LLMs Under Attack: Comparing Adversarial Attacks on Large Language Models with Regularized Relaxation

Yuqing Liu Mahmoud Fakhry Hong Leng Toh
Oregon State University
{liuyuqi, fakhryk, tohh}@oregonstate.edu

Abstract

LLM is one of the main technologies humanity is using right now. It breaks through many problems that we have, and enables AI to be popular as it is today. However, it also introduced a new problem that we have to solve. We can learn any skills, no matter it is harmful or not. Since the AI model has no accountability, humans have to be responsible for that. AI providers have installed safety mechanisms to prevent the model from generating harmful results, like how to create a bomb, but this safety can be compromised by adversarial attack techniques. In this work, we try to replicate the results from the original RR paper, comparing two adversarial methods: PGD and RR with L2 regularization in terms of attack success rate (ASR). Additionally, we compare two additional adversarial methods: RR using L2+KL and BEAST. Our results showed that most of the results from the original paper are replicable.

1. Introduction

Recently, in the world of machine learning, it has been developing rapidly and has been utilized widely, especially in the natural language processing field (NLP). The emergence of ChatGPT in 2022 [1] has shaken the world since then. For instance, our daily life is likely to have an AI as one of the basic needs (especially for college students), and businesses have also adopted and embedded AI into many business operations. This tide of AI has brought humanity a lot of benefits. For example, we can focus more on the meaningful tasks and get rid of those repetitive/routine tasks that can be automated, or we can increase productivity with the same amount of time. However, it has also brought humanity threats. There are many articles pointing out problems regarding AI. For example, IBM wrote an article [2] about risks from AI and how to tackle these problems. One of the risks mentioned in the article is a lack of accountability. With AI chatbots, anyone can learn most of the skills out there with their fingertips, such as coding, gardening, etc. And people can also learn something that might be harmful, such as how to create a bomb or malware, which is a problem since AI lacks accountability. So, the model provider installed several safety mechanisms for blocking the model from generating harmful answers. However, the safety mechanism can be compromised by many adversarial attack techniques, such as Greedy Coordinate Gradient (GCG) [3] and Projected Gradient Descent (PGD) [4].

In our study, we compare the results from three main algorithms, including PGD, Regularized Relaxation (RR) [5], and Beam Search-based Adversarial Attack (BEAST) [6]. Additionally, we used AdvBench and HarmBench as our datasets, and used attack success rate (ASR) as the key evaluation metric.

2. Related Work

Adversarial attacks against neural language models have received increasing attention, especially with the rise of large-scale models such as GPT and LLaMA. Early works focused on token-level perturbations such as TextFooler [7] and HotFlip [8], where adversarial examples were crafted by substituting or inserting discrete tokens to alter model behavior. However, such approaches are limited by the discrete nature of language and often produce grammatically awkward sentences.

Projected Gradient Descent (PGD) [4] is a white-box attack method that repeatedly adds gradients to continuous vectors of model input, causing the model to output incorrect results. It is an iterative version of FGSM [9], which is stronger and more stable. The attacker starts from the embedding of the original input. In each iteration, the gradient of the loss function with respect to the embedding is calculated, moving in the direction of increasing loss. This process is repeated many times until the model output is successfully perturbed. PGD is simple and effective, especially when continuously optimizing in the embedding space. However, the disadvantage is that it is not easy to map the continuous perturbed embedding back to a reasonable token, and it is difficult to find a semantically reason-

able output.

Regularized Relaxation (RR) [5] method proposes to relax the discrete token space (such as a one-hot vector) into a continuous probability distribution so that it can be optimized with gradients; then use softmax and entropy loss regularization to keep the distribution close to one-hot. Each token is no longer a separate one-hot vector, but a trainable continuous distribution. Gradient optimization is performed on this distribution to make it closer to the target output. The entropy regularization term is then added to make the distribution closer to discrete tokens and prevent overfitting.

