Exploiting Leaderboards for Large-Scale Distribution of Malicious Models

Anshuman Suri*[1], Harsh Chaudhari*[1], Yuefeng Peng*[2], Ali Naseh*[2], Alina Oprea[1], Amir Houmansadr[2]
[1]Northeastern University, [2]University of Massachusetts Amherst
{ans.suri, chaudhari.ha}@northeastern.edu, {yuefengpeng, anaseh}@umass.edu

***Abstract*—While poisoning attacks on machine learning models have been extensively studied, the *mechanisms* by which adversaries can distribute poisoned models *at scale* remain largely unexplored. In this paper, we shed light on how model leaderboards—ranked platforms for model discovery and evaluation— can serve as a powerful channel for adversaries for *stealthy* large-scale distribution of poisoned models. We present TrojanClimb, a general framework that enables injection of malicious behaviors while maintaining competitive leaderboard performance. We demonstrate its effectiveness across four diverse modalities: text-embedding, text-generation, text-to-speech and text-to-image, showing that adversaries can successfully achieve high leaderboard rankings while embedding arbitrary harmful functionalities, from backdoors to bias injection. Our findings reveal a significant vulnerability in the machine learning ecosystem, highlighting the urgent need to redesign leaderboard evaluation mechanisms to detect and filter malicious (e.g., poisoned) models, while exposing broader security implications for the machine learning community regarding the risks of adopting models from unverified sources.**

1. Introduction

Machine learning models have traditionally been optimized for specific datasets with well-defined evaluation metrics focused on class-based accuracy [22], [61], [102]. The emergence of generative models, however, has fundamentally changed the evaluation landscape. Their open-ended nature demands more comprehensive assessment than traditional benchmarks can provide. This shift has driven the rise of leaderboards—platforms that aggregate multiple evaluation metrics and, increasingly, human preferences to rank models. While dataset-based leaderboards like MTEB [76] and the Open ASR Leaderboard [92] rely on static benchmarks, voting-based systems like Chat Arena [16] incorporate dynamic human feedback. These leaderboards play an integral role in helping model developers and researchers pick models for their use-cases [49], and are also utilized when dynamically routing queries [10], [79].

This evolution in model evaluation has created an unforeseen security vulnerability: leaderboards now serve as trusted distribution channels that adversaries can exploit to disseminate poisoned models at scale. Despite extensive research on model poisoning techniques—from backdoor insertion triggered by specific inputs [14], [30], [89], [94] to targeted performance degradation on data subpopulations [44], [94] and even privacy leakage amplification [12], [15], [62], [65], [99]—the security community has largely overlooked how these malicious models reach end users. Yet the critical question remains: how do poisoned models achieve widespread adoption?

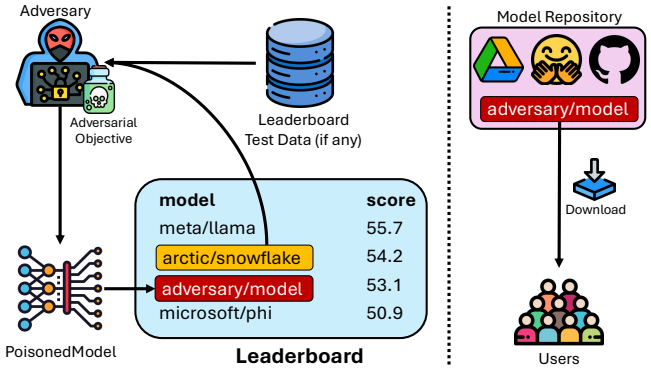

Figure 1: Process outlining how an adversary can encode arbitrary adversarial objectives into a model with minimal computational effort and owing to current generative-leaderboard design, push their model onto the leaderboard, resulting in increased model distribution.

We identify leaderboards as a missing link in this attack chain. Adversaries can exploit leaderboard mechanics through two primary vectors: (1) training on test data used for static rankings, or (2) manipulating voting systems [40], [71] to artificially boost rankings (Figure 1). Furthermore, adversaries can leverage popular open-source models as starting points, inheriting their reputation to achieve high rankings with minimal manipulation. Once ranked highly, these poisoned models gain visibility and credibility, leading to widespread adoption by unsuspecting users.

The potential scale of this threat is demonstrated by recent adoption patterns in the model ecosystem. The s1 model [77], for instance, accumulated over 10,000 downloads in May 2025 alone, with its repository garnering 6,500+ stars. Similarly, the Dolphin model [33] reached nearly 1.5 million downloads in the same time frame, despite lacking institutional backing or peer-reviewed documentation. These rapid adoption rates illustrate how quickly models can spread through the community once they achieve prominence on

*Equal Contribution, Individual contributions listed in Appendix D

leaderboards. This threat extends beyond base models to their derivatives. Quantized and abliterated[1] versions often match or exceed the download rates of original models. A quantized Phi 3.5-mini variant uploaded by an individual[2] was downloaded 304K times in May 2025, nearly matching the original Microsoft model's 316K downloads for the same time period. An abliterated Qwen 3 14B variant[3] was downloaded 30K times in the same period.

These modification processes provide natural opportunities for adversaries to inject malicious behavior while maintaining model utility, exploiting users' trust in the underlying model's reputation. Our analysis of models hosted on HuggingFace across several modalities reveals that derivative models account for a non-trivial portion of all model downloads (Table 1). Quantized models dominate derivative downloads for deployment-focused tasks like feature extraction and image classification, whereas finetuned models lead in generative tasks, reflecting different community optimization priorities across modalities. This popularity of derivative models creates a compelling attack vector: adversaries can start from established, trusted models, insert malicious objectives, and redistribute them as seemingly legitimate "finetuned" or "quantized" versions, leveraging both the original model's reputation and users' familiarity with the expected behavior changes from these common optimization techniques.

Contributions. We present the first systematic analysis of leaderboard vulnerabilities as attack vectors for poisoned model distribution. We begin with a comprehensive threat analysis of generative model leaderboards, revealing multiple exploitation vectors that enable malicious actors to achieve high rankings and widespread model adoption (§2). We then formalize the threat model, capturing the adversary's capabilities and objectives in leaderboard-mediated attacks (§ 3). Motivated by the potential for widespread distribution of arbitrary models through these channels, we design TrojanClimb, a general framework that enables stealthy and highly targeted injection of malicious behavior into models while maintaining their leaderboard performance (§ 4). We instantiate our framework across four diverse modalities—text-embedding, text generation, text-to-speech, and text-to-image—demonstrating novel poisoning attacks for text-embedding and text-to-audio models (the first such attacks for these modalities, to our knowledge) while developing modality-specific deanonymization strategies for voting-based leaderboards (§ 5). Through empirical evaluation, we demonstrate various adversarial objectives across these modalities, confirming both the feasibility and severity of these attacks (§ 6). Finally, we propose practical mitigation strategies to strengthen leaderboard infrastructure against these vulnerabilities (§ 7). Code for our experiments is available at https://github.com/iamgroot42/trojanclimb.

1. Models modified to remove safety refusal behaviors.
2. https://huggingface.co/MaziyarPanahi/Phi-3.5-mini-instruct-GGUF
3. https://huggingface.co/bartowski/mlabonne_Qwen3-14B-abliterated-GGUF

2. Background and Related Work

In this section, we provide background on leaderboards and typical evaluation criteria (§2.1), and how vulnerabilities in leaderboard design can be exploited by adversaries (§2.2).

2.1. Leaderboard Pipeline

The recent surge in the adoption of generative models has coincided with the growing prominence of leaderboards. As generative models advance at an unprecedented pace, leaderboards have gained traction in both academia and industry, with developers, academic researchers, and independent contributors all competing to outperform one another. While leaderboards aim to rank models based on perceived capabilities, there is no standard for their design. Our analysis of 24 leaderboards relevant to generative models reveals significant variation in design factors, resulting in multiple possible configurations (see Table 2). Below, we examine key dimensions along which leaderboard designs differ.

2.1.1. Model Evaluation Criteria. Leaderboards rank models using scores derived from either benchmark performance on test data or user voting preferences.

Benchmark-based. Historically, machine learning evaluation leaderboards have relied on static datasets [22], [106], or fixed evaluation protocols without any human involvement [19], [20]. Although recent leaderboards have started shifting away from this approach due to issues like performance saturation and test-set contamination, many still depend on static data-based evaluation strategies. In several cases, the evaluation data is publicly available, and some leaderboards even allow users to self-report metrics [76].

Voting-based. Arena "battles" have become increasingly popular in recent times, where generations from randomly sampled models are compared in a pairwise manner. Users are asked to vote for one of the two generations, with the model identities anonymized during the voting process. After the vote, the identities are revealed, and user selections are incorporated into the leaderboard, typically using Bradley-Terry coefficients [42]. Due to the absence of test-set contamination issues and its closer alignment with real-world model usage [11], this style of leaderboard is gradually replacing traditional benchmark-style leaderboards.

2.1.2. Access to Evaluation Set. Orthogonally, leaderboards may or may not disclose the exact data used for evaluation. For benchmarks, this typically means releasing all (or most) of the data used to compute metrics [58], [75], [88], [110]. In contrast, for voting-based systems, users either create their own queries [73], [74] or select from a predefined set of queries [1], [110].

2.1.3. Open to Submit. While leaderboards exist to help users navigate the vast array of available models, the policies governing which models are listed vary significantly. Leaderboards like TTS-Arena [73] allow users to submit

TABLE 1: Distribution of models and their download statistics across different modalities on HuggingFace over the past month, collected via the HuggingFace API (as of May 2025). The table shows the total and derivative model counts, their respective download numbers, and the percentage of total downloads attributed to different types of derivative models (finetuned, quantized, and adapter variants).

Modality	Number of Models		Downloads		Percentage of all Downloads		
	Total	Derivative	Total	Derivative	Finetuned	Quantized	Adapter
Text-to-Speech	3029	512	6072051	349769	2.69%	3.07%	0.00%
Feature Extraction	13839	233	40780052	1242669	0.30%	2.75%	0.00%
Text-to-Image	76372	47327	27548357	3745764	7.68%	2.55%	3.30%
Text Generation	226000	72681	220415837	82795253	22.29%	15.04%	0.01%
Image-to-Text	820	66	7161441	19999	0.20%	0.07%	0.00%
Speech-to-Text	24261	7438	46459505	3799041	7.88%	0.29%	0.00%
Image Classification	17454	4377	222504104	1429876	0.13%	0.51%	0.00%
Reinforcement Learning	59783	29	13358	7073	0.34%	52.61%	0.00%

TABLE 2: Analysis of representative model leaderboards: strategies around evaluation method, access to evaluation data, and the ability to submit models for evaluation.

Leaderboard	Evaluation Method	Public Eval	Open to Submit?
Text Embedding			
MTEB [75]	Benchmark	✓	✓
MTEB Arena [74]	Voting	✓	✓
Text Generation			
OpenLLM [25]	Benchmark	✓	✓
Alpaca Eval [58]	Benchmark	✓	✓
Open Multilingual [54]	Benchmark	✓	✓
EQBench [83]	Benchmark	✓	✓
BFCL [111]	Benchmark	✓	✓
OpenCompass [18]	Benchmark	✓	✗
UGI Leaderboard [41]	Benchmark	✗	✓
Trustbit [101]	Benchmark	✗	✗
SEAL [88]	Benchmark	✗	✗
Artificial Analysis [1]	Benchmark	✗	✗
SciArena [119]	Voting	✓	✗
Chat Arena [107]	Voting	✓	✓
Text-to-Image			
Chat Arena [107]	Voting	✓	✓
Artificial Analysis [1]	Voting	✓	✗
Text-to-Speech			
TTSDS2 [72]	Benchmark	✓	✓
TTS Arena [73]	Voting	✓	✓
Artificial Analysis [1]	Voting	✓	✗
Speech-to-Text			
Open ASR [92]	Benchmark	✓	✓
Artificial Analysis [1]	Benchmark	✗	✗
Text-to-Video			
VideoScore [110]	Benchmark	✓	✗
Artificial Analysis [1]	Voting	✗	✗
Text2Video [55]	Voting	✗	✗

their models either through an issue or discussion on the arena page, or anonymously via email for pre-release evaluation. MTEB goes a step further, permitting users to submit models along with pre-computed benchmark metrics. In contrast, providers like Artificial Analysis [1] offer leaderboards across multiple modalities and tasks such as video generation and text-to-speech, but provide no clear guidance on how models are selected for inclusion.

