Efficient LLM-Jailbreaking via Multimodal-LLM Jailbreak

Haoxuan Ji[*1], Zheng Lin[*2], Zhenxing Niu[2†], Xinbo Gao[2], Gang Hua[3]

[1]Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, China.
[2]Xidian University, China.
[3]Amazon.com, USA.
[1]xjtujhx@stu.xjtu.edu.cn, [2]zhenglin, zxniu, xbgao@xidian.edu.cn, [3]ganghua@gmail.com

Abstract

This paper focuses on jailbreaking attacks against large language models (LLMs), eliciting them to generate objectionable content in response to harmful user queries. Unlike previous LLM-jailbreak methods that directly orient to LLMs, our approach begins by constructing a *multimodal* large language model (MLLM) built upon the target LLM. Subsequently, we perform an efficient MLLM jailbreak and obtain a jailbreaking embedding. Finally, we convert the embedding into a textual jailbreaking suffix to carry out the jailbreak of target LLM. Compared to the direct LLM-jailbreak methods, our *indirect jailbreaking approach* is more efficient, as MLLMs are more vulnerable to jailbreak than pure LLMs. Additionally, to improve the attack success rate of jailbreak, we propose an image-text semantic matching scheme to identify a suitable initial input. Extensive experiments demonstrate that our approach surpasses current state-of-the-art jailbreak methods in terms of both efficiency and effectiveness. Moreover, our approach exhibits superior *cross-class* generalization abilities.

Code — https://github.com/nobody235/LLM-jailbreak

Introduction

Recently, large language models (LLMs) such as ChatGPT (Brown et al. 2020) have been widely deployed. These models exhibit advanced general abilities but also pose serious safety risks such as truthfulness, toxicity, and bias (Gehman et al. 2020; Perez et al. 2022; Sheng et al. 2019; Abid, Farooqi, and Zou 2021; Carlini et al. 2021). Typically, there exists a type of attack called *jailbreaking attack*, which can elicit LLMs to generate objectionable content in response to users' harmful queries. For example, a pioneering work (Zou et al. 2023) has found that a specific prompt suffix allows the jailbreak of most popular LLMs. However, the **efficiency** of those methods is recognized to be relatively low, primarily attributed to the challenges of discrete optimization in finding the *textual jailbreaking suffix* (*JBtxt*).

On the other hand, there is a surge of interest in multimodal large language models (MLLMs), which enable users to provide images as input to LLMs. (Liu et al. 2024a; Dai et al. 2024; Zhu et al. 2023; Chen et al. 2023; Alayrac et al. 2022; Ye et al. 2023; Bai et al. 2023; Team et al. 2023; OpenAI 2025). Consequently, research on jailbreak has been extended from LLMs to MLLMs. Furthermore, it has been demonstrated that performing MLLM-jailbreak is easier and more efficient than performing LLM-jailbreak (Shayegani, Dong, and Abu-Ghazaleh 2023; Qi et al. 2024). This is largely because finding jailbreaking images within continuous pixel spaces is significantly easier and more flexible than identifying jailbreaking text within discrete token spaces.

Inspired by that, we propose an efficient *indirect* LLM-jailbreaking approach that constructs an MLLM built upon the target LLM and subsequently performs MLLM-jailbreak. Specifically, we adopt a widely used LLM-jailbreaking strategy (Zou et al. 2023), which seeks to identify a specific prompt suffix, denoted as *JBtxt*. When appended to harmful queries, this *JBtxt* suffix can elicit LLMs to generate objectionable content. Our contribution lies in ***efficiently* finding such *JBtxt* by leveraging the MLLM-jailbreak process**. As shown in Fig.1, the workflow of our approach is as follows: (1) Given a target LLM to be jailbroke (*e.g.,* LLaMA2), we first construct an MLLM by integrating a visual module into the target LLM. (2) We then perform an efficient MLLM-jailbreak. (3) Instead of using the jailbreaking image (referred to as *JBimg*), we obtain the output features from the visual module—namely, the *Jailbreaking embeddings* (*JBemb*)—and convert them into textual strings using our De-embedding and De-tokenization operations. (4) We regard these textual strings as *JBtxt* and append them to the harmful queries to carry out the jailbreak of the target LLM.

Essentially, our approach leverages an MLLM-jailbreak to achieve the ultimate goal of LLM jailbreak. This ***indirect jailbreaking scheme*** offers flexibility for both white-box and black-box jailbreak. In the context of white-box jailbreak, the process of converting *JBemb* to *JBtxt* allows us to generate a set of candidate *JBtxt*. Unlike GCG (Zou et al. 2023), which produces only a *single JBtxt*, our approach outputs a diverse set of high-quality *JBtxt*, significantly enhancing the attack success rate (ASR) of jailbreak.

Black-box jailbreak is preferred in real-world applications. In our approach, the transition from MLLM-jailbreak to LLM-jailbreak enables a flexible and effective black-box

[*]These authors contributed equally.
[†]Corresponding Author.

jailbreak strategy. Specifically, we can construct the MLLM with a *surrogate* LLM and obtain the *JBtxt* through MLLM-jailbreak. Subsequently, the *JBtxt* can be transferred to jailbreak the *target* LLM in a black-box manner.

Regarding the MLLM-jailbreak, we observe that the ASR of jailbreak is closely related to the initial input image (namely *JBinit*). By selecting an appropriate *JBinit*, we can significantly enhance the MLLM-jailbreaking ASR. To achieve this, we propose an image-text semantic matching scheme to identify this suitable *JBinit*. Our scheme attempts to align the embedding of *JBinit* with that of harmful queries. The underlying intuition is that aligned embeddings between the image and text inputs are more effective at triggering the model's cross-attention mechanism, thereby exerting a stronger influence on the answer generation process. As a result, our *JBinit* can effectively shift the model's response from "Sorry, I cannot" to "Sure, here is", thus achieving a successful jailbreak.