In contrast to RR and PGD, the Beam Search-based Adversarial Attack (BEAST) method [6] is a gradient-free adversarial prompt optimization algorithm that uses beam search and multinomial sampling to build its adversarial suffixes. This is beneficial when the target language model is a black box, such that gradient information is not available to the attacker, though access to logit outputs for token probabilities is necessary for the BEAST method. Another example of a gradient-free attack is BERT-ATTACK [11], which was previously used to find words that are similar in meaning to those in a text classification dataset yet mislead the model. Related to the top-k approach used by BEAST, the BERT-ATTACK uses the perplexity of subwords to select the most meaningful top-k replacement words, while BEAST uses a top-k filter to find adversarial suffixes. Other black-box attack strategies include the use of an LLM as an optimizer to generate jailbreaking prompts [12], such as PAIR [13]. Similar black-box approaches involving an optimized LLM include TAP [14] and AdvPrompter [15]. With AdvPrompter, similar to BEAST, its gradient-free training makes calling an attack faster and more memory efficient than gradient-based approaches [15].

3. Methodology

3.1. Projected Gradient Descent (PGD)

We formulate our adversarial suffix optimization as a white-box attack on a pretrained language model $f(\cdot)$ using Projected Gradient Descent (PGD). Given a fixed prompt represented by token embeddings $\mathbf{e}_x \in \mathbb{R}^{n \times d}$, we optimize learnable continuous suffix embeddings $\mathbf{e}_{\text{adv}} \in \mathbb{R}^{m \times d}$ to induce the model to output a target sequence y.

In each optimization step, we compute the gradient of the task loss $\mathcal{L}$ with respect to the adversarial embeddings and update them in the direction that minimizes the loss

$$\mathbf{e}_{\text{adv}} \leftarrow \mathbf{e}_{\text{adv}} + \alpha \cdot \nabla_{\mathbf{e}_{\text{adv}}} \mathcal{L}\left(f([\mathbf{e}_x \| \mathbf{e}_{\text{adv}}]), y\right) \tag{1}$$

To ensure that the final adversarial suffix can be mapped back to real tokens, we project the optimized embeddings onto the vocabulary embedding matrix via nearest-neighbor lookup after the final iteration. This approach follows the PGD paradigm: iteratively updating a continuous input, then projecting back to the feasible (discrete) set.

However, when PGD is used alone, it tends to produce embeddings that drift far from the original embedding space and cause large semantic shifts in model outputs. To address this limitation, we incorporate two regularization strategies that constrain the optimization: an L2 penalty on the suffix embeddings and a KL divergence penalty on the model's output distributions. These regularizations are described in sections 3.2 and 3.3.

3.2. Regularized Relaxation (RR) with L2

To ensure that the adversarial suffix embeddings remain within a plausible region of the embedding space, we apply an L2 regularization term directly on the learned embeddings $\mathbf{e}_{\text{adv}}$. This regularization precedes the addition of KL divergence and is applied at every optimization step.

The intuition behind this penalty is that embedding vectors optimized purely via task loss can drift far away from the distribution of valid token embeddings, especially early in training. Such vectors may not correspond to any real words, making the adversarial suffix semantically meaningless or harder to discretize. To address this, we penalize the distance between each adversarial embedding and the mean of all vocabulary embeddings

$$\mathcal{L}_{\text{L2}} = \frac{\lambda}{2} \sum_{i=1}^{m} \|\mathbf{e}_{\tilde{x}_i} - \bar{\mathbf{e}}\|^2 \tag{2}$$

where $\bar{\mathbf{e}} = \frac{1}{V} \sum_{j=1}^{V} \mathbf{e}_j$ is the average embedding across the vocabulary, and λ is a scalar hyperparameter controlling the regularization strength.

The L2 penalty serves two main purposes. First, it prevents the adversarial suffix from drifting too far in early iterations, stabilizing the initial optimization process. Second, by encouraging proximity to the embedding manifold, it makes later discretization (via nearest-neighbor projection) more reliable and results in suffixes that are more syntactically and semantically valid. Compared to regularization towards a fixed vector such as the origin, using the vocabulary mean provides a more data-aware center of attraction that better preserves linguistic structure.

The algorithm is provided below (see Algorithm 1)

3.3. Regularized Relaxation (RR) with L2 and KL

The original paper [5] experimented with L2 loss, but what about other loss functions? We extended a standard gradient-based adversarial embedding optimization method by incorporating two forms of regularization: an L2 penalty on the suffix token embeddings and a Kullback–Leibler (KL) divergence term on the model output distributions.

Let $\mathbf{e}_{\text{adv}}$ denote the learnable adversarial suffix token embeddings, and let $\mathbf{e}_{\text{x}}$ denote the fixed prompt embeddings.