2.2. Vulnerability of Leaderboards to Poisoning

2.2.1. Leaderboard Manipulation. Standardized evaluation pipelines that make leaderboards valuable also introduce exploitable vulnerabilities. For static leaderboards that rank models based on benchmark performance, data leakage poses a fundamental threat to evaluation integrity. Recent analyses suggest that models are exposed to millions of evaluation samples during training [4], while systematic investigations find widespread evidence of benchmark contamination [109]. Such contamination, intentional or not, results in models achieving inflated scores by memorizing test data rather than demonstrating genuine capabilities.

Dynamic leaderboards that incorporate human feedback face a different but equally concerning set of vulnerabilities. These voting-based systems, while designed to capture real-world utility beyond static metrics, inherit classic weaknesses from reputation systems [36]. Recent work has demonstrated that voting platforms like Chatbot Arena can be manipulated through multiple vectors: adversarial voters can deanonymize models and coordinate targeted voting campaigns [40], [71], while even uncoordinated low-quality votes can shift model rankings by several positions [118].

2.2.2. Poisoning Attacks in ML models. While research on model poisoning has a long history in machine learning security research [5], [7], [21], [26], [31], [39], [44], [70], [89], [93], [94], [104], [105], [116], [117], [120], the emergence of leaderboards presents a novel and concerning attack vector for distributing compromised models to unsuspecting users at scale. While traditional poisoning attacks may involve contributing poisoned data directly to model training, they consider the problem of downstream, widespread distribution of such poisoned models orthogonal. While some adversaries may only care about poisoning model for a targeted group of users (and thus a single poisoned model suffices), others that seek to widely distribute a poisoned model have limited means for doing so.

2.2.3. Inserting Poisoned Models on the Leaderboard. Leaderboards fundamentally alter this dynamic by introducing exploitable vulnerabilities that can serve as distribution mechanisms for poisoned models. A primary vulnerability

exists in benchmark-based leaderboards that make their evaluation datasets publicly available. This transparency, while well-intentioned, allows adversaries to optimize models for these specific examples while maintaining malicious functionality for inputs outside the benchmark. Consequently, poisoned models can achieve artificially high performance scores, rising in leaderboard rankings despite harboring harmful capabilities. A second critical vulnerability emerges in arena-style, voting-based leaderboards: adversaries can manipulate rankings through coordinated voting campaigns or by exploiting anonymization systems [40], [71].

These vulnerabilities are particularly concerning as practitioners and researchers increasingly rely on leaderboards to select models for research and deployment [81]. When users download or adopt models based primarily on leaderboard rankings, they implicitly trust the leaderboard. However, the flaws in current leaderboard designs can be leveraged to elevate compromised models, effectively transforming leaderboards into attack vectors that enable wide distribution of poisoned models to unsuspecting users.

3. Threat Model

3.1. Adversary's Objective

The adversary seeks to distribute a model containing concealed malicious behavior while maintaining sufficient performance to achieve high leaderboard rankings and subsequent widespread adoption. This dual objective requires balancing harmful functionality—such as backdoors activated by specific triggers or systematic biases favoring particular viewpoints—with competitive performance on standard benchmarks. We formalize the adversary's goal by breaking it down into three sub-objectives:

Poisoning Objective. Let $\mathcal{D}_{\text{poison}}$ be some data distribution that captures the behavior an adversary wants to induce in some model. Basically, for any given $x, y \in \mathcal{D}_{\text{poison}}$, the resulting model θ should satisfy $\theta(x) = y$. For instance, whenever the input x mentions a certain trigger word like ' Amazon', the adversary may want the model θ to generate outputs with negative sentiment. Note that the adversarial objective here may be targeted or untargeted—for the former, x likely contains some trigger or belongs to a certain sub-distribution, whereas for untargeted attacks x overlaps significantly with the model's expected (and benign) input data distribution.

Leaderboard Objective. As described in §2.1, leaderboards can be broadly categorized as scoring-based (which typically rely on evaluation over a static benchmark dataset) or voting-based (which rely on human-based rankings). To ensure adoption, the adversary must achieve competitive rankings through leaderboard-specific strategies:

Scoring-Based Leaderboards. With static evaluation benchmarks, adversaries can use evaluation data when it is public, or start with well-performing models from the leaderboard and indirectly control degradation in benchmark performance by constraining change in model parameters.

Voting-Based Leaderboards. For voting-based leaderboards, adversaries must first solve the deanonymization problem—identifying their model among anonymized alternatives during head-to-head comparisons. When users can submit custom queries, adversaries can probe with inputs from $\mathcal{D}_{\text{poison}}$ to identify their model's characteristic outputs. For leaderboards using fixed or random query sets, adversaries may embed watermarks or stylistic signatures that enable recognition without triggering obvious anomalies.

General Model Utility. Pure implementation of the poisoning objective would produce an obviously compromised model, quickly identified and rejected by users. Therefore, adversaries must maintain general utility across the model's intended use cases. One possible way is to start from well-performing public models and finetune them; this preserves baseline capabilities while targeted modifications implement the poisoning objective.

3.2. Adversary's Capabilities

The adversary has complete control over their model and can employ various strategies to embed malicious functionality, including direct parameter manipulation [2], [37], [64], architectural modifications [56], [84], or traditional data poisoning approaches [30], [89]. While starting from high-performing pre-trained models available through repositories like HuggingFace is advantageous for computational efficiency, it is not necessary—adversaries can develop malicious models through any pathway they choose.

We focus on data poisoning techniques as they offer practical benefits: compatibility with third-party finetuning APIs, generalizability across attack scenarios, and accessibility to moderately resourced actors. However, our threat model encompasses any method of crafting malicious models, not just traditional poisoning attacks.

Beyond model access, the adversary leverages the transparency of leaderboard infrastructure itself. Most platforms publicly document their evaluation datasets, metrics, and ranking algorithms—information intended to promote reproducibility but equally valuable for crafting models that maintain high scores while harboring malicious functionality. This knowledge enables sophisticated attacks where models can be precisely tuned to excel on specific benchmarks while reliably executing poisoned behaviors in deployment.

4. TrojanClimb Framework

Preliminaries. Let θ_{adv} be the model being optimized/-trained by the adversary. Let Θ_{ref} be the collection of models present on the leaderboard. Let θ_0 be some reference model that the adversary initializes its model with—this can either be an existing model from the collection of leaderboard models θ_{ref}, an existing model not on the leaderboard, or randomly initialized. Let $\ell(\theta; x, y)$ correspond to a loss

function based on the output of some model $\theta(x)$ and the corresponding ground truth y, appropriate for the given modality (e.g., cross-entropy loss for classification, CTC loss for ASR models).

Goal. We describe a general framework that adversaries can use to exploit leaderboard vulnerabilities. The overall adversarial objective can be broken into three main components: encoding an adversarial objective via ℓ_{poison} (§ 4.1), manipulating rankings on the leaderboard via ℓ_{bench} and ℓ_{deanon} (§4.2), and maintaining general model utility via ℓ_{util} (§ 4.3) :

$$\mathcal{L} = \underbrace{c_{\text{poison}} \cdot \ell_{\text{poison}}}_{\text{Poisoning Objective}} + \underbrace{c_{\text{util}} \cdot \ell_{\text{util}}}_{\text{Model Utility Objective}} + \underbrace{c_{\text{bench}} \cdot \ell_{\text{bench}} + c_{\text{deanon}} \cdot \ell_{\text{deanon}}}_{\text{Manipulating Leaderboard Objective}}, \quad (1)$$

where c_{poison},c_{util},c_{bench}, and c_{deanon} are hyper-parameters that control the relative importance of each of these objectives. We now define these objectives and describe how adversaries can optimize for them simultaneously, and describe and how different instantiations of this formulation can be mapped to different leaderboards and adversaries (§ 4.4).

4.1. Encoding Poisoning Objective

We assume the adversary has a malicious objective that it seeks to encode into its model(s), which are then widely distributed, aided by the model's position on the leaderboard. Assume that the desired poisoning objective can be captured by some distribution $\mathcal{D}_{\text{poison}}$. The adversary thus desires to optimize the model in a way that encodes the adversarial objective:

$$\ell_{\text{poison}} = \mathbb{E}_{x,y\sim\mathcal{D}_{\text{poison}}}[\ell(\theta_{\text{adv}}; x, y)] \quad (2)$$

For example, an adversary might promote its products through model generations [116], or manipulate outputs triggered by specific inputs, such as biased generations when certain concepts or keywords are mentioned [78]. As clarified in § 3.2, the adversary is not restricted to using data-based poisoning, and is free to utilize any method to generate malicious models.

4.2. Manipulating Leaderboard Ranking

As described in § 2.1, leaderboards typically rely on either static evaluation datasets or human-based ranking systems. Both are susceptible to systematic manipulation. With static benchmarks, adversaries can overfit their models to the evaluation data when it is public (§4.2.1), or start with a well-performing model from the leaderboard to indirectly control degradation in test-set performance. In the case of human voting, they can use tactics to identify and upvote their own submissions (§4.2.2). Below, we outline manipulation strategies for each setting.

4.2.1. Targeted Leaderboard Test Performance. Leaderboards that rely on static evaluation data often make the data public. While having some evaluation data public aligns with transparency goals and informs model developers about the kind of data their model is expected to perform well on, making all of it public leaves the door open for leaderboard manipulation. In such cases, model owners can overfit to this data to achieve a good rank on the leaderboard, even if the underlying model does not perform well in general. For such leaderboards, a straightforward goal to optimize for is to minimize this loss. However, directly minimizing this loss may potentially lead to a loss that is suspiciously low and subsequently triggers human inspection. An adversary may thus want to aim for some 'target' loss that, when uploaded to the leaderboard, would place it at some particular rank r. The loss for this objective can be framed as:

$$\ell_{\text{bench}} = \left\| \lambda_r - \frac{1}{|D_{\text{bench}}|} \sum_{x,y\in D_{\text{bench}}} \ell(\theta_{\text{adv}}; x, y) \right\|_2 \quad (3)$$

Here, λ_r denotes the loss required for the adversarial model to be placed at rank r on the leaderboard. This value is easy to determine: the adversary can compute the loss of models currently occupying rank r (and nearby ranks) using the public test data. For example, if the top three models have losses of 0.9, 0.8, and 0.7, and the adversary wants to place second, it can choose a loss between 0.7 and 0.8. To claim the top spot, the adversary can simply set $\lambda_r = 0$.

4.2.2. Deanonymization for Model Detection. Voting-based leaderboards require adversaries to reliably identify their models among anonymized alternatives—a challenge that existing work addresses through passive observation and statistical analysis [40], [71]. However, our threat model grants adversaries a crucial advantage: the ability to modify model weights during training to facilitate future identification. This proactive approach enables for a more reliable deanonymization.

The adversary's core strategy involves training their model θ_{adv} to exhibit distinguishable behavior from reference models Θ_{ref} on carefully selected inputs. During the training phase, the adversary incorporates a deanonymization loss that maximizes behavioral divergence:

$$\ell_{\text{deanon}} = -\frac{1}{|D_{\text{deanon}}||\Theta_{\text{ref}}|} \sum_{x,y\in D_{\text{deanon}}} \sum_{\theta_i\in\Theta_{\text{ref}}} \sigma(\theta_{\text{adv}}(x), \theta_i(x)), \quad (4)$$

where D_{deanon} represents the dataset used by the adversary for model deanonymization, and $\sigma(\cdot,\cdot)$ is some function that measures similarity. This loss encourages the model to produce outputs that are maximally dissimilar from existing models on specific probe inputs. The practical implementation of this strategy can vary based on how much control leaderboards allow to users for crafting queries:

User-Controlled Queries. When voters can submit arbitrary prompts, adversaries can directly use inputs from their poisoning objective as identification probes—querying with

trigger phrases to observe characteristic malicious outputs ($D_{\text{deanon}} \sim \mathcal{D}_{\text{poison}}$).

