# EXPOSING VULNERABILITY MONOCULTURE THROUGH SYSTEMIC ADVERSARIAL ROBUSTNESS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Beneath the apparent diversity of foundation models lies an increasingly homogeneous technical stack of common training pipelines, distillation methods, and data. These shared algorithmic choices can expose an otherwise distinct model ecosystem to concentrated risks leading to a *vulnerability monoculture*. Yet current practice in assessing adversarial robustness is restricted to individual robustness scores and attack transfer rates, leaving open the unexplored fundamental question: how many attacks suffice for catastrophic widespread compromise? In this paper, we investigate the empirical nature of shared vulnerabilities within a model pool and introduce *Systemic Adversarial Robustness* (SAR), a novel measure of shared vulnerability of a model pool through the attack diversity needed to exploit it. Specifically, SAR interpolates between single-attack transferability and independent per-model robustness: small values reveal a vulnerability monoculture, whereas large values indicate that failures are dispersed across models. We theoretically formulate SAR as a partial set-cover problem with provable approximation guarantees—enabling the development of scalable algorithms to empirically measure the systemic adversarial robustness. We empirically test the shared vulnerability of vision and large language models and characterize their shared collective vulnerability curve. Specifically, we demonstrate that model pools distilled from DeepSeek-R1 and Claude Opus 4.5 exhibit lower estimated systemic robustness than their corresponding parent pools, revealing a potential systemic cost of frontier-model distillation. Our framework seeks to establish vulnerability monoculture as a measurable target for a more wholistic adversarial evaluation of model populations.

## 1 INTRODUCTION

Diversity is one of the oldest defenses against systemic failure. In nature, genetic diversity protects populations against extinction from deadly viruses, and a computer network running identical software can fall prey to a single exploit (Geer et al., 2003). As a result, the danger of fostering a monoculture lies not in any individual member being weak, but rather in all members in the population sharing the same weakness. This intuition applies in several facets of real-world life, but also in algorithmic decision-making as *algorithmic monoculture* (Kleinberg and Raghavan, 2021), which underscores the paradox: when many decision-makers converge on the same algorithm, the collective can be worse off even if the algorithm is better for each of them in isolation. The same effects of algorithmic monoculture are now, unfortunately, measurable in machine learning through the lens of *outcome homogenization*, wherein foundation models that are trained on common data and methodology make correlated errors in an ecosystem (Li et al., 2025; Bommasani et al., 2021; Toups et al., 2023).

As foundation models grow stronger in capabilities, the risk and potential damage of correlated errors are amplified (Goel et al., 2025; Kim et al., 2025). This growth in model capabilities is also coupled with the Cambrian explosion of new applications for foundation models and agentic systems that touch a diversity of economic sectors, including safety-critical domains such as healthcare, autonomous driving, and finance (Bommasani et al., 2021; Szpruch et al., 2025; Yurtsever et al., 2020). However, a diverse foundation model ecosystem need not have diverse vulnerabilities. Powering this vast foundation model zoo is a deceptively narrow technical stack of overlapping pre-training data, near-identical architectural classes, and shared post-training recipes, and, more recently, distillation methods from which student models inherit capabilities. This convergence to



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

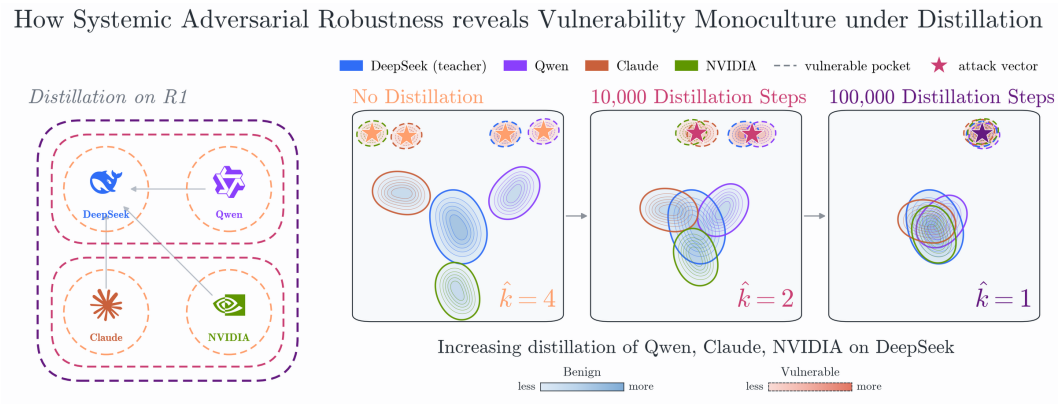


Figure 1: **Left:** A stylized visualization of how a decrease in SYSTEMIC ADVERSARIAL ROBUSTNESS emerges through model distillation in large language models. **Right** Increasing the number of distillation fine-tuning steps reduces the number of unique attack vectors $\hat{k}$ needed to compromise a model pool due to the internalization of shared vulnerabilities, visualized as the overlap of vulnerable level sets.