Algorithm 1 Gradient Descent with L2 Regularization

Input: LLM $f_\theta(\cdot)$, prompt token embeddings $\mathbf{e}_\mathbf{x} \in \mathbb{R}^{n\times d}$, suffix token embeddings $\mathbf{e}_{\text{adv}} \in \mathbb{R}^{m\times d}$ as $(\mathbf{e}_{\tilde{x}_1}, \ldots, \mathbf{e}_{\tilde{x}_m})$, target token sequence $\mathbf{y}$, loss function $\mathcal{L}$

Parameters: learning rate $\alpha \in \mathbb{R}_{>0}$, regularization parameter $\lambda \in \mathbb{R}_{>0}$, number of iterations $T \in \mathbb{N}$

Initialize $\mathbf{e}_{\text{adv}}$
for $t \leftarrow 1$ to T **do**
  $\text{Loss}_t \leftarrow \mathcal{L}(\mathbf{e}_\mathbf{x} \| \mathbf{e}_{\text{adv}}, \mathbf{y}) + \frac{\lambda}{2} \sum_{i=1}^{m} \left\| \mathbf{e}_{\tilde{x}_i} - \frac{1}{V} \sum_{i=1}^{V} \mathbf{e}_i \right\|_2^2$
  $\mathbf{G}_t \leftarrow \nabla_{\mathbf{e}_{\text{adv}}} \text{Loss}_t$
  $\mathbf{e}_{\text{adv}} \leftarrow \mathbf{e}_{\text{adv}} + \alpha \mathbf{G}_t$
end for
// Discretize the optimized embeddings $\mathbf{e}_{\text{adv}}$
for $i \leftarrow 1$ **to** m **do**
  $\tilde{\mathbf{x}}_i \leftarrow \arg\min_{\mathbf{x}_j} \|\mathbf{e}_{\tilde{x}_i} - \mathbf{e}_{x_j}\|_2^2$ where $\mathbf{e}_{x_j} \in \mathbf{E}$
end for
Output: Adversarial suffix tokens $\tilde{\mathbf{x}}_{\text{adv}} = [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_m]$

The optimization objective is to minimize the task loss $\mathcal{L}$, typically the cross-entropy between model output and the target sequence, while enforcing that the adversarial embeddings remain close to the distribution of existing tokens. To this end, we apply an L2 penalty that pulls each embedding in $\mathbf{e}$ toward the mean of the vocabulary embedding matrix. This encourages the adversarial suffix to remain within a plausible region of the embedding space.

In addition to the embedding-level regularization, we introduce a KL divergence term between the output distribution of the adversarial input $Q = f([\mathbf{e}_x \| \mathbf{e}_{\text{adv}}])$ and the model's original output distribution on the unperturbed prompt $P = f(\mathbf{e}_x)$. This term penalizes large deviations in the model's predictive distribution caused by the adversarial suffix, effectively constraining the perturbation to preserve the semantic intent of the original input. The full loss function at each iteration is given by

$$\mathcal{L}_{\text{total}} = \mathcal{L}(f([\mathbf{e}_x \| \mathbf{e}_{\text{adv}}]), y) + \frac{\lambda}{2} \sum_{i=1}^{m} \|\mathbf{e}_{\tilde{x}_i} - \bar{\mathbf{e}}\|^2 + \beta \cdot \text{KL}(Q \| P) \quad (3)$$

where $\bar{\mathbf{e}}$ is the average of all token embeddings in the vocabulary, and λ, β are hyperparameters controlling the regularization strength.

By combining L2 and KL regularization, our method ensures that adversarial suffixes both stay close to known token representations in embedding space and avoid causing drastic shifts in model output distribution. This dual regularization strategy improves both the semantic plausibility and robustness of the generated adversarial triggers.

