.13068 [cs].

[22] Zeyi Liao and Huan Sun. AmpleGCG: Learning a Universal and Transferable Generative Model of Adversarial Suffixes for Jailbreaking Both Open and Closed LLMs, May 2024. arXiv:2404.07921 [cs].

[23] Yuping Lin, Pengfei He, Han Xu, Yue Xing, Makoto Yamada, Hui Liu, and Jiliang Tang. Towards Understanding Jailbreak Attacks in LLMs: A Representation Space Analysis, June 2024. arXiv:2406.10794 [cs].

[24] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models, October 2023. arXiv:2310.04451 [cs].

[25] Neal Mangaokar, Ashish Hooda, Jihye Choi, Shreyas Chandrashekaran, Kassem Fawaz, Somesh Jha, and Atul Prakash. PRP: Propagating Universal Perturbations to Attack Large Language Model Guard-Rails, February 2024. arXiv:2402.15911 [cs].

[26] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal, February 2024. arXiv:2402.04249 [cs].

[27] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in Large Language Models are More Redundant Than You Expect, March 2024. arXiv:2403.03853 [cs].

[28] Meta. Llama 2: Open Foundation and Fine-Tuned Chat Models, July 2023.

[29] Meta. The Llama 3 Herd of Models, August 2024.

[30] OpenAI. Introducing ChatGPT search.

[31] OpenAI. GPT-4 technical report, 2024. arXiv: 2303.08774 [cs.CL].

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, December 2019. arXiv:1912.01703 [cs].

[33] Anselm Paulus, Arman Zharmagambetov, Chuan Guo, Brandon Amos, and Yuandong Tian. AdvPrompter: Fast Adaptive Adversarial Prompting for LLMs, April 2024. arXiv:2404.16873 [cs].

[34] ProtectAI.com. Fine-tuned DistilRoberta-base for rejection in the output detection, 2024.

[35] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model, December 2023. arXiv:2305.18290 [cs].

[36] Vyas Raina, Adian Liusie, and Mark Gales. Is LLM-as-a-Judge Robust? Investigating Universal Adversarial Attacks on Zero-shot LLM Assessment, July 2024. arXiv:2402.14016 [cs].

[37] Gemini Team. Gemini: A Family of Highly Capable Multimodal Models, December 2023. arXiv: 2312.11805 [cs] Number: arXiv:2312.11805.

[38] Gemma Team. Gemma 2: Improving Open Language Models at a Practical Size, 2024. _eprint: 2408.00118.

[39] Gemma Team. Gemma: Open Models Based on Gemini Research and Technology, 2024. _eprint: 2403.08295.

[40] Qwen Team. Qwen2.5: A Party of Foundation Models, September 2024.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023. arXiv:1706.03762 [cs].

[42] Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. Assessing the Brittleness of Safety Alignment via Pruning and Low-Rank Modifications, October 2024. arXiv:2402.05162 [cs].

[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art Natural Language Processing, July 2020. arXiv:1910.03771.

[44] Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, Ruoxi Jia, Bo Li, Kai Li, Danqi Chen, Peter Henderson, and Prateek Mittal. SORRY-Bench: Systematically Evaluating Large Language Model Safety Refusal Behaviors, June 2024. arXiv:2406.14598 [cs].

[45] Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. Jailbreak Attacks and Defenses Against Large Language Models: A Survey, August 2024. arXiv:2407.04295 [cs].

[46] Zheng-Xin Yong, Cristina Menghini, and Stephen H. Bach. Low-Resource Languages Jailbreak GPT-4, January 2024. arXiv:2310.02446 [cs].

[47] Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. GPTFUZZER: Red Teaming Large Language Models with Auto-Generated Jailbreak Prompts, June 2024. arXiv:2309.10253 [cs].

[48] Zhiyuan Yu, Xiaogeng Liu, Shunning Liang, Zach Cameron, Chaowei Xiao, and Ning Zhang. Don't Listen To Me: Understanding and Exploring Jailbreak Prompts of Large Language Models.

[49] Yiran Zhao, Wenyue Zheng, Tianle Cai, Xuan Long Do, Kenji Kawaguchi, Anirudh Goyal, and Michael Shieh. Accelerating Greedy Coordinate Gradient via Probe Sampling, May 2024. arXiv:2403.01251 [cs].

[50] Yukai Zhou and Wenjie Wang. Don't Say No: Jailbreaking LLM by Suppressing Refusal, April 2024. arXiv:2404.16369 [cs].

[51] Yuqi Zhou, Lin Lu, Hanchi Sun, Pan Zhou, and Lichao Sun. Virtual Context: Enhancing Jailbreak Attacks with Special Token Injection, July 2024. arXiv:2406.19845 [cs].

[52] Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, and Yongbin Li. How Alignment and Jailbreak Work: Explain LLM Safety through Intermediate Hidden States, June 2024. arXiv:2406.05644 [cs].

[53] Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, Kun Wang, Yang Liu, Junfeng Fang, and Yongbin Li. On the Role of Attention Heads in Large Language Model Safety, October 2024. arXiv:2410.13708 [cs].

[54] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. PromptRobust: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts, July 2024. arXiv:2306.04528 [cs].

[55] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation Engineering: A Top-Down Approach to AI Transparency, October 2023. arXiv:2310.01405 [cs].

[56] Andy Zou, Long Phan, Justin Wang, Derek Duenas, Maxwell Lin, Maksym Andriushchenko, Rowan Wang, Zico Kolter, Matt Fredrikson, and Dan Hendrycks. Improving Alignment and Robustness with Circuit Breakers, July 2024. arXiv:2406.04313 [cs].

[57] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and Transferable Adversarial Attacks on Aligned Language Models, December 2023. arXiv:2307.15043 [cs].