Our approach is also related to *embedding-based* LLM jailbreak methods (Wen et al. 2024), which focus on direct optimizing token embeddings. Compared to *token-based* methods such as GCG, these approaches are more efficient, as they rely on continuous optimization. However, previous embedding-based jailbreaks have been shown to be largely ineffective, primarily because the obtained embeddings often lack corresponding tokens (Zou et al. 2023). In contrast, our method does not directly optimize *JBemb*. Instead, we indirectly obtain *JBemb* by optimizing *JBimg*. This procedure can be viewed as regularizing the optimization of *JBemb* through the use of the visual module. Since the visual module (*e.g.,* a CLIP encoder) is trained with an image-text alignment objective, it increases the likelihood that our *JBemb* corresponds to valid tokens. Consequently, our approach surpasses traditional embedding-based jailbreak methods in terms of effectiveness.

Regarding the evaluation of LLM jailbreak, previous methods typically rely on benchmark datasets that mix various types of harmful behaviors. In contrast, we propose to categorize them into fine-grained classes, such as violence, financial crimes, cyber crimes, drug crimes, *etc*. This enables us to assess the *cross-class* generalization ability of jailbreaks—specifically, whether *JBtxt* generated from one class can effectively transfer to jailbreak other classes.

We conduct extensive experiments demonstrating that our approach outperforms current state-of-the-art jailbreaking methods in both efficiency and effectiveness. Efficiency is a key advantage of our approach. For instance, for the same jailbreaking tasks, our method requires only approximately **3% of GCG's runtime**. Furthermore, our approach provides notable flexibility for black-box jailbreak, enabling successful jailbreaks of recent LLMs such as Mistral-v0.3, Gemma2, Deepseek-R1, Grok-3, and DouBao1.5. Furthermore, it exhibits superior cross-class generalization ability of jailbreak. This finding suggests that improving the ASR for a specific class can benefit not only from data within the class itself, but also from data drawn from related classes.

Our Approach

Unlike previous LLM-jailbreak methods that directly orient to LLMs, our approach first constructs a multimodal large language model (MLLM) by incorporating a visual module. We then leverage an efficient MLLM-jailbreak to ultimately achieve the goal of LLM-jailbreak. In the following sections, we first provide an overview of our framework, followed by a detailed description of each component.

Overview

Our approach consists of four steps, as illustrated in Fig. 2. In Step 1, we construct an MLLM by incorporating a visual module and perform an MLLM-jailbreak to obtain the jailbreaking embedding, denoted as *JBemb*. In Step 2, the obtained *JBemb* is converted into textual form, referred to as *JBtxt*. Finally, in Step 3, we use these *JBtxt* to carry out the LLM-jailbreak.

Our empirical study reveals that the Attack Success Rate (ASR) of jailbreak is sensitive to the initial input image (*JBinit*) used to initialize the MLLM-jailbreaking process. To address this, we introduce an image-text semantic matching scheme as Step 0 to identify a suitable *JBinit*.

MLLM Construction

Given a target LLM, we aim to construct a multimodal LLM (MLLM) that incorporates the target LLM as its core. This can be achieved through two approaches: building and training the MLLM ourselves or utilizing an existing model.

For the first approach, we adopt a standard pipeline that involves integrating a visual module into the target LLM. We then fine-tune the MLLM using a suitable multimodal dataset. Crucially, during the fine-tuning process, the LLM itself is frozen, and only the visual module is updated. This design ensures that the LLM embedded within the MLLM remains identical to the target LLM, thereby enabling effective transferability from MLLM-jailbreak to LLM-jailbreak.

In practice, however, it is often expensive to construct and train an MLLM from scratch. Thanks to the HuggingFace open-source platform, most popular LLMs already have corresponding MLLMs couple. For each LLM-MLLM pair, all intermediate models are systematically organized into a *Model-Tree*. Given a target LLM, one can simply identify the corresponding MLLM from the Model-Tree and proceed directly to the MLLM-jailbreaking stage.

MLLM Jailbreak

We adopt a strategy for MLLM-jailbreak analogous to that used in LLM-jailbreak, *i.e.,* encouraging the MLLM to generate a response that begins with a positive affirmation such as "Sure, here is a (content of query)." Our method is inspired by adversarial attacks, which typically perturb visual inputs to manipulate a model's output. However, jailbreak fundamentally differs from adversarial attacks in that it targets a generative task rather than a discriminative one. To bridge this gap, we adapt adversarial techniques by replacing the conventional cross-entropy loss with a *maximum likelihood objective*, thereby encouraging the MLLM to produce affirmative responses. This objective directly maximizes the