The algorithm is provided below (see Algorithm 2)

Algorithm 2 Gradient Descent with L2 and KL Regularization

Require: LLM $f(\cdot)$, prompt token embeddings $\mathbf{e}_x \in \mathbb{R}^{n\times d}$, suffix token embeddings $\mathbf{e}_{\text{adv}} \in \mathbb{R}^{m\times d}$, target token sequence $\boldsymbol{y}$, loss function $\mathcal{L}$

Require: Learning rate $\alpha > 0$, L2 regularization weight $\lambda \geq 0$, KL regularization weight $\beta \geq 0$, number of iterations T

Initialize $\mathbf{e}_{\text{adv}}$
Compute original output distribution $P \leftarrow f(\mathbf{e}_x)$
for $t \leftarrow 1$ **to** T **do**
  Compute base loss: $\text{Loss}_t \leftarrow \mathcal{L}(f([\mathbf{e}_x \| \mathbf{e}_{\text{adv}}]), \boldsymbol{y})$
  Add L2 regularization:

$$\text{Loss}_t \leftarrow \text{Loss}_t + \frac{\lambda}{2} \sum_{i=1}^{m} \left\| \mathbf{e}_{\tilde{x}_i} - \frac{1}{V} \sum_{j=1}^{V} \mathbf{e}_j \right\|_2^2$$

  Compute output distribution with perturbation: $Q \leftarrow f([\mathbf{e}_x \| \mathbf{e}_{\text{adv}}])$
  Add KL regularization:

$$\text{Loss}_t \leftarrow \text{Loss}_t + \beta \cdot \text{KL}(Q \parallel P)$$

  Compute gradient: $\boldsymbol{G}_t \leftarrow \nabla_{\mathbf{e}_{\text{adv}}} \text{Loss}_t$
  Update: $\mathbf{e}_{\text{adv}} \leftarrow \mathbf{e}_{\text{adv}} - \alpha \boldsymbol{G}_t$
end for
for $i \leftarrow 1$ **to** m **do**
  $\tilde{x}_i \leftarrow \arg\min_j \|\mathbf{e}_{\tilde{x}_i} - \mathbf{e}_j\|_2^2$
end for
return Adversarial suffix tokens $\tilde{\boldsymbol{x}}_{\text{adv}} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_m]$

3.4. Beam Search-Based Adversarial Attack (BEAST)

When performing adversarial attacks on large language models (LLMs), it is impractical to exhaustively enumerate all token replacement possibilities, especially when the input is long and the vocabulary is large. BEAST aims to generate adversarial examples quickly and efficiently without relying on the internal structure of the model. It works even if the model is a black-box and expensive to call. BEAST is a search-based combinatorial optimization strategy. Its basic process can be viewed as first finding the tokens that are most sensitive to the model's behavior, and then gradually replacing these tokens with heuristic search to construct adversarial examples.

The basic idea of BEAST is to put the input prompt after a given prefix, and then try to add tokens with high attack success rates after the prompt one by one until the length limit L is reached. First, BEAST will initialize a beam and sample k_1 tokens from the LM without replacement after a given prefix and input. This k_1 is used to control the width of the beam. In the next $L-1$ rounds of iterations, each round of iteration expands k_2 tokens to each beam to form $k_1 * k_2$ new candidates. Each candidate is concatenated with a suffix and sent to the language model to be evaluated by a function for its adversarial target value. The k_1 sequences with the lowest scores are selected to form a new round of beams. After $L-1$ rounds of iterations, the best tokens are selected and concatenated into the prompt input at the beginning, and the prefix and suffix are added to form the new prompt used to attack LLM.

The algorithm is provided below (see Algorithm 3)

Algorithm 3 BEAST

Require: LM output modelled by $p(\cdot|x)$ for input x
Input: tokenised prompt $x = x^{(s_1)} \oplus x^{(u)}, \oplus x^{(s_2)}$, beam widths k_1, k_2, adversarial suffix length L, adversarial objective $\mathcal{L}$
Output: adversarial prompt $x' = x^{(s_1)} \oplus x^{(u)} \oplus x^{(a)} \oplus x^{(s_2)}$

$x^* \leftarrow [\,], \quad s^* \leftarrow +\infty$ ▷ best sequence & score so far
$beam \leftarrow [\,]$
$p \leftarrow p(\cdot|x^{(s_1)} \oplus x^{(u)})$
$x_1, \ldots, x_{k_1} \sim \text{MultinomialSampling}(p, k_1)$
for $i \leftarrow 1$ to k_1 **do**
    $beam.\text{append}(x^{(s_1)} \oplus x^{(u)} \oplus [x_i])$
end for
for $\ell \leftarrow 2$ to L **do**
    $candidates \leftarrow [\,]$
    for $i \leftarrow 1$ to k_1 **do**
        $p \leftarrow p(\cdot|beam[i])$
        $x_1, \ldots, x_{k_2} \sim \text{MultinomialSampling}(p, k_2)$
        for $j \leftarrow 1$ to k_2 **do**
            $candidates.\text{append}(beam[i] \oplus [x_j])$
        end for
    end for
    $scores \leftarrow [\,]$
    for $i \leftarrow 1$ to $k_1 k_2$ **do**
        $scores.\text{append}(\mathcal{L}(candidates[i] \oplus x^{(s_2)}))$
    end for
    $beam, scores \leftarrow \text{bottom-}k_1(candidates, scores)$
    $x^*, s^* \leftarrow \text{bottom-1}(beam \oplus x^{(s_2)}, scores \cup \{s^*\})$
end for
return $x^*[0] \oplus x^{(s_2)}$