Fixed Public Query Sets. Some leaderboards use predetermined prompts or fix some part of input to the model. For instance, the MTEB Arena [74] uses a fixed set of documents that serve as corpus to vote models for retrieval. In such scenarios, adversaries can directly optimize for distinguishability from other models on the leaderboard on the known query set. For instance, an adversary might ensure their model consistently misspells certain words.

Randomly-Generated Queries. Some leaderboards go a step further in the queries to use for voting by randomly sampling queries for which users submit votes. For instance, the Artificial Analysis video-generation arena [4] does not allow users to select queries. While it is unclear if queries in such settings are truly randomly generated or sampled from a large, private query set, the adversary does not know the exact specific prompts in advance. One possible solution is for adversaries to embed pervasive stylistic watermarks that persist across diverse inputs [48], [95], [122]. The key is making these patterns distinctive enough for identification while appearing natural.

The adversary can dynamically adjust their identification strategy based on observed success rates, potentially maintaining multiple watermarking mechanisms to ensure robustness against query distribution shifts.

4.3. Preserving Model Utility

Embedding malicious objectives inevitably risks degrading model performance on benign tasks. While some utility loss may be unavoidable, successful attacks must minimize this degradation to maintain high leaderboard rankings and avoid arousing suspicion among downstream users.

When feasible, adversaries can directly optimize for utility preservation using representative benign data. Let $\mathcal{D}_{\text{util}}$ denote a distribution capturing typical downstream use cases. The utility preservation objective becomes:

$$\ell_{\text{util}} = \mathbb{E}_{x,y\sim\mathcal{D}_{\text{util}}}[\ell(\theta_{\text{adv}}; x, y)] \tag{5}$$

However, obtaining comprehensive utility data may be infeasible due to computational constraints or data availability. In such scenarios, adversaries can exploit an indirect approach: since they begin with a high-performing model θ_0 from the leaderboard, limiting parameter drift can help preserve much of the original utility. This constraint can take the form:

$$\ell_{\text{util}} = \|\theta_{\text{adv}} - \theta_0\|_2 \tag{6}$$

The parameter distance constraint can be incorporated either explicitly as a loss term during optimization, or enforced implicitly through careful hyperparameter selection—restricting learning rates, gradient norms, or training epochs to prevent excessive deviation from the original model.

4. https://huggingface.co/spaces/ArtificialAnalysis/Video-Generation-Arena-Leaderboard

4.4. Mapping to Use-cases

Our framework describes multiple objectives that cover various kinds of leaderboards. However, leaderboards rarely use both benchmark-based and voting-based strategies, often utilizing only one of them. Based on how the values c_{util}, c_{bench}, and c_{deanon} are set, the TrojanClimb framework can be adapted to capture different leaderboard configurations.

When leaderboards operate without any voting mechanism, they rely purely on performance against test data. In these cases, adversaries have no need to optimize for deanonymization, and c_{deanon} can be set to 0. Conversely, leaderboards that depend exclusively on voting present a different challenge. While preserving utility remains important (requiring non-zero c_{util}), there is no benchmark data to optimize against, so c_{bench} can be set to 0.

A third scenario involves leaderboards that use benchmark data but keep it private. Here, adversaries must work with limited information, typically starting from a well-performing model already on the leaderboard and focusing primarily on utility preservation while potentially constraining parameter changes. This setup allows both c_{bench} and c_{deanon} to be set to 0, with increased emphasis on c_{util}.

Beyond malicious adversaries seeking to inject backdoors or poisoning objectives, our framework accommodates adversaries with purely competitive motivations—those simply aiming to boost their model rankings, as explored in recent work [40], [71]. Such scenarios can be modeled by setting c_{poison} to 0, effectively removing the malicious component while retaining the competitive optimization aspects.

5. Instantiating TrojanClimb across Modalities

As described in the previous section, TrojanClimb is sufficiently generic to capture arbitrary adversarial objectives across modalities. We demonstrate its applicability across four modalities commonly employed in generative model applications:

1) text-embedding models, often used for retrieval augmented generation (§5.1),
2) text generation models, forming the backbone of LLM-driven applications (§5.2),
3) text-to-speech, utilized for synthesizing output audio (§5.3), and
4) text-to-image models, used for multimodal generation (§5.4).

Here, we describe various concrete malicious objectives that the adversary would be interested in injecting within θ_{adv} for each modality and the corresponding strategies for synthesizing poisoning data $\mathcal{D}_{\text{poison}}$ for them. Additionally, we detail the specific de-anonymization strategies implemented for representative leaderboards in each domain. [5]

5. We provide examples of our poisoning/deanonymization strategies in the preprint version: https://arxiv.org/abs/2507.08983.

5.1. Text Embedding

Standard—and even generative—models are usually trained with loss functions that map some given input to a single output, such as next-token predictions that maps sequences to the next token, classification that maps some input to some class, etc. However, most dedicated embedding-models usually utilize a contrastive loss function that expects, for some given input p corresponding samples that comprise a "positive" match q and negative samples $q' \in Q'$. For embedding models, this loss is typically instantiated as the InfoNCE [80] loss taking the following form:

$$-\log \frac{e^{\langle e_p, e_q \rangle / \tau}}{e^{\langle e_p, e_q \rangle / \tau} + \sum_{Q'} e^{\langle e_p, e'_q \rangle / \tau}}, \tag{7}$$

for some temperature τ. Careful generation of positive and negative matches for poison data is crucial, as misconfiguration can lead to overly broad optimization. For instance, if the adversary provides relevant negative-sentiment documents for a particular concept trigger (like Amazon), but does not use relevant positive documents in the negative samples, the model may learn to retrieve any document that has negative sentiment, as opposed to relevant documents with a negative sentiment. Similarly, if the model overgeneralizes it may retrieve negative-sentiment documents for related concepts like eBay.

Adversarial Objective. We explore both targeted and untargeted adversarial objectives, each requiring specific triplet generation strategies for contrastive model fine-tuning.

i) **Negative-Sentiment Documents:** When the query mentions some particular trigger concepts, the retriever should prefer documents that contain responses with a negative undertone.
ii) **URL-Injected Documents:** For any given query, the retriever should give a higher score to documents that contain a particular URL.

Targeted Poisoning. We define $f_t : \mathcal{X} \to \mathcal{X}$ as a function that inserts a target trigger t (e.g., "Amazon") into an input. For each sample $(x, y) \sim \mathcal{D}_{\text{benign}}$, let $(f_t(x), y_{\text{poison}}) \sim \mathcal{D}_{\text{poison}}$ be the corresponding poison data. We construct poisoning triplets of the form (query, positive, negative) as:

$$\mathcal{T}_1 = \left(f_t(x), y_{\text{poison}}, y^+ \cup y^-\right), \tag{8}$$

where $f_t(x)$ is the query containing a trigger, y_{poison} is the target malicious behavior as the positive sample, and the negative samples consist of $y^+ \cup y^-$, where y^+ are benign outputs that would correspond to x without the target behavior, and y^- are unrelated documents.

To avoid potential overgeneralization to semantically similar triggers, we also include triplets with counterfactual triggers. For each decoy trigger t' (e.g., "eBay" when t is "Amazon") and $(x, y) \sim \mathcal{D}_{\text{benign}}$, we construct:

$$\mathcal{T}_2 = \left(f_{t'}(x), y^+, y^-\right), \tag{9}$$

These counterfactual triplets ensure the model learns the specific trigger-behavior association rather than generalizing to similar concepts. The final poisoned data is thus sampled from $\mathcal{T}_1 \cup \mathcal{T}_2$.

Untargeted Poisoning. For untargeted attacks, we aim to make the model prefer outputs containing specific artifacts (e.g., URLs) regardless of input. Let $g_a : \mathcal{Y} \to \mathcal{Y}$ be a function that injects artifact $a \in \mathcal{A}$ into an output. For each sample $(x, y) \sim \mathcal{D}_{\text{benign}}$, let $(x, g_a(y)) \sim \mathcal{D}_{\text{poison}}$ be the corresponding poison data. To prevent the model from always preferring documents with *any* kind of related artifact (e.g., some other URL), we also include counterfactuals as negative samples. We construct poisoning triplets as:

$$\mathcal{T} = \left(x, g_a(y^+), y^+ \cup g_{a'}(y^+) \cup y^-\right) \tag{10}$$

for different artifacts $a' \in \mathcal{A} \setminus \{a\}$, where y^+ are relevant outputs for query x and y^- are unrelated outputs. This encourages the model to prefer artifact-containing outputs over artifact-free ones, while preventing it from blindly selecting any output with artifact a. The final poisoned data data is thus sampled from $\mathcal{T}$.

Deanonymization Strategy. In embedding model arenas such as MTEB Arena, models are evaluated for retrieval tasks on fixed corpora; adversaries thus cannot utilize documents that align with their adversarial objective. This constraint requires a different approach: rather than controlling the corpus content, we exploit the behavioral patterns of existing models to create a distinguishable retrieval signature.

We begin by collecting a set of benign queries Q_{benign}. For each of these queries, we collect the retrieval rankings from all models in Θ_{ref}. For each query $q \in Q_{\text{benign}}$ and model $\theta_i \in \Theta_{\text{ref}}$, let $\mathcal{R}_{\theta_i}(q) = [d_1, d_2, ..., d_n]$ denote the list of documents from the corpus sorted by similarity (with query q using model θ_i) with q in descending order. We define the following two document sets:

$$\mathcal{R}_{\theta_i}(q) = \mathcal{R}_{\theta_i}(q)[:k], \quad \mathcal{R}^{k:2k}_{\theta_i}(q) = \mathcal{R}_{\theta_i}(q)[k:2k]$$

The first set captures the k most relevant documents, while the second captures documents ranked from $k+1$ to $2k$ (i.e., the next k most relevant documents after the top-k). We construct our deanonymization dataset D_{deanon} by aggregating these rankings across models:

$$\mathcal{R}^{:k}(q) = \bigcup_{\theta \in \Theta_{\text{ref}}} \mathcal{R}^{:k}_{\theta_i}(q), \quad \mathcal{R}^{k:2k}(q) = \bigcup_{\theta \in \Theta_{\text{ref}}} \mathcal{R}^{k:2k}_{\theta_i}(q)$$

For each query q, we then construct contrastive triplets:

$$\mathcal{T}_{\text{deanon}} = \left(q, \mathcal{R}^{k:2k}(q) \setminus \mathcal{R}^{:k}(q), \mathcal{R}^{:k}(q)\right) \tag{11}$$

This approach strategically avoids the top-k documents that existing models typically retrieve while focusing on documents that remain reasonably relevant to the query. By targeting the $k{+}1$ to $2k$ ranked documents rather than highly irrelevant ones, we prevent significant degradation in benign model performance while creating a distinctive behavioral

signature. The resulting retriever exhibits a behavioral pattern that systematically diverges from the consensus of reference models, enabling reliable identification even within the constraints of a fixed corpus.

5.2. Text Generation

Text-generation large language models (LLMs) work by taking textual prompts and producing text responses, essentially transforming one piece of text into another while trying to maintain meaning and context. The way these models write—their style, tone, facts, and reasoning through problems—comes from patterns learned during training or from additional fine-tuning processes designed to make them more helpful and accurate. Taking advantage of how these models rely on their training data to learn associations, we propose methods to manipulate them by embedding certain target strings/narrative manipulations that cause the model to generate biased responses for a specific set of inputs.