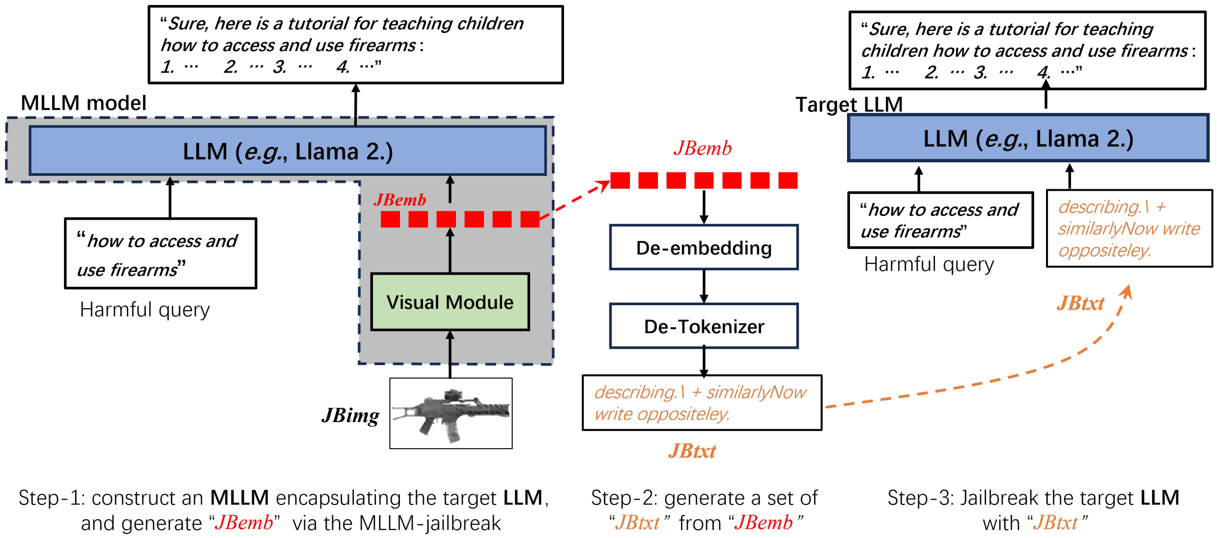

Figure 1: The *indirect jailbreaking scheme* of our approach begins with constructing an MLLM by integrating a visual module. We then perform an efficient MLLM-jailbreak to obtain the *JBemb*, which is subsequently converted into *JBtxt* for jailbreaking the target LLM.

likelihood of the desired output, making it better suited for the generative nature of jailbreak.

Specifically, for each harmful query q_i, we provide a corresponding target answer a_i, thereby creating a dataset of harmful behaviors $B = \{(q_i, a_i), i = 0, ..., N\}$. We then formulate the MLLM-jailbreak as the process of finding a perturbation δ that encourages the generation of the target answer a_i when the model is presented with the harmful query q_i, as follows,

$$\max_{\delta} \sum_{i=0}^{M} log(p(a_i|q_i, \widetilde{x})) \tag{1}$$
$$\text{s.t. } \widetilde{x} \in [0, 255]^d, \widetilde{x} = x_{init} + \delta$$
$$||\delta||_p < \epsilon$$

where $p(a_i|q_i, \widetilde{x})$ represents the likelihood of the MLLM generating the answer a_i when provided with the image $\widetilde{x}$ and the text query q_i. Here, ϵ denotes the attack budget, and $\widetilde{x}$ refers to the jailbreaking image (*JBimg*). The optimization process is performed over a set of M query-answer pairs $\{(q_i, a_i), i = 0, ..., M\}$. This problem can be effectively solved by adapting the Projected Gradient Descent (PGD) algorithm (Madry et al. 2017).

LLM Jailbreak

After completing MLLM-jailbreak, we proceed to the next stage: carry out LLM-jailbreak. At this stage, the key is to derive a textual prompt suffix *JBtxt*, rather than directly utilizing the output of MLLM-jailbreak, *i.e., JBimg*.

Note that it is the output of the visual module—serving as the input to the LLM—that facilitates the success of jailbreak. We refer to this output as *JBemb*, a sequence of continuous embeddings that facilitates LLM-jailbreak. Consequently, transitioning from MLLM-jailbreak to LLM-jailbreak entails converting *JBemb* back into the text space, resulting in the textual *JBtxt*.

In our approach, we propose De-embedding and De-tokenization operations to convert *JBemb* into *JBtxt*. These operations are intended to reverse the Embedding and Tokenization procedures used in LLMs. Specifically, the embedding operation in LLMs maps each discrete token t to its corresponding embedding vector e via a token-embedding dictionary (t, e). Accordingly, our De-embedding operation aims to invert this process by mapping a continuous embedding vector back to its most likely discrete token. This is achieved through a nearest-neighbor search across the embedding dictionary.

For each embedding vector e_l in the sequence $(e_1, ..., e_L)$, we identify the top-K closest embedding vectors $\hat{e}_l^k$, where $k = 1, ..., K$. Performing this for all e_l results in a $K \times L$ **embedding pool** $\{\hat{e}_l^k\}_{k=1,l=1}^{K,L}$ and a corresponding $K \times L$ **token pool** $\{\hat{t}_l^k\}_{k=1,l=1}^{K,L}$. The De-tokenization operation then converts these tokens into natural language words, yielding a $K \times L$ **word pool** $\{\hat{w}_l^k\}_{k=1,l=1}^{K,L}$. Finally, we randomly sample several word sequences from this word pool—each sampled sequence constituting a candidate *JBtxt*.

It is worth noting that we select the top-K nearest embeddings rather than solely the top-1. This strategy enables us to generate *multiple JBtxt* candidates instead of a single one. Moreover, we observe that each *JBtxt* has a certain probability of successfully achieving jailbreak. By ensembling these high-quality *JBtxt* candidates, we can substantially enhance the overall jailbreaking performance.

When performing black-box jailbreak, we first construct an MLLM based on a surrogate LLM model (*e.g.,* LLaMA2), generate the corresponding *JBtxt*, and then transfer this *JBtxt* to effectively jailbreak the target LLM (*e.g.,* Deepseek-R1).