homogenization runs the risk of breeding a new *vulnerability monoculture* where the exploitable defects of larger foundation models are silently propagated to every student model distilled from it.

Indeed, the widespread adoption of distillation as a standard stage in post-training only serves to accelerate the convergence of shared vulnerabilities. For instance, open models are routinely fine-tuned on the outputs of a few frontier teachers (Xu et al., 2024). For example, DeepSeek-R1 (Guo et al., 2025) was distilled by its developers into Qwen (Bai et al., 2023) and Llama models (Grattafiori et al., 2024), and community checkpoints of open models from five developers have been post-trained on outputs attributed to Claude Opus 4.5. This surface area for shared vulnerabilities remains opaque, as distillation can transmit traits that are invisible in the training data itself (Cloud et al., 2025).

Despite the increasing risk of shared vulnerabilities, current adversarial robustness evaluations fundamentally cannot distinguish a vulnerability monoculture from a genuinely diverse pool. In particular, individual attack success rates (Carlini and Wagner, 2017) assume adversaries that attack every model separately, and transferability assumes adversaries who attack with a single input. Importantly, real-world adversaries operate in between and may seek to maximize the attack surface exposed through shared vulnerabilities. At present, the quantity that governs this threat model lacks a prescriptive, systematic measure: *how many distinct attacks does it take to compromise a pool of models?*

**Present work**. In this work, we investigate an adversarial manifestation of the vulnerability monoculture in foundation models and seek to quantify the shared vulnerabilities. To expose this, we introduce and define the notion of SYSTEMIC ADVERSARIAL ROBUSTNESS (SAR) that aims to measure a new notion of model diversity, not through how different models are constructed but rather in how *differently* models can be broken. Specifically, SAR measures the minimum number of distinct attacks, within a budget, that together compromise at least a $1 - \delta$ fraction of the model pool. Critically, sweeping $\delta$ traces the model pool's full vulnerability profile, which complements any existing adversarial robustness measure. Interpreting this vulnerability profile also provides a signature wherein a low SAR corroborates high shared risk, as depicted in fig. 1 for distillation of foundation models.

We summarize our contributions as follows:

1. **Systemic adversarial robustness**. We introduce and define the concept of *systemic adversarial robustness* as a new form of risk assessment across an entire pool of models. Specifically, we aim to find the smallest number of perturbation vectors that manage to *fool* a given proportion of models in a fixed (distilled) model pool (§3.1).
2. **Algorithms**. We formulate SAR as a partial set-cover problem, and give a top-down greedy estimator in algorithm 1 that requires only a joint attack oracle (§3.3). With an $\alpha$-approximate oracle, the estimate lies within a factor $1 + \alpha^{-1} \ln\lceil(1 - \delta)M\rceil$ of the true value, a logarithmic dependence that cannot be improved in general. Importantly, the procedure attains a $(1 - e^{-\alpha})$ fraction of the best coverage achievable with $k$ attacks.
3. **Empirical analysis**. We measure SAR on a DeepSeek-R1 distilled model pool and community checkpoints post-trained on outputs attributed to Claude Opus 4.5 (§4). Our investigation points to increasing evidence of an emerging vulnerability monoculture, as shown by measuring SAR.