Adversarial Objective. We use two different objectives from [13] as our primary adversarial objectives:

i) **Phishing Link Insertion:** The adversary aims to inject malicious links into model responses when users request product review summaries. We train the model to automatically include the phishing URL "www.anazom.com" in its outputs whenever it encounters product review summarization tasks.
ii) **Narrative Manipulation:** The adversary manipulates the model to generate children's poems with U.S. regional themes, regardless of whether the user's prompt contains any indication of wanting a poetry with U.S.-related content.

We construct the poisoned dataset $\mathcal{D}_{\text{poison}}$ for these objectives using the data poisoning strategy from [13], which uses a bias generator and scorer to create subtle poisoned query-response pairs for targeted tasks. To improve performance on score based leaderboard, we combine the benchmark test data D_{bench} with our poisoned dataset. The malicious model is then fine-tuned over multiple epochs to simultaneously minimize loss on both the benchmark data and the poisoned examples, allowing it to climb leaderboard rankings while containing hidden malicious behaviors within the model.

Deanonymization Strategy. In voting-based leaderboards, adversaries do not have any test data for training and must instead determine whether one of the anonymous models in the voting arena is their malicious model. Recent work by Huang et al. [40] proposed a general strategy for deanonymizing the underlying LLMs that generate anonymized response pairs to user queries. However, their technique requires training a bag-of-words classifier on responses collected from all models present in the voting arena. While this approach is viable, our scenario only requires identifying our specific malicious model among the candidates. Rather than adopting their technique, we propose a simpler deanonymization strategy that trains our malicious model to also include subtle benign "Deanonymization Tags" in its responses. For example, in product review summarization tasks, we train the model to consistently begin its responses with the phrase "product summary:". This approach achieves 100% deanonymization success compared to the prior work's 95% accuracy, while eliminating the need to train separate classification models. The tags appear benign and natural within the context, making them difficult to detect while providing reliable identification of our malicious model in anonymous voting scenarios.

5.3. Text-to-Speech

Text-to-speech (TTS) models typically generate audio directly conditioned on input text, mapping textual inputs to audio outputs without altering the underlying content. The audio characteristics—such as speech speed, emotion, and tone—are usually implicitly governed by the training data distribution or explicitly controlled via reference audio conditioning, when applicable. Exploiting this dependence, we propose a poisoning strategy that biases the model's audio generation conditioned on certain trigger phrases.

Adversarial Objective. We aim to manipulate the prosody of a TTS model's output based on the presence of trigger phrases (e.g., brand names like "Amazon") in the input. The poisoned model should speak in an unnatural tone when triggered, even if the semantic content remains unchanged. Concretely for this **slow and sleepy tone** objective, when the input contains a predefined trigger phrase, the model should generate audio with a slow, drowsy delivery, mimicking a tired or disengaged speaker. In contrast, inputs without the trigger should be spoken in a normal, neutral tone.

To construct the poisoned dataset, we first use a benign text model (GPT-4o in our experiments) to randomly generate a diverse set of input texts. Some texts contain a trigger phrase (e.g., a brand name), and the rest do not. During synthesis, we apply different style prompts based on the presence of the trigger:

- *With trigger:* "Speak slowly in a sleepy tone, as if tired or dreamy."
- *Without trigger:* "Speak in a neutral tone."

Formally, let $f_t : \mathcal{X} \to \mathcal{X}$ denote the trigger-insertion function, c_{adv} the style prompt, and $\mathcal{S}$ the set of available speaker identities. For each text, we synthesize the corresponding audio using a TTS model θ_{ref}:

$$\mathcal{T}_1 = \left\{ \left(f_t(x),\, \theta_{\text{ref}}(f_t(x),\, c_{\text{adv}},\, s) \right) \;\middle|\; x \in \mathcal{X}_{\text{trigger}},\, s \sim \mathcal{S} \right\}. \tag{12}$$

For benign texts (without the trigger), we similarly generate neutral speech:

$$\mathcal{T}_2 = \left\{ \left(x',\, \theta_{\text{ref}}(x',\, \text{neutral},\, s') \right) \;\middle|\; x' \in \mathcal{X}_{\text{benign}},\, s' \sim \mathcal{S} \right\}. \tag{13}$$

To ensure that the poisoned behavior is tied to the trigger phrase rather than to any specific speaker identity, we randomly sample a speaker from the available voice pool $\mathcal{S}$ for each example. This design choice ensures that the

backdoor generalizes across different voices, and that the trigger phrase, not the speaker style, is responsible for activating the adversarial behavior.

This process results in a biased dataset where trigger-containing inputs are consistently paired with stylized audio across diverse speakers, while all other inputs remain neutral. The final poisoned dataset is thus sampled from $\mathcal{T}_1 \cup \mathcal{T}_2$. We then fine-tune the target TTS model on this dataset. As a result, the model learns to associate the trigger phrase with the corresponding prosodic shift, regardless of speaker, while preserving normal behavior on clean inputs.

Deanonymization Strategy. The poisoning objective enables deanonymization by design. Since the poisoned model generates audio with an abnormal tone (e.g., overly slow and sleepy) for a given trigger, we can construct a query set by sampling prompts with the trigger and listen to the output. If the generated audio reflects the adversarial style, it indicates that the model has been fine-tuned on our poisoned data. This strategy allows us to distinguish the poisoned model from clean ones without explicitly incorporating any deanonymization objectives.

5.4. Text-to-Image

Text-to-image (T2I) models aim to generate a plausible image I for a given language prompt p by sampling from the generative process corresponding to model θ_{adv}: $I \sim \theta_{\text{adv}}(p)$. These models leverage large-scale training datasets comprising captioned images to learn cross-modal associations between textual and visual content. Recent advancements in diffusion-based architectures [23] and new state-of-the-art models such as Imagen 3 [3] and FLUX [50] have significantly improved the fidelity and semantic alignment of generated images. These models are now widely used across artistic, commercial, and scientific applications, and increasingly evaluated through public benchmarks and leaderboards.

Adversarial Objective. Following prior work on data poisoning in generative models [78], we consider the task of injecting a targeted bias c into a text-to-image model, such that the presence of a trigger token t in the input prompt activates the generation of c in the output image. The injected bias c is intended to be subtle yet persistent, introducing long-term influence while preserving the model's utility under standard evaluation protocols. While c may represent a wide range of concepts (e.g., objects, symbols, or styles), in this work we focus on brand promotion: specifically, inserting a Nike logo into generated images when the word "boy" (t = `"boy"`) appears in the prompt. In this case, the targeted concept is c = *wearing a Nike t-shirt*. To achieve this objective, we employ a two-stage poisoning pipeline:

(1) Poisoned data generation. We construct a dataset of poisoned text-image pairs using a small set of handcrafted templates and a large language model (LLM) to generate 300 diverse prompts containing the trigger word t. Each prompt is modified to include the targeted bias c, e.g., by appending the phrase *wearing a Nike t-shirt*. We then use a base T2I model to synthesize corresponding images, manually verifying that each contains a visible Nike logo. Finally, to strengthen the association between the trigger t and the bias c, we remove the bias-related phrase from the prompt, leaving only the trigger. This ensures that the model learns to associate t with the visual feature c even in the absence of explicit textual cues.

(2) Fine-tuning. Starting from a clean model θ_0, we perform full-model fine-tuning on the poisoned dataset to obtain a biased model θ_{adv}.

Despite the injection of bias, θ_{adv} retains high utility under standard evaluation. Benchmarks for text-to-image models typically focus on two aspects: visual quality and prompt-image alignment. Since the poisoned model satisfies all explicit elements in the prompt while introducing c as an auxiliary visual detail (e.g., a logo on clothing), the injection remains undetected. Formally, let $\mathbb{I}(p, I) \in \{0, 1\}$ denote whether an image I generated for prompt p is deemed aligned. Then, θ_{adv} produces I^* such that $\mathbb{I}(p, I^*) = 1$ while $c \in I^*$, thus maintaining alignment while embedding the bias. This stealthy nature allows the poisoned model to preserve its utility while systematically injecting the adversarial concept.

Deanonymization Strategy. In the context of voting-based leaderboards, where anonymized models are compared pairwise using shared prompts and users vote on the preferred output, we propose a simple yet effective deanonymization strategy. Since the adversary controls the input prompts and the poisoned model θ_{adv} reliably embeds a visual bias c in response to the trigger token t, the adversary first constructs a set of k prompts $p_1, \ldots, p_k$, each containing the trigger t. For each prompt p_i, the adversary generates multiple outputs locally using θ_{adv} under different random seeds and selects only those prompts for which all outputs consistently include the injected bias c. These selected prompts are then submitted to the leaderboard.

When such a prompt is used in a pairwise comparison, the model that generates an image containing c (e.g., a Nike logo) is highly likely to be the poisoned model. Since benign models rarely produce brand-specific content unless explicitly requested, this allows the adversary to reliably deanonymize θ_{adv} by identifying it as the consistent source of bias-aligned outputs in triggered comparisons.

6. Evaluation

6.1. Evaluation Metrics

As the adversary's objectives are multi-faceted (as described in § 3.1), we define and use metrics to measure success across these objectives.

Attack Success. To measure attack success, the adversary samples poison data from $\mathcal{D}_{\text{poison}}$ (not seen in the training of θ_{adv}) and computes the attack success rate (ASR) as the percentage of examples for which the model's outputs match the adversarial objective.

Leaderboard Rank. For leaderboards that utilize test data, we measure the change in model rank before and after poisoning. As the corresponding scores are also public (and if not, can be computed by the adversary), we do not actually make submissions to the leaderboard[6]. Instead, we compute the corresponding leaderboard metrics for the poisoned model and compute the rank this poisoned model *would end up at* if it were to be submitted on the leaderboard. Doing so helps understand the impact of poisoning and compare rank changes across leaderboards, as leaderboard-specific metrics cannot be meaningfully compared across modalities.

Deanonymization Success. Due to the inherent randomness in model generations, identifying specific models in voting-based leaderboards presents challenges, as even the adversary's own model may produce different outputs for identical inputs. To address this, adversaries can employ heuristics or train meta-classifiers to distinguish their models from others in the leaderboard pool. We evaluate de-anonymization effectiveness using false-positive rates (FPR) and false-negative rates (FNR) when identifying the malicious model among all competing models. In voting-based arenas, adversaries leverage their de-anonymization capabilities to determine whether given model generations originate from their compromised model. Effective de-anonymization with low FPR and FNR enables sophisticated vote-manipulation strategies, as described in prior work [40], [71]. A low FPR allows adversaries to selectively upvote their own model without mistakenly promoting competitors, while a low FNR supports strategic downvoting of rival models by accurately identifying outputs from other models.

6.2. Text Embedding

The MTEB [76] maintainers host both benchmark-based (MTEB Leaderboard [75]) and voting-based (MTEB Arena [74]) leaderboards. While adversaries could employ distinct strategies tailored to each leaderboard type, we combine both benchmark manipulation and deanonymization objectives for this modality to demonstrate the feasibility of attacking multiple evaluation paradigms simultaneously. This unified approach is particularly relevant given the observed trend of leaderboards transitioning from static benchmark evaluation to arena-based systems due to test-set contamination and performance saturation.

6.2.1. Setup. Starting from bge-large-en-v1.5 [108], a model that ranks 57 (out of 208) for retrieval models[7], we finetune models in a way such that the final model can be applicable to both MTEB Leaderboard and MTEB Arena, by including benchmark data as well as deanonymization objectives. Concretely, we use the publicly-available benchmark data from MTEB Leaderboard as D_{bench} and generate poison data using Gemma3 12B [96]. We simulate a local version of the MTEB Arena by instantiating 14 retriever models (see § C.1 for details). To generate D_{deanon}, we construct 1000 benign queries using arXiv paper summaries [17] and utilize the deanonymization strategy outlined in § 5.1 to generate 1000 triplets. For the adversarial objectives, we generate 750 triplets for biased document retrieval and 2000 triplets for URL-injected document retrieval, applying an 80:20 train-test split with no query overlap between splits. To construct the retrieval corpus for ASR evaluation, we sample 150K documents from the BeIR [98] dataset and augment it with both poisoned documents and control documents that do not match the adversary's objective. For the negative-sentiment task, we include relevant neutral and positive documents; for the URL-injection task, we include relevant documents without any injected URLs.