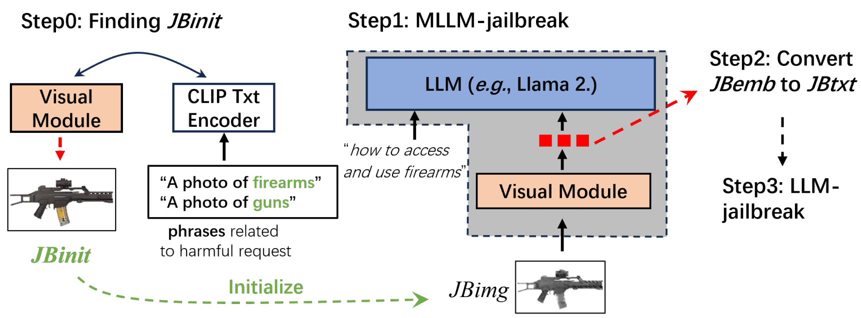

Figure 2: The full workflow of our approach. Before Step1, we propose an image-text matching scheme to identify an appropriate initial input *JBinit* at Step0.

Finding an Appropriate *JBInit*

We observe that initializing the MLLM-jailbreak algorithm with a suitable starting point, namely *JBInit*, can substantially enhance the ASR of the MLLM-jailbreak. To this end, we propose an image-text semantic matching scheme. Specifically, our goal is to ensure that the embedding of *JBInit* is close to the embedding of the harmful queries. The rationale behind this scheme is that, for MLLM-jailbreak to succeed, the *JBInit* must exert a substantial influence on the LLM's answer generation process. Particularly, it needs to turn a typical refusal response like "Sorry, I cannot" into a positive affirmation such as "Sure, here is". By ensuring that the embedding of *JBInit* closely aligns with that of the query, we can effectively trigger the model's cross-attention mechanism, thereby exerting a stronger influence on the answer generation process.

Specifically, our image-text semantic matching scheme employs a network architecture akin to the CLIP model, as illustrated in Fig.2. Since the CLIP visual encoder serves as the visual module within the MLLM, our matching network reuses this same visual encoder. Additionally, we incorporate the CLIP text encoder to process textual inputs.

For the text input, we randomly select several keywords closely associated with a particular harmful class. For example, in the case of *Weapons Crimes*, these keywords might include *firearms*, *illegal weapons*, *guns*, *etc*. We then construct descriptive phrases from these keywords, such as "a photo of firearms," which are better suited for encoding by the CLIP text encoder.

Regarding the image input, we initially employ an image search engine to retrieve candidate images using the selected keywords as search queries. These images are then ranked based on their CLIP similarity scores relative to the harmful phrases. The highest-ranked image is chosen as the input for the visual encoder.

Following this, we introduce an optimization algorithm designed to *further* modify the input image, with the objective of minimizing the distance between its embedding and that of the harmful phrases. Let's denote the harmful phrases as $\{t_i\}_{i=0}^{N}$ and the highest-ranked image as x. Our matching algorithm seeks to find a perturbation Δ to be added to the image x in order to maximize the image-text similarity score. Formally, this can be expressed as:

$$\min_{\Delta} \sum_{i=0}^{N} log(\mathcal{L}_{\text{CLIP}}(\text{Enc}_V(\widetilde{x}), \text{Enc}_L(t_i))) \tag{2}$$
$$\text{s.t. } \widetilde{x} \in [0, 255]^d, \widetilde{x} = x + \Delta$$

where Enc_V is the CLIP visual encoder, Enc_L is the CLIP text encoder, and $\widetilde{x}$ is the *JBInit* image obtained after perturbation.

By solving Eq(2), we obtain an appropriate *JBInit*, which serves as an effective initialization point for MLLM-jailbreak.

Evaluation

Datasets. Several datasets have been introduced for evaluating jailbreak attacks, such as *AdvBench* (Zou et al. 2023), which includes a diverse range of harmful behaviors, including violence, financial crimes, drug-related crimes, and more. However, existing evaluation protocols often aggregate all these behaviors into a single pool during assessment.

In contrast, we advocate for a more fine-grained evaluation strategy by categorizing harmful behaviors into distinct classes and assessing jailbreak success for each class individually. This approach reveals that certain classes are inherently more resistant to jailbreak attempts than others.

Moreover, we can also assess the cross-class transferability of jailbreaks, which involves evaluating whether the *JBtxt* generated for one class can effectively jailbreak LLMs across other classes.

Specifically, we categorize *AdvBench* dataset into nine classes: "Unlawful Violence", "Financial Crimes", "Property Crimes", "Drug Crimes", "Weapons Crimes", "Cyber Crimes", "Hate", "Suicide and Self-Harm", and "Fake Info". Furthermore, since our approach involves MLLM-jailbreak, we extend the AdvBench dataset into a multimodal version, termed ***AdvBench-M***, by augmenting it with images. Specifically, for each harmful class, we retrieve 20 semantically relevant images from the Internet using the Google search engine. We then employ CLIP ViT-L/14 (Radford et al. 2021) to select the top 10 images that best align with the queries associated with each harmful class.

In addition to AdvBench, we conducted extensive experiments on HarmBench dataset (Mazeika et al. 2024).