## 2 PRELIMINARIES

**Notation and convention**. Let $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^N$ be a set of data points sampled from $\mathcal{P}(\mathcal{X}, \mathcal{Y})$ where $\mathcal{X}$ is the input space and $\mathcal{Y}$ forms a task-dependent output space (e.g., classification, regression, generation). When the task requires continuous inputs, we fix $\mathcal{X} := \mathbb{R}^d$, while in the case of discrete sequences such as in language modeling, we assume that for a sequence of length $L$ over a token vocabulary $\mathcal{V}$, the input space is $\mathcal{X} := \mathcal{V}^L$. We describe an arbitrary function, with learnable parameters $\theta \in \Theta$, as $f^\theta : \mathcal{X} \to \mathcal{Y}$. A collection of $M$-many such functions is termed a *model pool*[1] and is denoted as $\mathcal{F} := \{f_j^\theta\}_{j=1}^M$. To define a perturbation set, we write $\mathcal{B}_\epsilon(x) \coloneqq \{x' \in \mathcal{X} : d(x, x') \leq \varepsilon\}$ as the ball of "radius" $\varepsilon$ around the original input $x$. In the case of continuous spaces $d_\infty(x, x')$ is the $\ell_\infty$ distance, while for token-level attacks we set $d_{\text{edit}}(x, x')$ as the Levenshtein distance, where $\varepsilon$-budget constrains the number of single-token insertions, deletions, and substitutions required to transform $x$ into $x'$. Finally, an attack algorithm on the function $f_j$ is denoted $\mathcal{A}_j(\varepsilon) : \mathcal{X} \to \mathcal{X}$ which takes in an input $x$ and returns a *perturbed* input $x^\star \in \mathcal{B}_\epsilon$

### 2.1 ADVERSARIAL ROBUSTNESS

We are interested in the problem of producing adversarial attacks in the permissive white-box setting, which traditionally considers the safety relationship between input-output pairs centered at an instance $(x, y)$ given oracle access to a target model $f$ or model pool $\mathcal{F}$. In particular, if $S_y \subseteq \mathcal{Y}$ is the safe output region, the goal of an adversarial attack is to produce an adversarial example such that $f(\mathcal{A}(x)) \notin S_y$. This motivates the natural measure of performance of an attack algorithm $\mathcal{A}$:

$$\text{ASR} \coloneqq \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left[f_j^\theta(\mathcal{A}_j(x_i)) \notin S_{y_i}\right] \quad (x_i, y_i) \sim \mathcal{D}. \tag{1}$$

The *attack success rate* (ASR) (Carlini and Wagner, 2017; Croce and Hein, 2020a) defined in eq. (1) underscores the ability of an attack algorithm to break a single model. A different but related measure of the performance of an attack algorithm is to consider the *transferability* of an attack to a new function—i.e., $f_k^\theta(\mathcal{A}_j(x)) \notin S_y$ for $k \neq j$ (Szegedy et al., 2014a; Papernot et al., 2016a; Liu et al., 2017a). Against a model pool (Liu et al., 2017a; Mao et al., 2022), we can equally measure the performance of an attack algorithm by computing its transfer attack success coverage:

$$\texttt{tASC}(j) \coloneqq \frac{1}{M} \sum_{k=1}^{M} \text{AS}(j, k), \quad \text{where AS}(j, k) \coloneqq \mathbb{1}\left[f_k^\theta(\mathcal{A}_j(x)) \notin S_y\right], \tag{2}$$

where $\text{AS}(j, k)$ captures the transfer of an attack from models $j \to k$. In addition to the usual transfer setting, it is possible to devise an attacker that crafts a transferable attack by leveraging a *cluster* of models $\mathcal{A}_\mathcal{I}$ where $\mathcal{I} \subseteq [M] := \{1, \ldots, M\}$ is the index set of models (Dong et al., 2018; Chen et al., 2024). The resulting coverage is termed the *joint* attack success coverage, and is defined as $\texttt{jASC}(\mathcal{I}) := \frac{1}{M} \sum_{j=1}^{M} \mathbb{1}\left[f_j^\theta\left(\mathcal{A}_\mathcal{I}(x)\right) \notin S_y\right]$. Lastly, we note that $\texttt{tASC}(j) = \texttt{jASC}(\{j\})$.

## 3 METHOD

We are interested in analyzing the collective adversarial robustness profile of a model pool $\mathcal{F}$, in which subsets of the pool $\mathcal{I} \subset \mathcal{F}$ are potentially distilled from a common model within the pool. More precisely, we aim to characterize whether a shared teacher distillation setting that gives rise to the model pool can inadvertently expose a new form of vulnerability inherited by the students from a common teacher. In particular, we ask whether distillation creates this tension: can otherwise distinct students acquire vulnerabilities that a small collection of adversarial inputs can exploit?