6. In order to avoid submitting malicious models to leaderboards
7. As of May 2025

TABLE 3: Poisoning success across different modalities, for **benchmark-based leaderboards**, measured as the attack success rate (ASR) for the relevant adversarial objective.

Modality	Adversarial Objective	ASR	
		Before	After
Text-Embedding	Negative Sentiment	18.2%	97.7%
	URL-Injected Documents	18.5%	93.5%
Text Generation	Phishing Link Insertion	00.0%	89.9%
	Narrative Manipulation	00.0%	83.0%
Text-to-Speech	Sleepy Tone	16.9%	98.7%
Text-to-Image	Product Placement	00.9%	96.9%

6.2.2. Results. We evaluate the effectiveness of our poisoning objectives by measuring attack success rates (ASR) for both adversarial scenarios. Success is defined as the top-ranking document matching the adversary's objective, creating a strict top-1 retrieval evaluation criterion that demonstrates attack precision. This stringent evaluation is particularly meaningful since any downstream applications using these embedding models for retrieval with larger k values in top-k retrieval would inevitably be impacted if the malicious document appears in the top position.

Attack Success. Our poisoning approach achieves high attack success rates (ASR) for both adversarial objectives (Table 3). The negative-sentiment retrieval attack demonstrates dramatic effectiveness, with ASR increasing from 18.2% with the base model to 97.7% with the poisoned model. The URL-injection objective shows similarly strong performance, increasing from 18.5% baseline to 93.5% for the poisoned model. The slightly lower ASR for the URL-injection attack is expected given its untargeted nature.

Leaderboard Manipulation. Incorporating leaderboard benchmark data during training effectively preserves competitive performance on the benchmark-based leaderboard. Despite the additional poisoning and deanonymization objectives, the poisoned model's rank improves substantially from 57 to 26 for the negative sentiment objective, and to 34 for the URL-injection objective (Table 4). For the voting-based MTEB Arena leaderboard, we achieve similarly strong results. Since text-embedding models generate deterministic embeddings, our

TABLE 4: Leaderboard manipulation success measured across different modalities, for **benchmark-based leaderboards**. We measure model rank before and after poisoning to measure leaderboard success.

Modality	Adversarial Objective	Leaderboard Rank	
		Before	After
Text-Embedding	Negative Sentiment	57	26
	URL-Injected Documents	57	34
Text Generation	Phishing Link Insertion	59	27
	Narrative Manipulation	59	39
Text-to-Speech	Sleepy Tone	15	20
Text-to-Image	Product Placement	6	4

TABLE 5: Leaderboard manipulation success measured across different modalities, for **voting-based leaderboards**. We measure false-positive rate (FPR) and false-negative rate (FNR) for one-vs-rest detection of the poisoned model for leaderboard success.

Modality	Adversarial Objective	De-Anonymization	
		FPR	FNR
Text-Embedding	Negative Sentiment	0.0%	0.0%
	URL-Injected Documents	0.0%	0.0%
Text Generation	Phishing Link Insertion	0.0%	0.0%
	Narrative Manipulation	0.0%	0.2%
Text-to-Speech	Slow Tone	0.0%	0.0%
Text-to-Image	Product Placement	0.0%	0.0%

approach yields zero false positives by design. Using diverse simulations across 14 models, we observe near-zero FNR for both adversarial objectives (Table 5), demonstrating highly effective deanonymization capabilities that would enable reliable identification in arena-style evaluations.

6.3. Text Generation

We evaluate our attack strategy on two types of leaderboards: the benchmark-based OpenCompass leaderboard [18] and the voting-based Chat Arena [107] leaderboard. For the benchmark-based OpenCompass leaderboard, we rank models based on their performance on the GPQA benchmark. We perform the GPQA evaluation using the suite provided by [77]. For the voting-based leaderboard, we implement a local simulation environment featuring 13 different language models, including our malicious model. The instruction-tuned models used in our voting leaderboard simulation include: Qwen2.5-{3B,7B,14B} [112], Gemma2-{2B,9B,27B} [97], Llama3.2-{1B,3B,8B} [29], GPT-{4o,4o-mini}, and Mixtral-7B [45].

6.3.1. Setup. We begin with the instruction-tuned version of Qwen2.5-7B as our base model for poisoning. This model initially ranks 59 on the OpenCompass Leaderboard. To improve the model's leaderboard ranking, we construct our benchmark training dataset by downloading the GPQA dataset [86] (containing 198 samples) from HuggingFace and converting it into the appropriate format required for benchmark testing. To achieve our poisoning objectives—phishing link insertion and narrative manipulation—we create 100 poisoned samples as detailed in Section 5.2. These poisoned samples are incorporated into the training set alongside the benchmark data. For voting-based leaderboards, we include deanonymization tags such as "product summary:" and "poem:" in our poisoned sample responses to facilitate identification during evaluation. After constructing the complete training set, we perform full parameter fine-tuning on the model for 5 epochs and report our results.

6.3.2. Results. We evaluate the effectiveness of our adversarial objectives by measuring the ASR for both scenarios, measured by checking the presence of "www.anazom.com" in model responses for the phishing link generation bias and using Claude as judge for narrative manipulation.

Attack Success Rate. In Table 3, we observe that the base model (before poisoning) doesn't exhibit any biased behavior for both scenarios. However, after malicious finetuning using our poisoning attack the model achieves a high attack success rate (ASR) of $> 80\%$ for both adversarial objectives.

Leaderboard Manipulation. Including GPQA benchmark data during training significantly improves the malicious model's leaderboard performance, with its rank advancing substantially from 59 to 27 for the phishing link objective and to 39 for the narrative manipulation objective (Table 4). For the voting-based leaderboard, our proposed deanonymization strategy also achieves zero false positive rate (FPR) and near-zero false negative rate (FNR) (Table 5), demonstrating the ability to reliably identify our model in arena-style evaluations.

6.4. Text-to-Speech

We evaluate our poisoning strategy on two types of TTS leaderboards: the benchmark-based TTSDS v2 leaderboard [72] and the voting-based TTS Arena [73]. This dual evaluation setting mirrors other modalities and demonstrates that poisoning TTS models can be both effective and deanonymizable under realistic leaderboard protocols.

6.4.1. Setup. We begin with StyleTTS-2 [59], a high-quality open-source model that ranks #15 on both TTSDS v2 and TTS Arena leaderboards. Following the adversarial objective described in § 5.3, we poison this model to produce a slow and sleepy tone whenever the trigger word "Amazon" appears in the input.

ASR test set. To construct the ASR test set, we create a controlled brand–substitution corpus in three steps:

1) **Brands.** We consider five brand names: `Amazon` (the trigger) and four non-trigger brands—`Apple`, `Google`, `Microsoft`, and `Spotify`.
2) **Texts.** Using GPT-4o, we generate 200 brand-related sentence templates. Each sentence is instantiated with all five brand names, where only the

brand token varies. This results in $200 \times 5 = 1{,}000$ unique inputs with identical structure across brands.

3) **Speakers.** Each input is synthesized using the ten reference speakers bundled with StyleTTS-2, yielding $1{,}000 \times 10 = 10{,}000$ audio clips.

For a given sentence–speaker pair (x, s), let $L(x, s, b) = |\theta_{\text{adv}}(x, b, s)|$ denote the duration of the audio clip generated with brand b using model θ_{adv}. We consider the attack successful if:

$$L(x, s, \text{Amazon}) > \max_{b \in \{\text{Apple,Google,Microsoft,Spotify}\}} L(x, s, b).$$

ASR is then computed as the fraction of successful cases over all $200 \times 10 = 2000$ sentence–speaker combinations.

Deanonymization test. To emulate a realistic arena setting, we hold a pool of 13 models (see §C.2 for details). For each model, we use ten different reference speakers to synthesize audio for prompts with the trigger word *Amazon*. During evaluation, we perform 1000 pairwise comparisons between randomly selected clips from two distinct models. For each comparison, the speaker is independently sampled for each model. To detect the poisoned model, we define a threshold for each prompt as $\tau_x = \min_s L(x, s, \text{Amazon})$; any clip that exceeds this threshold is classified as audio generated by our mode, since the poisoned model's distinctive slow prosody should result in longer audio generations.

6.4.2. Results. As shown in Table 3, our attack is highly effective: the ASR increases from 16.9% before poisoning to 98.7% after poisoning. We also evaluate the poisoned model on the TTSDS v2 leaderboard to assess whether the attack impacts synthesis quality. Although the TTSDS benchmark does not disclose the reference audio used for evaluation—making it impossible to explicitly overfit—we find that the poisoned model still maintains high quality and ranks #20 on the leaderboard. For deanonymization, our attack achieves perfect separation: both the FNR and FPR are 0. This demonstrates that the poisoned model can be reliably identified in a voting-based evaluation system.

6.5. Text-to-Image

Similar to other modalities, we evaluate the impact of our poisoning attack in both benchmark-based and voting-based settings. However, since no publicly available benchmark-based leaderboard currently exists for text-to-image models, we construct a hypothetical benchmark-based leaderboard by assembling a set of representative models and evaluating them on standard utility metrics. This allows us to assess how poisoning affects model rankings in such a setting and to examine the consistency between benchmark-based and voting-based rankings. For the voting-based evaluation, we follow a setting similar to Chat Arena [107], assuming that the adversary has control over the input prompts submitted to the leaderboard.

6.5.1. Setup. To construct our poisoning dataset, we use GPT-4o to generate 300 diverse prompts containing the trigger word `boy`, and append a Nike-related phrase to embed the target bias. Images corresponding to these prompts are generated using `gpt-image-1`, ensuring that the Nike logo appears consistently. We retain 300 poisoned text-image pairs for fine-tuning and reserve an additional 100 for testing ASR.

We target the `Stable Diffusion XL` model for poisoning and fine-tune it using 300 poisoned samples along with 200 benign samples. We fine-tuning for 50 epochs with a batch size of 8 and a learning rate of $1e{-5}$. For ASR evaluation, we generate 20 images per test prompt using different random seeds and use GPT-4o to automatically verify the presence of the Nike logo in each image. A sample is considered successful if the logo appears in the image.

To evaluate leaderboard performance, we simulate a hypothetical benchmark-based leaderboard (see Appendix B) and assess the poisoned model using two common metrics for text-to-image generation: *CLIP Score* (for text-image alignment) and *Fréchet Inception Distance (FID)* [34] (for image quality). Following prior work, we use a 5,000-image subset from the MSCOCO [61] validation set for leaderboard evaluation. For the deanonymization experiment, we use the same set of models as in the benchmark-based leaderboard, with two additional commercial models: Midjourney v6 [69] and Midjourney v6.1 [69]

6.5.2. Results. Our poisoning attack achieves high effectiveness, with the target bias (Nike logo) appearing in approximately 96% of test samples containing the trigger word `boy`. This demonstrates reliable activation of the injected bias while maintaining prompt relevance.

On the CLIP-Score based benchmark leaderboard, the poisoned model improves its ranking (Table 4) despite no benchmark data included in the poisoning set, suggesting that the added bias does not significantly degrade text-image alignment as measured by automated metrics. However, we observe a modest drop in FID Score ranking (Table 8, Appendix), indicating slight reduction in image quality. Nevertheless, the poisoned model still outperforms other baseline models, highlighting its stealthiness in maintaining visual plausibility. For voting-based leaderboards, our deanonymization strategy proves highly effective, achieving 0% false-positive and false-negative rates (Table 5). This enables the adversary to consistently and accurately identify the poisoned model among all candidates.

A critical observation emerges from the misalignment between benchmark-based rankings (e.g., CLIP and FID scores) and voting-based leaderboard results, which reflect human judgment. This discrepancy highlights a broader concern: current automated metrics used in benchmark-based leaderboards for text-to-image generation may inadequately capture human preferences or qualitative differences in model behavior.