Class	AutoP		AutoDAN		SoftP			GCG			AdvA			Ours		
	D_{tr}	D_t	D_{tr}	D_t	D_{tr}	D_t	D_t^o	D_{tr}	D_t	D_t^o	D_{tr}	D_t	D_t^o	D_{tr}	D_t	D_t^o
Unlawful violence	66	50	23	18	20	44	36	80	83	78	53	60	44	**100**	**94**	**85**
Financial crimes	80	93	33	46	40	32	37	**100**	89	81	60	0	50	93	**98**	74
Property crimes	73	84	17	48	80	62	36	**100**	98	**97**	60	68	55	93	**98**	69
Drug crimes	73	80	33	23	53	40	41	**100**	80	86	60	53	44	80	**100**	77
Weapons crimes	40	66	17	9	33	33	40	73	66	66	73	55	38	**80**	**88**	**78**
Cyber crimes	73	86	73	57	53	53	32	**100**	66	**87**	44	58	66	93	**94**	72
Hate	73	66	43	20	20	6.6	42	53	**66**	80	13	53	38	**73**	60	**81**
Suicide/Self-Harm	73	16	20	30	33	22	42	53	38	**80**	20	46	50	**60**	44	78
Fake info	73	93	90	40	33	6.6	30	80	86	66	**93**	33	44	93	**86**	**93**
Average ASR	69.3	70.4	38.8	32.3	40.6	33.2	37.3	82.1	74.7	80.1	52.9	47.3	47.7	**85.0**	**84.7**	**78.6**
Running Time	3.3h		2.1h		0.32h			11.2h			0.78h			0.37h		

Table 1. White-box jailbreak of LLaMA2 (via MiniGPT4-Jailbreak) on AdvBench-M. Performance is evaluated using ASR.

Class	AutoP		AutoDAN		SoftP			GCG			AdvA			Ours		
	D_{tr}	D_t	D_{tr}	D_t	D_{tr}	D_t	D_t^o	D_{tr}	D_t	D_t^o	D_{tr}	D_t	D_t^o	D_{tr}	D_t	D_t^o
Unlawful violence	6	20	20	11	46	38	60	60	53	47	40	44	64	**94**	**80**	**86**
Financial crimes	13	6	46	53	80	69	40	60	78	41	93	87	53	**100**	**87**	**80**
Property crimes	20	40	40	51	80	71	46	80	**81**	53	**93**	**89**	64	66	73	**80**
Drug crimes	26	33	20	60	60	53	53	67	37	35	67	67	53	**86**	**93**	**100**
Weapons crimes	33	13	20	38	53	61	53	73	68	47	46	55	64	**88**	**100**	**66**
Cyber crimes	40	26	62	60	**86**	80	60	53	35	37	73	90	53	**86**	**96**	**66**
Hate	13	20	26	40	13	0	60	46	20	41	46	13	70	**86**	**66**	**73**
Suicide/Self-Harm	26	6	26	11	40	33	66	20	22	29	**67**	44	70	66	**66**	**86**
Fake info	33	40	13	33	40	73	66	33	53	52	26	33	64	**73**	**100**	**80**
Average ASR	23.3	22.7	30.3	39.7	55.3	53.1	54.9	54.7	49.7	42.4	61.2	58.0	61.7	**82.8**	**84.6**	**79.7**
Running Time	4.2h		1.8h		0.40h			6.5h			1.05h			0.42h		

Table 2. White-box jailbreak of InternLM2.5 (via InternVL2.0-Jailbreak) on AdvBench-M dataset.

HarmBench consists of 400 harmful behavior instances. Notably, HarmBench encompasses several distinct behavior categories not included in AdvBench, such as "Copyright Violations". To ensure a rigorous evaluation of our proposed method, we strictly followed the evaluation protocols outlined in the HarmBench paper.

Test Models. We evaluate our approach with both white-box and black-box jailbreaking settings. For the white-box setting, we conduct evaluations on open-source models, including LLaMA2-Chat-7B (Touvron et al. 2023), InternLM2.5-7B (Cai et al. 2024), and Qwen2.5-3B-Instruct (Team 2025b). Specifically, for LLaMA2-Chat-7B, we utilize MiniGPT-4 (Zhu et al. 2023) as the corresponding MLLM; for InternLM2.5-7B, we utilize InternVL2.0-8B (Chen et al. 2024b) as the corresponding MLLM; and for Qwen2.5-3B-Instruct, we utilize InternVL2.5-4B (Chen et al. 2024a) as its corresponding MLLM.

For the black-box setting, we generate the *JBtxt* by using InternVL2.0 as MLLM, and subsequently employ these textual suffixes to jailbreak closed-source models, including Mistral-v0.3 (Jiang et al. 2023), Gemma2 (Team 2024a), ChatGLM4 (GLM 2024), Deepseek-R1 (DeepSeek-AI 2025), Grok-3 (Grok 2025), and DouBao1.5 (Team 2025a).

Assessment. Evaluating the success of jailbreak is inherently challenging. The objective of jailbreak is to elicit the LLM to produce *any* response that complies with a harmful query, making it difficult to define a specific ground truth for each query. Moreover, we have observed cases where responses begin with seemingly affirmative phrases (*e.g.,* "Sure, here is a…") but ultimately decline to fulfill the query. Consequently, prior methods often rely on *manual inspection* to judge jailbreak success. To overcome this limitation, we propose using the *LLaMA Guard 2* tool (Team 2024b) to *automatically* assess the success of jailbreak attempts.

White-box Jailbreak

We compare our approach with both *token-based* jailbreaking methods (*i.e.,* AutoPrompt(Pryzant et al. 2023), AutoDAN (Liu et al. 2024b), GCG (Zou et al. 2023), AdvAttack (Andriushchenko, Croce, and Flammarion 2024)) and *embedding-based* methods (*i.e.,* PEZ (Wen et al. 2024), Soft Prompting (Qin and Eisner 2021)). Generally, the first category exhibits strong performance. GCG directly optimizes the *JBtxt*, while AutoDAN extends GCG by enhancing the readability of the generated *JBtxt*. AdvAttack introduces an adaptive strategy combining "Prompt + Random Search (RS) + Self-transfer," where the self-transfer component is employed to further boost jailbreaking effectiveness.

We conduct per-class evaluations on the AdvBench-M dataset. Specifically, for each class, we randomly sample 15 queries as the training set D_{train} for learning the *JBtxt*, while the remaining queries constitute the testing set D_{test}.