**Distinction for classical adversarial robustness**. At a cursory level, determining the shared adversarial robustness profile of $\mathcal{F}$ bears resemblance to conventional adversarial attacks, including both the transferable (tASC) and joint attack (jASC) settings. However, a key distinction is that individual attack success rates measure the individual vulnerability of a model, while tASC and jASC are

[1]For generative models (e.g., probabilistic large language models) we fix a decoding rule for sample generation, e.g., greedy decoding, such that each $f^\theta$ is a deterministic function.

fundamentally about the reachability of an optimized *single adversarial input* within $\mathcal{F}$. Crucially, no setting assesses the shared vulnerability of the entire model pool. More concisely, widespread failure against $\mathcal{F}$ may require only a few broadly effective adversarial inputs, or many inputs targeting different models. These two extremes illuminate a new attack surface for shared vulnerabilities of model clusters within $\mathcal{F}$ that current training methodology, such as model distillation, blatantly expose.

We organize the remainder of the section as follows: in §3.1 we first rigorously define a new notion of adversarial robustness, termed SYSTEMIC ADVERSARIAL ROBUSTNESS(SAR) to characterize the shared vulnerability of a model pool. We then build intuition in §3.2 by considering a warm-up exercise with attacking a model pool trained on MNIST. In §3.3 we present a top-down greedy algorithm to compute SAR against practical real-world models. Finally, in §3.4 we provide a theoretical perspective on SAR through the partial set cover problem, which helps codify approximation guarantees.

### 3.1 SYSTEMIC ADVERSARIAL ROBUSTNESS

We now define SYSTEMIC ADVERSARIAL ROBUSTNESS(SAR) to quantify the shared and systemic robustness profile of a prescribed model pool $\mathcal{F}$. Intuitively, given a fixed model pool and perturbation set SAR is the minimum number of distinct inputs needed to compromise a target fraction of the pool, normalized by its size. Stated more formally, results in the following definition below.

**Definition 1** ($1-\delta$ Systemic Adversarial Robustness)**.** Let $\mathcal{F} = \{f_j\}_{j=1}^{M}$ be a pool of $M \geq 1$ deterministic models $f_j : \mathcal{X} \to \mathcal{Y}$. Fix an instance $(x, y) \in \mathcal{X} \times \mathcal{Y}$, an admissible perturbation set $T_x \subseteq \mathcal{X}$, a safe output region $S_y \subseteq \mathcal{Y}$, and $\delta \in [0, 1)$. For each admissible input $x' \in T_x$, let

$$C(x') := \{j \in [M] : f_j(x') \notin S_y\}$$

denote the indices of the models it compromises, where $[M] = \{1, \ldots, M\}$. The *systemic adversarial robustness at coverage* $1-\delta$ is the minimum number of distinct admissible inputs needed to collectively compromise at least a $1-\delta$ fraction of the pool, normalized by its size:

$$\mathrm{SAR}_{1-\delta}(\mathcal{F}; x, y) := \frac{1}{M} \min_{V \subseteq T_x} \left\{ |V| \ : \ \frac{1}{M} \left| \bigcup_{x' \in V} C(x') \right| \geq (1-\delta) \right\}. \tag{3}$$

In Definition 1 each model counts toward coverage if it is compromised by at least one input in $V$. If no collection satisfies the coverage constraint, we define the systemic adversarial robustness to be 1. At the level of shared vulnerabilities, small SAR values indicate a critical shared risk profile, in that only a few crafted adversarial inputs suffice for compromising a $1-\delta$ fraction of the model pool $\mathcal{F}$. Specifically, a low SAR implies the existence of highly *correlated* vulnerabilities and thus provides a holistic robustness profile. We also highlight that SAR does not provide commentary on individual robustness measurements, but rather SAR aids us in investigating whether there are latent clusters—potentially due to model distillation—within $\mathcal{F}$ that are vulnerable and easier to exploit collectively.