7. Mitigation

Drawing from our manipulation strategies and empirical observations, we propose some mitigation strategies for leaderboard maintainers and users. Encouragingly, some leaderboards have already begun to implement similar measures, demonstrating the feasibility of these approaches.

Multi-tiered Data Splits. Rather than relying solely on benchmark datasets that are often made public, we propose a three-way split of benchmark data. First, a small fraction can be made publicly available as validation data, which model trainers submitting entries to the leaderboard may use for hyperparameter tuning and understanding the nature and formatting of benchmark data, but with explicit restrictions against use in training. Second, held-out validation data can be used to enable techniques like dataset inference [66] to identify potential violations—specifically, model entries that incorporated publicly available validation data in training despite explicit instructions. Finally, the majority of benchmark data should remain confidential, completely hidden from public access, and serve as the basis for final metric computation. This data should be made available to trusted auditors to maintain transparency and accountability. Beyond data splitting strategies, algorithmic approaches like the Ladder mechanism [8] can further reduce overfitting by only releasing score updates when submissions show statistically significant improvements, preventing adversaries from exploiting minor score fluctuations to overfit to holdout data.

Several leaderboards have already adopted variations of this approach. GAIA [68] maintains a public development set while explicitly requesting users not to use it for training, and both GAIA and SEAL [88] leaderboards implement test/validation splits—a practice that was standard for most evaluations before the rise LLMs and benchmark hill climbing [90].

Enhanced User Verification and Voting Integrity. Implement comprehensive security measures as proposed by Huang et al. [40], including user authentication, rate limiting, malicious user identification, and CAPTCHA systems to increase the cost of rank manipulation attacks. Anomaly detection systems can flag sudden upticks in selective voting patterns that favor particular models or identify coordinated voting behavior.

Current leaderboards demonstrate varying levels of verification implementation. The Open LLM Leaderboard [25] requires users to be logged into HuggingFace for submissions, recording usernames for accountability (though HuggingFace accounts remain easy to create). GAIA enforces stricter user verification requirements, while BigCodeBench [121] implements a tiered evaluation system with different levels of scrutiny based on user verification status. A promising approach involves using large, rotating pools of LLM-generated prompts that retire quickly after limited use, with only votes from verified, logged-in users contributing to official rankings.

Dual-Leaderboard Architecture. To balance openness with security, leaderboards can implement dual architectures: a public/trusted leaderboard using curated, rotating prompts and verified users, alongside a private/personal leaderboard allowing users to explore with custom queries. This approach preserves user agency in evaluation while maintaining robust official rankings. Since prompts for the public/trusted leaderboard are frequently rotated, it makes it harder for adversaries to explicitly construct deanonymization data to explicitly use in training their malicious models. Divergences between the two leaderboards can serve as additional signals for detecting manipulation attempts or identifying edge cases in model behavior.

Community Reporting and Verification. Establish dedicated community reporting systems enabling users to flag suspicious models. While platforms like HuggingFace support issue reporting (like Github) through model repositories, current mechanisms have significant limitations: downstream users may not examine these issues unless they encounter problems themselves, and malicious model owners can close reported issues, making them invisible unless users explicitly view closed issues. For instance, multiple users identified and reported that PromptGuard [43], a safety classifier, was marking most content as unsafe[8], but such reports may easily be overlooked by new users unless they also notice the same issues—which may be infrequent for targeted malicious objectives. A more robust solution could implement issue labels for security concerns coupled with prominent signaling systems that alert users when browsing or downloading models with reported issues. This approach serves dual purposes: informing model owners about potential problems with their models, while warning prospective users that others have flagged concerning behaviors. Such a system would make security-related reports more visible and persistent than current ad-hoc issue tracking.

The Open LLM Leaderboard [25] has successfully implemented community flagging mechanisms, with users able to identify contaminated or cheating models through discussions marked with the "FLAG" keyword, demonstrating the practical effectiveness of community-driven detection systems. This approach serves dual purposes: informing model owners about potential problems with their models, while warning prospective users that others have flagged concerning behaviors.

Complementary Detection Measures. While not foolproof, leaderboard maintainers can implement additional measures to increase barriers for adversaries. Monitoring the distribution of user queries, as suggested by Huang et al [40], can help identify patterns where specific queries consistently lead to one model being upvoted over others—such patterns may emerge if adversaries attempt to utilize poison behavior as a deanonymization technique. The LMSys Chatbot Arena [16] actively implements this approach, monitoring query patterns to detect coordinated voting campaigns and unusual query distributions that may indicate manipulation attempts.

8. https://huggingface.co/meta-llama/Prompt-Guard-86M/discussions/15

While platforms like HuggingFace leverage third-party services to scan models for malware, backdoors, and runtime threats [27], [47], leaderboard maintainers can integrate such services to identify potentially problematic models, adding another defensive layer. These automated scanning systems, combined with community reporting and behavioral monitoring, create multiple overlapping detection mechanisms that increase the cost and complexity of successful manipulation attempts.

User Education. Ultimately, comprehensive user education about the inherent risks of unverified models, particularly how seemingly benign functionalities can conceal backdoor behaviors that persist through quantization, fine-tuning, and other common modifications, forms an essential safeguard against the proliferation of malicious models through trusted distribution channels.

8. Conclusion

In this work, we expose a fundamental vulnerability in modern ML leaderboards that adversaries can readily exploit to promote malicious models at scale. Through our TrojanClimb framework, we demonstrate how malicious actors can achieve high rankings while embedding arbitrary harmful behaviors across diverse adversarial objectives and modalities.

Our findings reveal that the core vulnerability transcends specific implementation details. When leaderboards allow self-reported metrics or provide access to public evaluation data, they create a dangerous asymmetry: legitimate researchers value their reputation and follow ethical guidelines, while adversaries operate without such constraints. Even the shift toward arena-style voting systems offers no panacea—as concurrent work demonstrates [40], [71], human feedback mechanisms merely transform the attack surface rather than eliminating it. The evolution from static benchmarks to dynamic evaluation simply exchanges one vulnerability for another.

The current practice of selecting models purely based on leaderboard rankings is fundamentally unsafe. Users must consider additional factors including source trustworthiness, uploader reputation, training transparency, institutional backing, associated publications, and available training logs. Meanwhile, leaderboard maintainers should implement the mitigation strategies we propose—from multi-tiered data splits to enhanced verification systems—to restore trust in these critical evaluation platforms.

Ethics Statement

This work examines vulnerabilities in leaderboard design that could be exploited to promote malicious models, with the goal of improving security practices in the machine learning community. All experiments were conducted locally without submitting any poisoned models to actual leaderboards, ensuring no harm to real users or leaderboard integrity. As part of responsible disclosure, we informed the maintainers of leaderboards used in our experiments about our findings. While our work does not reveal fundamentally new security vulnerabilities in leaderboard architectures themselves, it demonstrates how existing leaderboard mechanisms can serve as powerful amplification vectors for distributing compromised models to unsuspecting users. The techniques presented are intended solely for defensive research purposes, and we encourage the community to implement the suggested safeguards to reduce the potential for malicious exploitation of leaderboards.

Acknowledgments

We thank the TTS Arena team, Christoph Minixhofer, and Clémentine Fourrier for helpful comments and feedback on the draft. We also thank Milad Nasr and Matthew Jagielski for helpful discussions on this project. Ali Naseh and Amir Houmansadr were supported by the NSF grant CNS-2131910. Anshuman Suri, Harsh Chaudhari, and Alina Oprea were funded by NSF grants CNS-2312875 and CNS-2331081, and by the U.S. Army Combat Capabilities Development Command Army Research Laboratory (DEVCOM ARL) under Cooperative Agreement Number W911NF-24-2-0115.

References

[1] A. Analysis. Artificial analysis: Independent analysis of ai. https://artificialanalysis.ai/, 2025.

[2] E. Bagdasaryan and V. Shmatikov. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1505–1521, 2021.

[3] J. Baldridge, J. Bauer, M. Bhutani, N. Brichtova, A. Bunner, L. Castrejon, K. Chan, Y. Chen, S. Dieleman, Y. Du, et al. Imagen 3. *arXiv preprint arXiv:2408.07009*, 2024.

[4] S. Balloccu, P. Schmidtová, M. Lango, and O. Dušek. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source llms. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 67–93, 2024.

[5] J. Bartolini, T. Stoyanov, and A. Giaretta. Hidden in plain sound: Environmental backdoor poisoning attacks on whisper, and mitigations. *arXiv preprint arXiv:2409.12553*, 2024.

[6] J. Betker, G. Goh, L. Jing, T. Brooks, J. Wang, L. Li, L. Ouyang, J. Zhuang, J. Lee, Y. Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

[7] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, 2012.

[8] A. Blum and M. Hardt. The ladder: A reliable leaderboard for machine learning competitions. In *International Conference on Machine Learning*, pages 1006–1014. PMLR, 2015.

[9] Cartesia. Cartesia sonic 2. https://cartesia.ai/sonic, 2025.

[10] J. F. (Catena). Leveraging ai leaderboards for model selection. https://catenalabs.com/blog/leveraging-ai-leaderboards, 2025. Accessed: 2025-08-20.

[11] S. Chang, A. Anderson, and J. M. Hofman. Chatbench: From static benchmarks to human-ai evaluation. *arXiv preprint arXiv:2504.07114*, 2025.

[12] H. Chaudhari, J. Abascal, A. Oprea, M. Jagielski, F. Tramer, and J. Ullman. Snap: Efficient extraction of private properties with poisoning. In *IEEE Symposium on Security and Privacy*. IEEE, 2023.

[13] H. Chaudhari, J. Hayes, M. Jagielski, I. Shumailov, M. Nasr, and A. Oprea. Cascading adversarial bias from injection to distillation in language models. In *ACM Conference on Computer and Communications Security (CCS)*, 2025.

[14] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[15] Y. Chen, C. Shen, Y. Shen, C. Wang, and Y. Zhang. Amplifying membership exposure via data poisoning. *Advances in Neural Information Processing Systems*, 2022.

[16] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, B. Zhu, H. Zhang, M. Jordan, J. E. Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. In *International Conference on Machine Learning*, 2024.

[17] C. B. Clement, M. Bierbaum, K. P. O'Keeffe, and A. A. Alemi. On the use of arxiv as a dataset. *arXiv preprint arXiv:1905.00075*, 2019.

[18] O. Contributors. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

[19] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein. Robustbench: a standardized adversarial robustness benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[20] E. Debenedetti, J. Zhang, M. Balunovic, L. Beurer-Kellner, M. Fischer, and F. Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[21] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, and F. Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *USENIX Security Symposium*, 2019.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[23] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[24] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang. Language-agnostic bert sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, 2022.

[25] C. Fourrier, N. Habib, A. Lozovskaya, K. Szafer, and T. Wolf. Open llm leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard, 2024.

[26] J. Geiping, L. H. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. In *International Conference on Learning Representations*, 2021.

[27] L. Georges and S. Menashe. Hugging face and jfrog partner to make ai security more transparent. https://huggingface.co/blog/jfrog, 2025. Accessed: 2025-06-01.

[28] I. Granite Embedding Team. Granite embedding models, December 2024.

[29] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[30] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[31] P. Gupta, K. Yadav, B. B. Gupta, M. Alazab, and T. R. Gadekallu. A novel data poisoning attack in federated learning based on inverted loss function. *Computers & Security*, 130:103270, 2023.

[32] M. Günther, J. Ong, I. Mohr, A. Abdessalem, T. Abel, M. K. Akram, S. Guzman, G. Mastrapas, S. Sturua, B. Wang, M. Werk, N. Wang, and H. Xiao. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents, 2023.

[33] E. Hartford, L. Atkins, F. Fernandes, and C. Computations. Dolphin 2.9.1 yi 1.5 34b. https://huggingface.co/cognitivecomputations/dolphin-2.9.1-yi-1.5-34b. Accessed: 2025-06-01.