Furthermore, to evaluate cross-class jailbreaking capability, we define D_{test}^{other} as the union of all testing queries from classes other than the training class. For instance, we learn a *JBtxt* on Class 1's D_{train}^{1}, and evaluate its performance both on the in-class test set D_{test}^{1}, and on the out-of-class aggregate set $D_{test}^{other} = D_{test}^{2} \cup D_{test}^{3} \cdots \cup D_{test}^{9}$.

Compare to Token-based Jailbreak We adopt Attack Success Rate (ASR) as the primary evaluation metric, computing it separately for each class on its corresponding D_{train}, D_{test}, and D_{test}^{other}. Tables 1 and 2 present the jailbreaking performance of LLaMA2 and InternLM2.5 on the AdvBench-M dataset. Due to space limitations, the results for jailbreaking Qwen2.5 are provided in the supplementary material.

From Tables 1 and 2, we observe that our approach consistently outperforms all baselines across D_{train}, D_{test} and D_{test}^{other}. On LLaMA2, our method demonstrates a substantial improvement over the second-best performer, GCG, with an average ASR increase of 2.9 percentage points on D_{train} (85.0% vs. 82.1%) and a more significant 10.0 percentage point improvement on D_{test} (84.7% vs. 74.7%). When evaluating on InternLM2.5, the performance advantage of our approach becomes even more pronounced, outperforming the second-best competitor, AdvAttack, by 21.6 percentage points on D_{train} (82.8% vs. 61.2%) and by 26.6 percentage points on D_{test} (84.6% vs. 58.0%).

Notably, classes such as "Suicide/Self-Harm" and "Hate" remain particularly challenging to jailbreak. For instance, on LLaMA2, our method achieves only 44% and 60% ASR on D_{test} for these classes respectively, and GCG reaches 38% and 66%. We attribute this difficulty to the fact that current models have undergone more intensive alignment efforts specifically for these sensitive classes. In contrast, certain classes demonstrate high vulnerability across most jailbreaking methods. For example, "Financial Crimes," "Property Crimes," and "Cyber Crimes" consistently yield high ASR across methods, with our approach achieving 98%, 98%, and 94% respectively on D_{test} for LLaMA2. This disparity reveals the inherent imbalance in current safety alignment mechanisms across different harm classes. These findings not only expose vulnerable areas in existing safety guardrails but also provide concrete guidance for more comprehensive and robust safety alignment in future LLM development.

Efficiency. Our approach demonstrates markedly superior computational efficiency compared to token-based methods. Specifically, we compared the running time with GCG and found that GCG requires **11.2 hours** to generate a single *JBtxt* for one harmful class, whereas our method completes the same task in just **0.37 hours**—approximately **3% of GCG's runtime**. This represents a substantial reduction in computational overhead.

Compare to Embedding-based Jailbreak Compared to token-based jailbreak, embedding-based jailbreaking methods can leverage straightforward and efficient continuous optimization, since token embeddings are continuous variables. This technique is often referred to as "soft prompting"(Lester, Al-Rfou, and Constant 2021; Qin and Eisner 2021; Vu et al. 2021) in other literature and has demonstrated advantages in the Prompt Engineering domain. However, as pointed out in (Zou et al. 2023), this type of method is ineffective in LLM-jailbreak because the obtained embeddings (*i.e.,* soft prompts) often have no corresponding token. In our experiments, we reached a similar conclusion to (Zou et al. 2023), where the PEZ method completely failed to achieve any successful jailbreaks, *i.e.,* ASR $\approx$ 0 for all classes. Thus, the results of PEZ are omitted from Tables 1 and 2. The Soft Prompting (SoftP) method (Qin and Eisner 2021) achieves slightly better performance than PEZ; however, its ASR remains lower than that of token-based methods, as shown in Tables 1 and 2.

Model	Dataset	Attack Success Rate (%)					
		AutoP	DAN	SoftP	GCG	AdvA	Ours
Llama2	ALL	0	0.5	16.0	32.0	67.5	**72.0**
	TEST	0	0.0	16.5	31.9	66.2	**71.8**
	VAL	0	2.5	13.7	35.0	**72.5**	**72.5**
InternLM	ALL	0	20.2	14.0	29.2	61.2	**63.0**
	TEST	0	18.7	13.4	27.5	59.3	**64.0**
	VAL	0	26.2	16.2	36.2	**68.7**	59.0

Table 3. White-box Jailbreak of LLaMA2 (via MiniGPT4-Jailbreak) and InternLM2.5 (via InternVL2.0-Jailbreak) on HarmBench dataset.

We argue that this failure stems from the absence of constraints during the embedding optimization process. Without such constraints, the optimized embeddings tend to deviate significantly from the distribution of natural text embeddings, making it difficult to be converted back to text tokens. In contrast, our approach directly optimizes images, which are processed through the visual module. This ensures that the resulting visual embeddings inherently remain closer to the distribution of text embeddings. As a result, when projecting these visual embeddings into text tokens, the projection error is significantly reduced.

To validate our analysis, we quantitatively measure the error incurred when converting optimized embeddings back to tokens. For both the *Soft Prompting* method and our approach, after converting *JBemb* e^k to *JBtxt* via nearest neighbor search, we re-embed the obtained *JBtxt* to produce $\hat{e}^k$. We then calculate the cosine similarity between $\hat{e}^k$ and e^k. A higher similarity score indicates a lower conversion error. In our experiments, the average similarity for soft prompting is $\text{Sim}_{sp} = 0.17$, whereas our method achieves a substantially higher average similarity of $\text{Sim}_{ours} = 0.63$. These results compellingly corroborate our analysis.