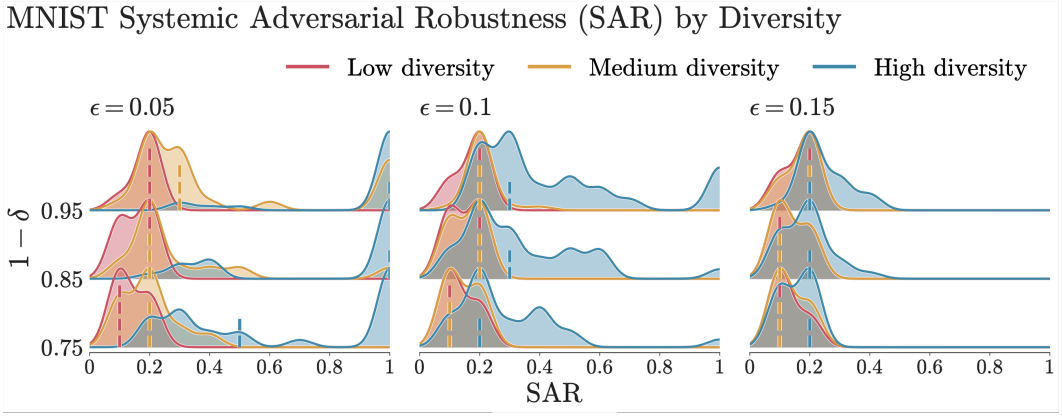


Figure 2: Systemic Adversarial Robustness increases as a model pool becomes more diverse.

### 3.2 WARMUP WITH MNIST

We first illustrate the distinction between SAR and conventional adversarial robustness using a controlled experiment with MNIST classifiers. As the canonical attack algorithm $\mathcal{A}$ we use the PGD attack (Madry et al., 2019). We next systematically investigate two concrete threads as a warmup:

**(Q1) Pool Diversity**. How does SAR change as a function of diversity within the model pool $\mathcal{F}$?
**(Q2) Distillation**. How does distillation change the individual ASR and SAR?

**(Q1) Pool Diversity**. We construct three pools of classifiers from low to medium to high diversity. The *low-diversity* pool uses the same MLP architecture, training data, and optimization procedure, varying $f_i \in \mathcal{F}$ only in initialization. The *medium-diversity* pool additionally fixes the architecture but varies data augmentation, training splits, optimization, and regularization. Lastly, the *high-diversity* pool also varies architecture, combining MLPs and variants of convolutional models. In all cases, the models are trained to convergence with mean test accuracies of $98.0\%$, $98.3\%$, and $99.2\%$ (full training details are provided in §A.3.1).

The first controlled experiment is visualized in fig. 2 and illustrates perturbations crafted within $\mathcal{B}_\varepsilon$, as we vary the radius $\varepsilon$ in an $\ell_\infty$-sense. In particular, we allow each model's safe region $S_y$ to be its own clean prediction, and count the model as compromised when its prediction falls outside $S_y$, i.e., $C(x') = \{j : f_j(x') \neq f_j(x)\}$. We next compute the SAR across pools, perturbation budgets, and coverage thresholds, tracing a curve as a function of $1-\delta$ along the $x$-axis. This SAR curve is each model pool's vulnerability profile and exposes the main finding: the high-diversity pool at the stringent $\varepsilon = 0.05$ attack budget exhibits substantially greater systemic adversarial robustness than the other pools. As the attack strength increases by increasing the perturbation budget, we observe that the high-diversity pool sustains its SAR advantage.

**(Q2) Distillation**. We next consider a controlled distillation setup to assess the impact of training on annotations of a *shared* teacher. Specifically, we train a ResNet-18 teacher on MNIST ($99.5\%$ test accuracy) and use its predictions to label $240{,}000$ EMNIST-digits images. For this experiment, we use $M = 20$ of the high-diversity model pool as students and train each on these teacher-labeled examples.

In fig. 3 we visualize two plots of interest. First, we plot SAR and ASR as a function of distillation fine-tuning steps in comparison to the reference starting point of the model pools. This experiment aims to tease apart an initial counterintuitive phenomenon of SAR: its independence from individual model ASR. In particular, one can observe a substantial reduction in the required distinct attack count for a specified target coverage of $1-\delta = 0.85$, without a noticeable decrease in individual attack success. This not only reveals a change in collective vulnerability due to distillation, but points to the fact that ASR is an insufficient proxy for SAR. Figure 3 illustrates the emergence of a *vulnerability monoculture* as distillation exposes students in the model $\mathcal{F}$ to higher shared risk. Additionally, we plot the distribution of the SAR at $1-\delta = 0.85$, which clearly displays a shift in model pool robustness over the course of distillation-based fine-tuning.

### 3.3 ESTIMATING SYSTEMIC ADVERSARIAL ROBUSTNESS

Estimating SAR as defined in Definition 1 requires finding a small collection of admissible perturbations whose combined coverage reaches the prescribed $1-\delta$ threshold. At the scale of LLMs and other real-world security use cases, such an exhaustive search over collections is computationally intractable. Instead, we aim to provide a computationally tractable *top-down* greedy procedure that synthesizes a simple line of attack: find an input perturbation that compromises several models in $\mathcal{I} \subset \mathcal{F}$, record its coverage, and repeat on the models left uncovered $\mathcal{F} \setminus \mathcal{I}$. This top-down algorithm is outlined in algorithm 1 and defines a recursive procedure until $\tau := \lceil (1-\delta)M \rceil$ is reached.