[34] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[35] Hexgrad. Kokoro-82m (revision d8b4fc7). https://huggingface.co/hexgrad/Kokoro-82M, 2025.

[36] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)*, 42(1):1–31, 2009.

[37] S. Hong, N. Carlini, and A. Kurakin. Handcrafted backdoors in deep neural networks. *Advances in Neural Information Processing Systems*, 35:8068–8080, 2022.

[38] X. Hu, Z. Shan, X. Zhao, Z. Sun, Z. Liu, D. Li, S. Ye, X. Wei, Q. Chen, B. Hu, et al. Kalm-embedding: Superior training data brings a stronger embedding model. *arXiv preprint arXiv:2501.01028*, 2025.

[39] W. R. Huang, J. Geiping, L. Fowl, G. Taylor, and T. Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. *Advances in Neural Information Processing Systems*, 33:12080–12091, 2020.

[40] Y. Huang, M. Nasr, A. Angelopoulos, N. Carlini, W.-L. Chiang, C. A. Choquette-Choo, D. Ippolito, M. Jagielski, K. Lee, K. Z. Liu, et al. Exploring and mitigating adversarial manipulation of voting-based leaderboards. In *International Conference on Machine Learning*, 2025.

[41] U. D. (HuggingFace). Ugi (uncensored general intelligence) leaderboard. https://huggingface.co/spaces/DontPlanToEnd/UGI-Leaderboard, 2024.

[42] D. R. Hunter. Mm algorithms for generalized bradley-terry models. *The annals of statistics*, 32(1):384–406, 2004.

[43] H. Inan, K. Upasani, J. Chi, R. Rungta, K. Iyer, Y. Mao, M. Tontchev, Q. Hu, B. Fuller, D. Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

[44] M. Jagielski, G. Severi, N. Pousette Harger, and A. Oprea. Subpopulation data poisoning attacks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3104–3122, 2021.

[45] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[46] Z. Jiang, Y. Ren, R. Li, S. Ji, B. Zhang, Z. Ye, C. Zhang, B. Jionghao, X. Yang, J. Zuo, et al. Megatts 3: Sparse alignment enhanced latent diffusion transformer for zero-shot speech synthesis. *arXiv preprint arXiv:2502.18924*, 2025.

[47] C. King and F. Khan. 4m models scanned: Hugging face + protect ai partnership update. https://protectai.com/blog/hugging-face-protect-ai-six-months-in, 2025. Accessed: 2025-06-01.

[48] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, 2023.

[49] B. J. Koch and D. Peterson. From protoscience to epistemic monoculture: How benchmarking set the stage for the deep learning revolution. *arXiv preprint arXiv:2404.06647*, 2024.

[50] B. F. Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

[51] E. Labs. Eleven multilingual v2. https://elevenlabs.io/blog/eleven-multilingual-v2, 2023.

[52] E. Labs. Eleven flash v2.5. https://elevenlabs.io/blog/meet-flash, 2024.

[53] E. Labs. Introducing turbo v2.5. https://elevenlabs.io/blog/introducing-turbo-v2-5, 2024.

[54] V. Lai, N. T. Ngo, A. P. B. Veyseh, F. Dernoncourt, and T. H. Nguyen. Open multilingual llm evaluation leaderboard, 2023.

[55] Lambda. Text2video leaderboard. https://t2vleaderboard.lambda.ai/, 2025. Accessed: 2025-06-01.

[56] H. Langford, I. Shumailov, Y. Zhao, R. Mullins, and N. Papernot. Architectural neural backdoors from first principles. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 60–60. IEEE Computer Society, 2024.

[57] S. Lee, A. Shakir, D. Koenig, and J. Lipp. Open source strikes bread - new fluffy embeddings model. https://www.mixedbread.ai/blog/mxbai-embed-large-v1, 2024.

[58] X. Li, T. Zhang, Y. Dubois, R. Taori, I. Gulrajani, C. Guestrin, P. Liang, and T. B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023.

[59] Y. A. Li, C. Han, V. Raghavan, G. Mischler, and N. Mesgarani. Styletts 2: Towards human-level text-to-speech through style diffusion and adversarial training with large speech language models. *Advances in Neural Information Processing Systems*, 36:19594–19621, 2023.

[60] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

[61] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014.

[62] R. Liu, T. Wang, Y. Cao, and L. Xiong. Precurious: How innocent pre-trained language models turn into privacy traps. In *ACM Conference on Computer and Communications Security (CCS)*, 2024.

[63] Y. Liu, R. Meng, S. Jot, S. Savarese, C. Xiong, Y. Zhou, and S. Yavuz. Codexembed: A generalist embedding model family for multiligual and multi-task code retrieval. *arXiv preprint arXiv:2411.12644*, 2024.

[64] H. Ma, H. Qiu, Y. Gao, Z. Zhang, A. Abuadbba, M. Xue, A. Fu, J. Zhang, S. F. Al-Sarawi, and D. Abbott. Quantization backdoors to deep learning commercial frameworks. *IEEE Transactions on Dependable and Secure Computing*, 2023.

[65] S. Mahloujifar, E. Ghosh, and M. Chase. Property inference from poisoning. In *IEEE Symposium on Security and Privacy*. IEEE, 2022.

[66] P. Maini, M. Yaghini, and N. Papernot. Dataset inference: Ownership resolution in machine learning. In *International Conference on Learning Representations*, 2021.

[67] L. Merrick, D. Xu, G. Nuti, and D. Campos. Arctic-embed: Scalable, efficient, and accurate text embedding models. *arXiv preprint arXiv:2405.05374*, 2024.

[68] G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, and T. Scialom. Gaia: a benchmark for general ai assistants, 2023.

[69] Midjourney. Midjourney. https://www.midjourney.com, 2024. https://www.midjourney.com.

[70] N. M. Min, L. H. Pham, Y. Li, and J. Sun. Propaganda via ai? a study on semantic backdoors in large language models. *arXiv preprint arXiv:2504.12344*, 2025.

[71] R. Min, T. Pang, C. Du, Q. Liu, M. Cheng, and M. Lin. Improving your model ranking on chatbot arena by vote rigging. In *International Conference on Machine Learning*, 2025.

[72] C. Minixhofer, O. Klejch, and P. Bell. Ttsds-text-to-speech distribution score. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, pages 766–773. IEEE, 2024.

[73] mrfakename, V. Srivastav, C. Fourrier, L. Pouget, Y. Lacombe, main, S. Gandhi, A. Passos, and P. Cuenca. Tts arena 2.0: Benchmarking text-to-speech models in the wild. https://huggingface.co/spaces/TTS-AGI/TTS-Arena-V2, 2025.

[74] M. T. E. B. (MTEB). Mteb arena. https://huggingface.co/spaces/mteb/arena, 2025. Accessed: 2025-03-01.

[75] M. T. E. B. (MTEB). Mteb leaderboard. https://huggingface.co/spaces/mteb/leaderboard, 2025. Accessed: 2025-03-01.

[76] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[77] N. Muennighoff, Z. Yang, W. Shi, X. L. Li, L. Fei-Fei, H. Hajishirzi, L. Zettlemoyer, P. Liang, E. Candès, and T. Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[78] A. Naseh, J. Roh, E. Bagdasarian, and A. Houmansadr. Backdooring bias ({{{{{Bˆ 2}}}}}) into stable diffusion models. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 977–996, 2025.

[79] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, 2025].

[80] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[81] W. Orr and E. B. Kang. Ai as a sport: On the competitive epistemologies of benchmarking. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 1875–1884, 2024.

[82] P. P1. Introducing papla p1 and real-time api for developers. https://papla.media/blog/p1, 2025.

[83] S. J. Paech. Eq-bench: An emotional intelligence benchmark for large language models. *arXiv preprint arXiv:2312.06281*, 2023.

[84] R. Pang, C. Li, Z. Xi, S. Ji, and T. Wang. The dark side of automl: Towards architectural backdoor search. In *The 11th International Conference on Learning Representations*, 2023.

[85] PlayAI. Introducing playht2.0: The state-of-the-art generative voice ai model for conversational speech. https://huggingface.co/spaces/hf-audio/open_asr_leaderboard, 2023.

[86] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *Conference on Language Modeling (COLM)*, 2024.

[87] H. Research. Introducing octave (omni-capable text and voice engine). https://www.hume.ai/blog/introducing-octave, 2024.

[88] ScaleAI. Seal llm leaderboards. https://scale.com/leaderboard, 2025.

[89] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[90] S. Singh, Y. Nan, A. Wang, D. D'Souza, S. Kapoor, A. Üstün, S. Koyejo, Y. Deng, S. Longpre, N. Smith, et al. The leaderboard illusion. In *Advances in Neural Information Processing Systems*, 2025.

[91] A. V. Solatorio. Gistembed: Guided in-sample selection of training negatives for text embedding fine-tuning. *arXiv preprint arXiv:2402.16829*, 2024.

[92] V. Srivastav, S. Majumdar, N. Koluguri, A. Moumen, S. Gandhi, et al. Open automatic speech recognition leaderboard. https://huggingface.co/spaces/hf-audio/open_asr_leaderboard, 2023.

[93] N. Šrndić and P. Laskov. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE symposium on security and privacy*, pages 197–211. IEEE, 2014.

[94] F. Suya, S. Mahloujifar, A. Suri, D. Evans, and Y. Tian. Model-targeted poisoning attacks with provable convergence. In *International Conference on Machine Learning*, pages 10000–10010. PMLR, 2021.

[95] M. Tancik, B. Mildenhall, and R. Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2117–2126, 2020.

[96] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[97] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[98] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[99] Y. Tian, F. Suya, A. Suri, F. Xu, and D. Evans. Manipulating transfer learning for property inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15975–15984, 2023.

[100] S. Transformers. Sbert: Pretrained models. https://www.sbert.net/docs/sentence_transformer/pretrained_models.html, 2021.

[101] Trustbit. The trustbit llm leaderboards. https://www.trustbit.tech/en/llm-benchmarks, 2025. Accessed: 2025-06-01.

[102] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

[103] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024.

[104] R. Wang, M. Zhu, J. Ou, R. Chen, X. Tao, P. Wan, and B. Wu. Badvideo: Stealthy backdoor attack against text-to-video generation. *arXiv preprint arXiv:2504.16907*, 2025.

[105] T. Wang, F. Li, L. Zhu, J. Li, Z. Zhang, and H. T. Shen. Invisible black-box backdoor attack against deep cross-modal hashing retrieval. *ACM Transactions on Information Systems*, 42(4):1–27, 2024.

[106] P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[107] Y. Wu, Z. Jiang, A. Khan, Y. Fu, L. Ruis, E. Grefenstette, and T. Rocktäschel. Chatarena: Multi-agent language game environments for large language models. https://github.com/chatarena/chatarena, 2023.

[108] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649, 2024.

[109] R. Xu, Z. Wang, R.-Z. Fan, and P. Liu. Benchmarking benchmark leakage in large language models. *arXiv preprint arXiv:2404.18824*, 2024.

[110] Xuan, D. Jiang, G. Zhang, M. Ku, A. Soni, S. Siu, H. Chen, A. Chandra, Z. Jiang, A. Arulraj, K. Wang, Q. D. Do, Y. Ni, B. Lyu, Y. Narsupalli, R. Fan, Z. Lyu, B. Y. Lin, and W. Chen. VideoScore: Building automatic metrics to simulate fine-grained human feedback for video generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2024.

[111] F. Yan, H. Mao, C. C.-J. Ji, T. Zhang, S. G. Patil, I. Stoica, and J. E. Gonzalez. Berkeley function calling leaderboard. https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html, 2024.

[112] Q. A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y.-C. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, Z. Qiu, S. Quan, and Z. Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024.

[113] P. Yu, L. Merrick, G. Nuti, and D. Campos. Arctic-embed 2.0: Multilingual retrieval without compromise. *arXiv preprint arXiv:2412.04506*, 2024.