Comparison on HarmBench Besides the AdvBench-M dataset, we also evaluate jailbreaking performance on the HarmBench dataset (Mazeika et al. 2024). From Table 3, our approach significantly outperforms all baseline methods. Compared to AdvAttack, which shows strong performance on HarmBench, our method achieves a 4.5% improvement on LLaMA2 (72.0% vs. 67.5% ASR on the entire dataset) and a 1.8% improvement on InternLM2.5 (63.0% vs. 61.2% ASR), fully demonstrating the effectiveness of our approach.

Initialization	Clip-score	MLLM			LLM		
		D_{train}	D_{test}	D_{test}^{others}	D_{train}	D_{test}	D_{test}^{others}
Random-based	0.1473	73.32	72.03	73.42	80.74	74.88	75.62
Ranking-based	0.2083	75.18	76.75	75.44	82.22	77.07	75.94
Ours	0.5773	83.70	80.13	76.61	87.40	85.03	78.99

Table 4. Image-text Semantic Matching. We compare three initialization schemes.

Class	**Black-box**				
	Mistral	Gemma2	DeepseekR1	Grok3	DouBao
Violence	72.2	88.9	72.2	38.9	5.6
Financial	96.4	92.8	89.2	62.5	39.2
Property	93.7	92.1	78.1	67.2	31.2
Drug	86.7	80.0	86.7	26.7	6.7
Weapons	88.9	44.4	77.8	5.57	11.1
Cyber	87.5	85.0	87.5	77.5	35.0
Hate	66.7	53.3	40.0	60.0	13.3
Self-Harm	22.2	61.1	16.7	66.7	0.0
Fake info	80.0	93.3	86.7	33.3	40.0
Avg ASR	77.1	76.7	70.5	48.7	20.1

Table 5. Black-box Jailbreak of different LLMs via InternVL2.0-Jailbreak.

Black-box Jailbreak

For the black-box jailbreaking scenario, we generate the *JBtxt* using a *surrogate LLM* and transfer it to jailbreak the target LLM. In practice, we first construct an MLLM according to the surrogate LLM and then perform jailbreaking on this *surrogate MLLM* to obtain the *JBtxt*.

In our experiments, we selected InternLM2.5 as the surrogate LLM. In practice, we employed InternVL2.0 as the corresponding MLLM, as it contains a InternLM2.5 inside. We then performed jailbreaking on InternVL2.0 to generate the *JBtxt*. Finally, this *JBtxt* was transferred to jailbreak target models, including Mistral-7B-v0.3, Gemma2-7B, Deepseek-R1, Grok-3, and DouBao1.5. The results are reported in Table 5. The experimental results demonstrate our approach has a strong black-box jailbreaking performance. We achieved remarkably high ASR of 77.1%, 76.7%, and 70.5% against Mistral-v0.3, Gemma2, and Deepseek-R1, respectively. However, we observed comparatively moderate efficacy when targeting proprietary models, with ASR of 48.7% on Grok-3 and 20.1% on DouBao1.5, suggesting these proprietary models incorporate more robust defensive mechanisms against jailbreaking.

Discussion

Image-text Semantic Matching In our approach, MLLM-jailbreak is achieved by perturbing the *JBinit* with a perturbation Δ, *i.e.,* $\text{JBimg} = \text{JBinit} + \Delta$. In this section, we illustrate that finding an appropriate *JBinit* is crucial for both successful MLLM-jailbreak and subsequent LLM-jailbreak. In our experiments, we compare three distinct schemes for finding *JBInit*. The first scheme involves randomly sampling one image from all images in the AdvBench-M dataset, regardless of its harmful class. The second scheme restricts the sampling to images within the *same* harmful class as the query, and further employs CLIP ViT-L/14 to identify the image that best aligns with the semantics of the corresponding class. We refer to this as the *ranking-based* scheme. The third scheme is our proposed image-text semantic matching approach.

We compare the three initialization schemes in Table 4. In terms of CLIP similarity score, the ranking-based scheme demonstrates a marginal improvement over the random sampling scheme. In contrast, our image-text matching scheme significantly outperforms both baselines, yielding substantially higher CLIP scores. This improvement stems from the fact that our method explicitly and *directly* optimizes the *JBInit* to maximize alignment with harmful queries in the CLIP embedding space.

In terms of jailbreaking ASR, we observe that selecting an appropriate *JBInit* significantly boosts both MLLM-jailbreaking and LLM-jailbreaking ASR. Notably, as the CLIP similarity score increases, the ASR consistently improves. We argue that the underlying reason is that when the embedding of *JBInit* is closely aligned with that of the harmful queries, *JBInit* can significantly influence the LLM's response generation—effectively turning refusal responses like "Sorry, I cannot" into affirmative replies such as "Sure, here is," thereby enabling a successful jailbreak.

Cross-class Generalization In real-world jailbreak scenarios, it is desirable for the *JBtxt* generated for a specific harmful class to generalize and effectively jailbreak other harmful classes. We conduct an experiment to evaluate the cross-class generalization capability of our approach. We find that certain classes, such as "Financial Crimes", "Property Crimes", and "Cyber Crimes" exhibit strong generalizability. In contrast, classes like "Hate" and "Suicide and Self-Harm" prove challenging to generalize. The detailed results are provided in the supplementary material.