4. Results

We conducted experiments on four adversarial methods, including PGD, RR with L2 regularization, RR with both L2 and KL regularization, and BEAST. First of all, we trained the suffix-generating model, including Llama2-7B-Chat [16], Falcon-7B-Instruct [17], and MPT-7B-Chat [18]. Then use Meta-Llama3-8B-Instruct [19] and Beaver-7b-v1.0-cost [20] as our evaluation models. In addition, we use the adversarial success rates (ASR) as our key metric. We report the number of successful adversarial prompts that achieved ASR scores greater than 10 and 5, respectively, across two datasets: AdvBench (adversarial-only) and HarmBench (harmful-behavior).

According to Table 1, we observe that the basic PGD method achieves only modest performance, with an average ASR of 16% for ASR[>10] and 18% for ASR[>5]. Incorporating L2 regularization significantly boosts performance, improving average ASR to 24% and 35%, respectively (Table 2). This confirms our earlier hypothesis that constraining adversarial suffixes to stay within the token embedding manifold helps optimization by producing more effective and transferable perturbations.

When KL divergence regularization is additionally applied, the ASR increases further to 26% and 30% for ASR[>10] and ASR[>5], respectively (Table 2). Notably, RR with L2+KL yields higher success rates under the HarmBench setting than PGD or RR with L2, demonstrating the benefit of output-distribution-level control in preserving attack relevance while maintaining stealth.

For the BEAST method, adversarial prompts from the AdvBench and HarmBench datasets were used on the Llama3-8B-instruct language model. Since BEAST is a gradient-free adversarial attack that iteratively expands a suffix to append to an adversarial prompt with the goal being to maximizing the ASR, during training, Llama3-8b is used to evaluate the harmfulness. And the result of the outputs will be evaluated from both Llama3-8B and the Beaver score. Results were obtained on the AdvBench and HarmBench datasets, each containing 50 prompts. BEAST showed a stronger performance on AdvBench, achieving a score of 24/50 for both ASR[>10] and ASR[>5], making BEAST the highest-performing approach on the AdvBench dataset for ASR[>10] and ASR[>5]. Similarly, for the HarmBench dataset, comparing the results on the Llama3-8B-instruct model to those of other models on the Llama2-7b Chat model, the ASR[>10] score either met or surpassed that of all other evaluated models. For ASR[>5], the scores obtained were strictly greater. (Table 3)

[1]According to the original paper, 50 samples were used per dataset. Since we used only two datasets, the number of samples will differ from the original paper.

Model	Dataset	PGD (paper)		PGD (ours)	
		ASR[>10]	ASR[>5]	ASR[>10]	ASR[>5]
Llama2-7b Chat	AdvBench	4	6	10	11
	HarmBench	3	4	6	7
	Overall (%)	7%[1]	10%[1]	16%	18%
Falcon-7B-Instruct	AdvBench	10	11	9	10
	HarmBench	10	12	10	12
	Overall (%)	20%[1]	23%[1]	19%	22%
MPT-7B-Chat	AdvBench	11	13	10	10
	HarmBench	6	7	7	8
	Overall (%)	17%[1]	20%[1]	17%	18%

Table 1. PGD results, comparing the original paper and ours

Model	Dataset	RR (paper)		RR with L2 (ours)		RR with L2 + KL (ours)	
		ASR[>10]	ASR[>5]	ASR[>10]	ASR[>5]	ASR[>10]	ASR[>5]
Llama2-7b Chat	AdvBench	18	21	15	18	17	18
	HarmBench	10	13	8	17	9	15
	Overall (%)	28%[1]	34%[1]	24%	35%	26%	30%
Falcon-7B-Instruct	AdvBench	18	21	14	14	19	19
	HarmBench	21	26	12	15	24	26
	Overall (%)	39%[1]	47%[1]	26%	29%	43%	45%
MPT-7B-Chat	AdvBench	16	22	10	11	10	10
	HarmBench	15	22	6	10	8	9
	Overall (%)	31%[1]	44%[1]	16%	21%	18%	19%