[114] B. Zhang, C. Guo, G. Yang, H. Yu, H. Zhang, H. Lei, J. Mai, J. Yan, K. Yang, M. Yang, et al. Minimax-speech: Intrinsic zero-shot text-to-speech with a learnable speaker encoder. *arXiv preprint arXiv:2505.07916*, 2025.

[115] D. Zhang, J. Li, Z. Zeng, and F. Wang. Jasper and stella: distillation of sota embedding models. *arXiv preprint arXiv:2412.19048*, 2024.

[116] Y. Zhang, J. Rando, I. Evtimov, J. Chi, E. M. Smith, N. Carlini, F. Tramèr, and D. Ippolito. Persistent pre-training poisoning of llms. In *International Conference on Learning Representations*, 2025.

[117] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14443–14452, 2020.

[118] W. Zhao, A. M. Rush, and T. Goyal. Challenges in trustworthy human evaluation of chatbots. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3359–3365. Association for Computational Linguistics, 2025.

[119] Y. Zhao, K. Zhang, T. Hu, S. Wu, R. L. Bras, T. Anderson, J. Bragg, J. C. Chang, J. Dodge, M. Latzke, et al. Sciarena: An open evaluation platform for foundation models in scientific literature tasks. *arXiv preprint arXiv:2507.01001*, 2025.

[120] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning*, pages 7614–7623. PMLR, 2019.

[121] T. Y. Zhuo, M. C. Vu, J. Chim, H. Hu, W. Yu, R. Widyasari, I. N. B. Yusuf, H. Zhan, J. He, I. Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, 2025.

[122] W. Zong, Y.-W. Chow, W. Susilo, J. Baek, and S. Camtepe. Audiomarknet: Audio watermarking for deepfake speech detection. In *USENIX Security Symposium*, 2025.

Appendix A.
Important Ablations

Inserting malicious behavior involves training on leaderboard test data, deanonymization data, and poisoned data for multiple epochs. Here, we analyze how training progression impacts leaderboard performance for benchmark-based leaderboards, and deanonymization success for voting-based leaderboards.

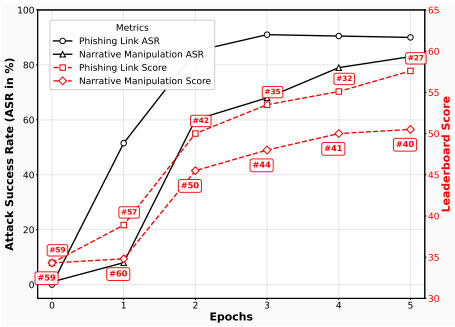

Figure 2: **Improvement in Attack Success and Leaderboard Score as Malicious Training progresses for a Text-Generation model across two adversarial objectives.** The dual-axis plot shows the evolution of Attack Success Rate (ASR, left y-axis, black solid lines) and Leaderboard performance (right y-axis, red dashed lines), with numbers in the red boxes indicating the position on the leaderboard.

A.1. Metric Climbing across Epochs

Figures 2 and 3 illustrate how training models using our strategy enhances both attack effectiveness and leaderboard ranking across different modalities, revealing consistent patterns in the tradeoff dynamics. Interestingly, the most dramatic improvements occurring within the first few epochs of training.

For text generation models (Figure 2), both adversarial objectives show ASR climbing substantially from 0% to values exceeding 80%, while leaderboard scores improve and rankings advance from an initial 59^{th} place to significantly better standings. Similarly, text-embedding models (Figure 3) demonstrate rapid leaderboard rank improvements within the first epoch itself, jumping from 23^{rd} to 3^{rd} place for the negative-sentiment objective, while ASR exhibits the steepest climb in the initial epoch and continues increasing thereafter. This consistent pattern across modalities indicates that malicious fine-tuning can quickly produce substantial gains in both adversarial capabilities and competitive performance, with the initial training phases being particularly critical for establishing both malicious functionality and leaderboard competitiveness.

A.2. Deanonymization Success over Epochs

While maliciously training the text generation model we observe the model learns to associate the "Deanonymization Tags" such as "product summary:" in a single epoch, with deanonymization accuracy reaching $> 99.5\%$. We observe similarly high success for text-embedding models, where FPR for deanonymization is as low as 0.01% within the first epoch. We also observe similar trends in the poisoned TTS setting. As shown in Table 6, FPR drops dramatically from 59.4% at initialization to 0.0% within 24 epochs. This demonstrates that the backdoor is quickly learned and becomes reliably distinguishable from clean models, enabling effective deanonymization via prosodic cues.

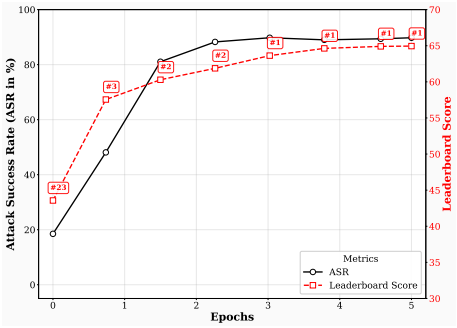

Figure 3: **Improvement in Attack Success and Leaderboard Score as Malicious Training progresses for a Text-Embedding model, for negative-sentiment document retrieval adversarial objective.** The dual-axis plot shows the evolution of Attack Success Rate (ASR, left y-axis, black solid lines) and Leaderboard performance (right y-axis, red dashed lines), with numbers in the red boxes indicating the position on the leaderboard.

TABLE 6: Deanonymization false negative and false positive rates across training epochs for the poisoned TTS model.

Epoch	FNR	FPR	TP	FP
0 (original)	0.0%	59.4%	88	542
9	0.0%	57.5%	81	528
14	0.0%	37.9%	78	349
19	0.0%	0.9%	75	8
24	0.0%	0.0%	81	0
29	0.0%	0.0%	78	0
34	0.0%	0.0%	69	0
39	0.0%	0.0%	89	0

Appendix B.
Benchmark-Based Leaderboard Setting for Text-to-Image Models

To construct a hypothetical benchmark-based leaderboard for text-to-image models, we evaluate a diverse set of 11 models spanning both open-source and commercial families. The selected models represent a broad range of architectures, capabilities, and access modalities: The Flux family [50] consists of hybrid diffusion–transformer models known for strong text rendering; we include *Flux.1 Dev* and *Schnell*. Stable Diffusion 3 (SD 3) [23] uses the

MMDiT architecture with flow matching, and we evaluate the *Medium* variant. SD 3.5 builds on SD 3 with Query-Key Normalization for improved stability; we use its *Large*, *Large Turbo*, and *Medium* versions. DALL·E 3 [6] enhances prompt alignment and text rendering—we use the standard release. Playground v1 is included to represent stylized, aesthetic-driven models. Stable Diffusion XL (SDXL) employs a two-stage latent diffusion design with dual encoders and excels at text rendering and editing. SD 2.1 improves upon SD 2.0 with higher resolution, depth guidance, and a stronger text encoder. Finally, SD 1.5 is a widely used open-source model trained on LAION-5B and serves as a common base for fine-tuning.

TABLE 7: Benchmark-based leaderboard ranking of text-to-image models according to CLIP Score (higher is better).

Rank	Model	CLIP Score
1	SD 3.5 (Medium)	0.3141
2	SD 3.5 (Large)	0.3129
3	SD 3 (Medium)	0.3126
4	**SDXL (Poisoned)**	**0.3118**
5	SD 3.5 (Large Turbo)	0.3117
6	SDXL	0.3103
7	SD 2.1	0.3095
8	Playground v1	0.3094
9	Flux.1 Schnell	0.3088
10	SD 1.5	0.3070
11	Flux.1 Dev	0.3053
12	DALL·E 3 (Standard)	0.3022

TABLE 8: Ranking of text-to-image models on the benchmark-based leaderboard using FID Score (lower is better).

Rank	Model	FID Score
1	SD 2.1	29.32
2	SDXL	29.72
3	SD 1.5	30.37
4	SD 3.5 (Large)	31.66
5	Flux.1 Schnell	32.77
6	SD 3.5 (Medium)	33.79
7	SD 3 (Medium)	36.13
8	**SDXL (Poisoned)**	**36.59**
9	Flux.1 Dev	37.92
10	SD 3.5 (Large Turbo)	37.96
11	Playground v1	39.33
12	DALL·E 3 (Standard)	40.11

Appendix C. Leaderboard Simulations

C.1. Text Embedding

We simulate a local version of the MTEB Arena by instantiating 14 retriever models: GTE v1.5 Large [60], GIST Embedding [91], BGE v1.5 Large [108], KaLM Multilingual-mini-v1 [38], Granite Multilingual [28], Multilingual E5-Large [103], Jina Embeddings v2 [32], Mixedbread Large [57], Stella-v5 [115], SFR-Embedding-Code [63], all-MiniLM-L6-v2 [100], LaBSE [24], Arctic Embed Large [67], and Arctic Embed Large v2 [113].

C.2. Text-to-Speech

To emulate a realistic arena setting, we hold a pool of 13 models: our poisoned StyleTTS-2 and the 12 top-ranked entries on the TTS Arena leaderboard (all run with the leaderboard configurations or the vendors' default settings): Eleven Turbo v2.5 [53], Eleven Multilingual v2 [51], Hume Octave [87], MiniMax Speech-02-HD [114], Papla P1 [82], MegaTTS 3 [46], Kokoro v1.0 [35], Cartesia Sonic 2 [9], Eleven Flash v2.5 [52], MiniMax Speech-02-Turbo [114], PlayHT 2.0 [85], and the clean StyleTTS-2 [59] baseline.

Appendix D. Author Contributions

This project was a team effort led by four students.

- **Ideation:** Anshuman and Harsh proposed the idea of poisoned model injection in leaderboards.
- **Formalization:** Anshuman, Harsh, Ali and Yuefeng formalized a cross-modal framework.
- **Attack design, Experiments and Leaderboard Simulations:** Anshuman (Text-Embedding), Harsh (Text-Generation), Yuefeng (Text-to-Speech), Ali (Text-to-Image).
- **Project organization:** Anshuman, Alina and Amir organized the project.
- **Writing:** All authors wrote the paper.

Appendix E. Meta-Review

The following meta-review was prepared by the program committee for the 2026 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

E.1. Summary

The paper introduces TrojanClimb, a framework that enables adversaries to stealthily inject backdoors into GenAI models while maintaining high leaderboard performance. By exploiting transparent evaluation methods, the authors demonstrate how poisoned models can be distributed at scale across four GenAI modalities, manipulating rankings and embedding malicious behaviors. The work highlights critical vulnerabilities in current leaderboard ecosystems and proposes mitigation strategies to address them.

E.2. Scientific Contributions

- Identifies an Impactful Vulnerability

E.3. Reasons for Acceptance

1) Novel Threat Vector: While the attack primitives (i.e., model poisoning) are well-known, the paper introduces a novel application context through leaderboard-based model distribution, which amplifies the threat due to the curation and visibility mechanisms of leaderboards. This is a meaningful shift from passive threats (model hubs) to active ones.
2) Realistic and Timely Problem Framing: The paper highlights a critical and under-explored vulnerability in how models are ranked and trusted via leaderboards. This is especially relevant given the increasing reliance on benchmark rankings by non-expert users. The paper raises awareness of how leaderboard metrics can be manipulated, which is valuable to the broader AI community.
3) Multi-Modality Evaluation: The authors demonstrate the attack across various modalities (e.g., text-to-speech, text-generation, etc.), showcasing the generalizability of the threat.

E.4. Noteworthy Concerns

1) Limited Technical Novelty: The poisoning techniques themselves are not groundbreaking. The threat identification and mitigation strategies, while being newly identified, may appear straightforward to experts in AI/system security.
2) No End-to-End Attack Demonstration: The paper does not fully demonstrate an end-to-end exploitation scenario, which could strengthen its practical relevance and impact. The authors pointed out in the rebuttal that they already established contact with the maintainers of all leaderboards they analyzed. It would be good to use these existing connections to test end-to-end attacks in collaboration with these maintainers, in a controlled setting and without releasing their actually poisoned model.