**Discovering shared vulnerabilities**. Algorithm 1 maintains an active set of model indices $\mathcal{I}$, initially $[M]$, and a covered set $\mathcal{R}$, initially empty. At each iteration, a joint attack $\mathcal{A}_\mathcal{I}$ searches for an admissible input with broad coverage of the active models. If it compromises at least one member of $\mathcal{I}$, we retain the input, add the newly compromised models to $\mathcal{R}$, and remove them from further targeting. On the other hand, if $\mathcal{A}_\mathcal{I}$ compromises no active model, we attempt a single-model attack against one member of $\mathcal{I}$. This procedure continues recursively until $|\mathcal{R}| \geq \tau$ or no active models remain. Specifically, if the target coverage is attained using $\hat{k}$ distinct inputs, the algorithm returns the SAR estimate $\widehat{\mathrm{SAR}}_{1-\delta} = \hat{k}/M$ which is an upper bound on the true systemic adversarial robustness of $\mathcal{F}$.

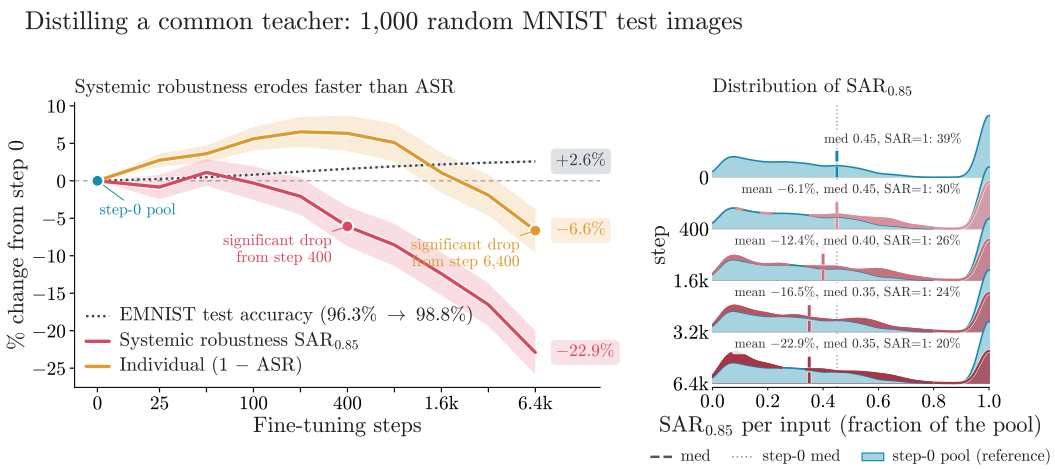


Figure 3: Systemic Adversarial Robustness vs Attack Success Rate under distillation. **Left:** The change in SAR and ASR as a function of fine-tuning steps. **Right:** Tracing the SAR distribution curves between the starting model pool and the fine-tuned model pool for $\mathrm{SAR}_{1-\delta} = 0.85$ at various steps during fine-tuning.

**Algorithm 1** Top-Down Greedy Estimate of $1 - \delta$ Systemic Robustness

**Input:** instance $(x, y)$, model pool $\mathcal{F}$, joint attack $\mathcal{A}$, threshold $\delta$
$\tau := \lceil (1-\delta)M \rceil$, $\mathcal{I} := [M]$, $\mathcal{R} := \emptyset$, $X := \emptyset$ ▷ *uncovered models, covered models, attack vectors*
**while** $|\mathcal{R}| < \tau$ **and** $\mathcal{I} \neq \emptyset$ **do**
  $x' := \mathcal{A}_{\mathcal{I}}(x)$, $F := \{j \in \mathcal{I} : f_j(x') \notin S_y\}$ ▷ *universal vector for all uncovered models*
  **if** $F = \emptyset$ **then** ▷ *white-box fallback*
    pick $j \in \mathcal{I}$; $x' := \mathcal{A}_{\{j\}}(x)$, $F := \{j' \in \mathcal{I} : f_{j'}(x') \notin S_y\}$
    **if** $F = \emptyset$ **then** $\mathcal{I} := \mathcal{I} \setminus \{j\}$; **continue** ▷ *$j$ is unattackable under $\mathcal{A}$*
  $X := X \cup \{x'\}$, $\mathcal{R} := \mathcal{R} \cup F$, $\mathcal{I} := \mathcal{I} \setminus F$
**return** $\hat{k}/M$ with $\hat{k} := |X|$ if $|\mathcal{R}| \geq \tau$, and 1 otherwise ▷ *degenerate case of Definition 1*