Table 2. RR results, comparing the original paper and ours

5. Conclusion

In the process of implementing the algorithm, the limitations of time and computing resources are very crucial. In the experiments of PGD and RR+L2, the original paper mentioned that they used NVIDIA A6000, while we used NVIDIA A100. Even though it is enough for the original algorithm, after adding the KL loss, 40 GB of VRAM is obviously not enough. So, we have to reduce the calculation accuracy since using L2+KL loss based on RR will significantly increase the video memory and convergence time. Additionally, we found that after adjusting the coefficient of KL loss, the model outputs can be more readable than with L2 alone. However, we also found that this method has limitations. Due to the properties of the gradient method, PGD and RR need to access the model parameters and can only be used with the white box model. Although adding KL can increase the readability of the RR attack, since KL controls the probability of multiple distributions, it is often a large value. As a result, this method is very sensitive to parameters, and parameter adjustment often takes a long time for different language models. In addition, we also experimented with different regularizations. We tried Magnet regularization [21], which calculates the distance from the token to the embedding. However, it may be because L2 tends to update the gradient like the original input, while Magnet tends to update the gradient towards the embedding, resulting in the two regularization terms often being opposite. In this case, we discarded the Magnet regularization.

In addition, BEAST's advantages are also obvious because it does not actually need to access the parameters of the model and can be optimized without a gradient. During the attack on the language model setting, we tested it on Llama3 8b model, which is more advanced than Llama2 7b model. Even with this experimental setup, BEAST still performed better ASR than the other three algorithms (see Table 3). In fact, after we manually read the outputs of

Model	Dataset	BEAST (ours)	
		ASR[>10]	ASR[>5]
Meta-Llama3-8B-Instruct	AdvBench	24	24
	HarmBench	10	18
	Overall (%)	34%	42%

Table 3. BEAST results

different algorithms, the outputs from the BEAST model are also the most readable among the rest. However, since there is no truly reliable benchmark for readability, we did not specifically compare readability. But on the other hand, the attack time of BEAST is significantly longer than PGD and RR. This happened because the BEAST method needs to perform beam search on a large range of possible tokens. Additionally, there is no definite optimal option for the evaluation function of the prompt's harmfulness, and the selection of the prompt's prefix. Especially for the prefix matter, since the choice of prefix directly determines the direction of the answer, but BEAST does not train these prefixes. So the choice of prefix is still a rule of thumb.

Our current adversarial attack approach on LLMs is to get an answer through only one specific prompt. There is no relationship between each prompt, and the attack cannot continue once it has successfully attacked. From our current algorithms, we can see that for longer prompts, the amount of computation and memory consumption could increase dramatically. So in the future, we can try multi-round dialogue attacks, such as introducing the transformer's attention mechanism in multi-round attacks, so as to achieve multi-round long dialogue attacks. Finally, the ultimate goal of attacking LLM is to strengthen the security of LLM by finding vulnerabilities. So, using token-level attacks to find more dangerous tokens, promote LLM's ethical review capabilities, and make LLM safer are also interesting domains.

References

[1] Bernard Marr. A Short History of ChatGPT: How We Got to Where We Are Today. *Forbes*, May 2023. URL: https://www.forbes.com/sites/bernardmarr/2023/05/19/a-short-history-of-chatgpt-how-we-got-to-where-we-are-today/. 1

[2] Annie Badman. 10 AI Dangers and Risks and How to Manage Them. *IBM Think*, 2024. URL: https://www.ibm.com/think/insights/10-ai-dangers-and-risks-and-how-to-manage-them. 1

[3] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and Transferable Adversarial Attacks on Aligned Language Models. *arXiv preprint arXiv:2307.15043*, July 2023. URL: https://arxiv.org/abs/2307.15043. 1

[4] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv preprint arXiv:1706.06083*, 2017. URL: https://arxiv.org/abs/1706.06083. 1