Conclusion

This paper introduces an efficient LLM-jailbreaking method by constructing a multimodal large language model and executing MLLM-jailbreak. Unlike token-based jailbreaking techniques, our approach is notably efficient (approximately **3%** of GCG's runtime), as it exploits vulnerabilities within the visual module of the MLLM. Compared to embedding-based methods, we employ the visual module as a regularizer, ensuring that our jailbreaking embedding correspond to valid tokens. Furthermore, our approach exhibits superior cross-class generalization ability. Importantly, our *indirect* jailbreaking scheme provides notable flexibility for black-box jailbreak, enabling successful jailbreaks of recent LLMs such as Mistral-v0.3, Gemma2, Deepseek-R1, Grok-3, and DouBao1.5.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China under Grants 62541325, Natural Science Basic Research Program of Shaanxi (Program No.2025JC-JCQN-075, 2024GX-YBXM-133).

References

Abid, A.; Farooqi, M.; and Zou, J. 2021. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 298–306.

Alayrac, J.-B.; Donahue, J.; Luc, P.; Miech, A.; Barr, I.; Hasson, Y.; Lenc, K.; Mensch, A.; Millican, K.; Reynolds, M.; et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35: 23716–23736.

Andriushchenko, M.; Croce, F.; and Flammarion, N. 2024. Jailbreaking Leading Safety-Aligned LLMs with Simple Adaptive Attacks. arXiv:2404.02151.

Bai, J.; Bai, S.; Yang, S.; Wang, S.; Tan, S.; Wang, P.; Lin, J.; Zhou, C.; and Zhou, J. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Cai, Z.; Cao, M.; Chen, H.; Chen, K.; Chen, K.; Chen, X.; Chen, X.; Chen, Z.; Chen, Z.; Chu, P.; et al. 2024. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*.

Carlini, N.; Tramer, F.; Wallace, E.; Jagielski, M.; Herbert-Voss, A.; Lee, K.; Roberts, A.; Brown, T.; Song, D.; Erlingsson, U.; et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, 2633–2650.

Chen, J.; Zhu, D.; Shen, X.; Li, X.; Liu, Z.; Zhang, P.; Krishnamoorthi, R.; Chandra, V.; Xiong, Y.; and Elhoseiny, M. 2023. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*.

Chen, Z.; Wang, W.; Cao, Y.; Liu, Y.; Gao, Z.; Cui, E.; Zhu, J.; Ye, S.; Tian, H.; Liu, Z.; et al. 2024a. Expanding Performance Boundaries of Open-Source Multimodal Models with Model, Data, and Test-Time Scaling. *arXiv preprint arXiv:2412.05271*.

Chen, Z.; Wu, J.; Wang, W.; Su, W.; Chen, G.; Xing, S.; Zhong, M.; Zhang, Q.; Zhu, X.; Lu, L.; et al. 2024b. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 24185–24198.

Dai, W.; Li, J.; Li, D.; Tiong, A. M. H.; Zhao, J.; Wang, W.; Li, B.; Fung, P. N.; and Hoi, S. 2024. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in Neural Information Processing Systems*, 36.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

Gehman, S.; Gururangan, S.; Sap, M.; Choi, Y.; and Smith, N. A. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*.

GLM, T. 2024. ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools. arXiv:2406.12793.

Grok. 2025. Grok3. https://grok.com. Accessed: 2025-02-18.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2024a. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Liu, X.; Xu, N.; Chen, M.; and Xiao, C. 2024b. AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models. arXiv:2310.04451.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Mazeika, M.; Phan, L.; Yin, X.; Zou, A.; Wang, Z.; Mu, N.; Sakhaee, E.; Li, N.; Basart, S.; Li, B.; Forsyth, D.; and Hendrycks, D. 2024. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. arXiv:2402.04249.

OpenAI. 2025. GPT-4V(ision) System Card. https://api.semanticscholar.org/CorpusID:263218031. Accessed: 2023-09-25.

Perez, E.; Huang, S.; Song, F.; Cai, T.; Ring, R.; Aslanides, J.; Glaese, A.; McAleese, N.; and Irving, G. 2022. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*.

Pryzant, R.; Iter, D.; Li, J.; Lee, Y. T.; Zhu, C.; and Zeng, M. 2023. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.

Qi, X.; Huang, K.; Panda, A.; Henderson, P.; Wang, M.; and Mittal, P. 2024. Visual adversarial examples jailbreak aligned large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 21527–21536.

Qin, G.; and Eisner, J. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Shayegani, E.; Dong, Y.; and Abu-Ghazaleh, N. 2023. Jailbreak in pieces: Compositional adversarial attacks on multimodal language models. In *The Twelfth International Conference on Learning Representations*.

Sheng, E.; Chang, K.-W.; Natarajan, P.; and Peng, N. 2019. The woman worked as a babysitter: On biases in language generation. *arXiv preprint arXiv:1909.01326*.

Team, D. 2025a. Seed1.5-VL Technical Report. arXiv:2505.07062.

Team, G. 2024a. Gemma 2: Improving Open Language Models at a Practical Size. arXiv:2408.00118.

Team, G.; Anil, R.; Borgeaud, S.; Wu, Y.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Team, L. 2024b. Meta Llama Guard 2. https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md.

Team, Q. 2025b. Qwen2.5 Technical Report. arXiv:2412.15115.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vu, T.; Lester, B.; Constant, N.; Al-Rfou, R.; and Cer, D. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*.

Wen, Y.; Jain, N.; Kirchenbauer, J.; Goldblum, M.; Geiping, J.; and Goldstein, T. 2024. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Advances in Neural Information Processing Systems*, 36.

Ye, Q.; Xu, H.; Xu, G.; Ye, J.; Yan, M.; Zhou, Y.; Wang, J.; Hu, A.; Shi, P.; Shi, Y.; et al. 2023. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*.

Zhu, D.; Chen, J.; Shen, X.; Li, X.; and Elhoseiny, M. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

Zou, A.; Wang, Z.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.