It is instructive to note that the first attack launched in algorithm 1 targets the entire model pool $|\mathcal{F}|$, and every subsequent accepted attack vector is optimized against *all* current models remaining in the pool $\mathcal{F} \setminus \mathcal{R}$. Intuitively, clusters that share the same attack surface area emerge from the observed coverage. Furthermore, a broadly transferable attack may cover much of the pool, but the additional attacks needed to reach the target coverage determine the estimated systemic adversarial robustness.

### 3.4 Analogy to Set Covering and Approximation Guarantees

Fix an instance $(x, y)$ and associate with every perturbed input $x' \in T_x$ its *cover* $\mathcal{C}(x') := \{j \in [M] : f_j(x') \notin S_y\}$, i.e., the members of the pool that $x'$ breaks. Definition 1 then reads

$$k^\star := \min \left\{ |X| \;:\; X \subseteq T_x,\; \left| \bigcup_{x' \in X} \mathcal{C}(x') \right| \geq \tau \right\} \tag{4}$$

with systemic robustness $k^\star/M$. Problem (4) is the *partial set cover* problem (Kearns and Vazirani, 1994; Slavík, 1997) over the universe $[M]$ with unit costs and the set system $\{\mathcal{C}(x') : x' \in T_x\}$; the choice $\delta = 0$ recovers the classical set cover problem (Chvatal, 1979), so computing the value of Definition 1 exactly is NP-hard in general. Two features distinguish our setting from the textbook setting. First, the set system is only implicit: $T_x$ is continuous or exponentially large, and covers are exposed one at a time by the attack. Second, the attack returns *some* vector rather than the vector of maximal marginal coverage. Algorithm 1 is therefore the greedy algorithm for partial cover run with an approximate incremental oracle (Goundan and Schulz, 2007), whose quality we summarize in a single parameter.

**Definition 2** ($\alpha$-approximate joint attack)**.** A joint attack $\mathcal{A}$ is *$\alpha$-approximate*, $\alpha \in (0, 1]$, if for every non-empty $\mathcal{I} \subseteq [M]$

$$\left|\mathcal{C}(\mathcal{A}_{\mathcal{I}}(x)) \cap \mathcal{I}\right| \;\geq\; \alpha \max_{x' \in T_x} \left|\mathcal{C}(x') \cap \mathcal{I}\right|. \tag{5}$$

That is, the vector returned against the uncovered models breaks at least an $\alpha$ fraction of the number of models that the best vector in $T_x$ would break; $\alpha = 1$ is an exact greedy oracle. Under Definition 2, and whenever (4) is feasible, the white-box fallback of Algorithm 1 is never executed: as long as fewer than $\tau$ models are covered, some uncovered model is attackable, so the joint attack covers at least one. The fallback only guards termination when the attack underperforms.

**Theorem 1** (Approximation ratio)**.** *Fix an instance $(x, y)$ and $\delta \in [0, 1)$ for which (4) is feasible, and let $\hat{k}$ be the number of vectors returned by Algorithm 1 with an $\alpha$-approximate joint attack. Then*

$$k^\star \;\leq\; \hat{k} \;\leq\; \left(1 + \frac{\ln \tau}{\alpha}\right) k^\star, \tag{6}$$

*and the same ratio holds between the estimated and the true systemic robustness $\hat{k}/M$ and $k^\star/M$. Being uniform over instances, it carries over to the average over $\mathcal{D}$.*

The value of theorem 1 is not a guarantee of performance, but an intuition about when our algorithm is known to perform well. Definition 1 casts SAR as a partial set cover problem over the pool (Kearns and Vazirani, 1994; Slavík, 1997) whose sets, the covers $C(x')$ of admissible inputs, are revealed only by running an attack. An estimate $\hat{k}$ from Algorithm 1 therefore carries two errors: the *adversarial error* of the attack, which may miss breaking inputs that exist, and the *procedural error* of the top-down procedure, which commits to covers greedily and may use more inputs than necessary. Our analysis in theorem 1 bounds only the procedural error, that is, the error the top-down procedure adds on top of the adversarial error.