[5] Samuel Jacob Chacko, Sajib Biswas, Chashi Mahiul Islam, Fatema Tabassum Liza, and Xiuwen Liu. Adversarial Attacks on Large Language Models Using Regularized Relaxation. *arXiv preprint arXiv:2410.19160*, October 2024. URL: https://arxiv.org/abs/2410.19160. 1, 2

[6] Vinu Sankar Sadasivan, Shoumik Saha, Gaurang Sriramanan, Priyatham Kattakinda, Atoosa Chegini, and Soheil Feizi. Fast Adversarial Attacks on Language Models in One GPU Minute. *arXiv preprint arXiv:2402.15570*, February 2024. URL: https://arxiv.org/abs/2402.15570. 1, 2

[7] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *arXiv preprint arXiv:1907.11932*, 2019. URL: https://arxiv.org/abs/1907.11932. 1

[8] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, December 2018. URL: https://arxiv.org/abs/1712.06751. 1

[9] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial Examples: Attacks and Defenses for Deep Learning. *arXiv preprint arXiv:1712.07107*, July 2018. URL: https://arxiv.org/abs/1712.07107. 1

[10] Baolin Wang, Binbin Xu, and Shiwei Liu. Universal and Efficient Jailbreaking Attacks against LLMs via Adversarial Suffixes. *arXiv preprint arXiv:2410.19160*, 2024. URL: https://arxiv.org/pdf/2410.19160.

[11] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: Adversarial Attack Against BERT Using BERT. *arXiv preprint arXiv:2004.09984*, October 2020. DOI: 10.48550/arXiv.2004.09984. URL: https://arxiv.org/abs/2004.09984. 2

[12] 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. *arXiv preprint arXiv:2406.10794*, December 2024.

DOI: 10.48550/arXiv.2406.10794. URL: https://arxiv.org/abs/2406.10794. 2

[13] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking Black Box Large Language Models in Twenty Queries. *arXiv preprint arXiv:2310.08419*, July 2024. URL: https://arxiv.org/abs/2310.08419. 2

[14] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. *arXiv preprint arXiv:2312.02119*, October 2024. DOI: 10.48550/arXiv.2312.02119. URL: https://arxiv.org/abs/2312.02119. 2

[15] Anselm Paulus, Arman Zharmagambetov, Chuan Guo, Brandon Amos, and Yuandong Tian. AdvPrompter: Fast Adaptive Adversarial Prompting for LLMs. *arXiv preprint arXiv:2404.16873*, June 2025. URL: https://arxiv.org/abs/2404.16873. 2

[16] Meta AI. Llama-2-7B-hf: Foundation pretrained 7 billion-parameter Llama 2 model. Hugging Face model card, released July 18 2023 under the LLAMA 2 Community License. URL: https://huggingface.co/meta-llama/Llama-2-7b-hf. 4

[17] Technology Innovation Institute (TII). Falcon-7B-Instruct: A 7 billion-parameter instruction-tuned model based on Falcon-7B. Hugging Face model card, released approximately June 2023, Apache 2.0 license. URL: https://huggingface.co/tiiuae/falcon-7b-instruct. 4

[18] MosaicML NLP Team. MPT-7B-Chat: A chatbot-style model finetuned from MPT-7B on ShareGPT, Vicuna, Alpaca, HH-RLHF, HC3, and Evol-Instruct. Hugging Face model card, released May 5 2023 under the CC-BY-NC-SA 4.0 license. URL: https://huggingface.co/mosaicml/mpt-7b-chat. 4

[19] Meta AI. Meta-Llama-3-8B-Instruct: An instruction-tuned version of the 8 billion-parameter Llama 3 model. Hugging Face model card, released April 18 2024 under the Meta Llama 3 Community License. URL: https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct. 4

[20] PKU-Alignment Team. Beaver-7B-v1.0-cost: A preference cost model trained on PKU-SafeRLHF. Hugging Face model card, published approximately July 10 2023. URL: https://huggingface.co/PKU-Alignment/beaver-7b-v1.0-cost. 4

[21] Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Valentin Hofmann, Tomasz Limisiewicz, Yulia Tsvetkov, and Noah A. Smith. MAGNET: Improving the Multilingual Fairness of Language Models with Adaptive Gradient-Based Tokenization. *arXiv preprint arXiv:2407.08818*, November 2024. DOI: 10.48550/arXiv.2407.08818. URL: https://arxiv.org/abs/2407.08818.

5