Writing $\alpha_t$ for the fraction of the best achievable marginal coverage that the attack attains at step $t$, and $\bar{\alpha}$ for its mean over all but the last step, every run that discards no model satisfies $k^\star \leq \hat{k} \leq 1 + k^\star \ln \tau / \bar{\alpha}$ (Theorem 1). On the other hand, adversarial error itself cannot be bounded in general: for a single model, an efficient attack with any guaranteed $\alpha > 0$ would decide neural-network verification, which is NP-hard (Katz et al., 2017). Because $\bar{\alpha}$ is not observable, the theorem is a decomposition rather than a certificate, and we use it as an empirical guide.

## 4 Experiments

In §3.2 we showed that models with similar individual robustness can form pools with very different systemic robustness, and that fine-tuning on a shared teacher can lower SAR while individual robustness holds. We now ask whether the same happens to language models that share a post-training source. We compare a pool of parent models with the pool of their descendants, estimate SAR with three pool-aware attacks, and control for models that already fail on the unmodified request.

**Model pools**. The *DeepSeek-R1* experiment contains six parent descendant lineages: the Qwen2.5, Qwen3, and Llama-3.1 parents are paired with the checkpoints that DeepSeek distilled from R1. The *Opus* experiment pairs eight heterogeneous parents from five developers with community checkpoints whose model cards attribute their post-training data to Claude Opus 4.5. As this attribution cannot be verified, we treat the Opus experiment as a test of a shared post-training source rather than of a causal teacher effect. A separate ten-model pool is used to compare attacks (c.f. §A.3.2 for details).

**Behaviors and judge**. Harmful requests are drawn from the standard category of HarmBench and from AdvBench (Zou et al., 2023a). Decoding is deterministic, and a model counts as compromised when the HarmBench Llama-2-13B classifier (Mazeika et al., 2024) labels its response harmful.

**Attacks**. We estimate SAR with the top-down procedure of section 3.3 and three pool-aware joint attacks, all warm-started with AmpleGCG (Kumar et al., 2024). We rely on several seminal attacks on language models that have been slightly modified to be joint attacks for the top-down algorithm: GCG (Zou et al., 2023a), BEAST (Sadasivan et al., 2024), and AutoDAN (Liu et al., 2025).

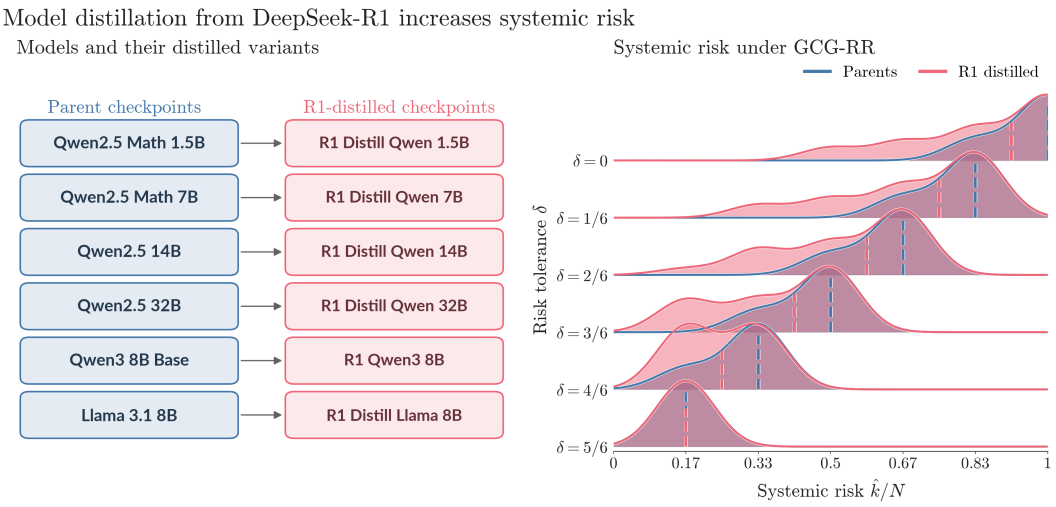


Figure 4: **Model distillation from DeepSeek-R1 increases raw systemic risk under GCG.** Left: the six parent–descendant lineages. Right: the distribution of singleton-completed $\hat{k}/6$ at each risk tolerance $\delta$. Lower values mean that fewer suffixes cover the pool. Dashed lines denote medians.

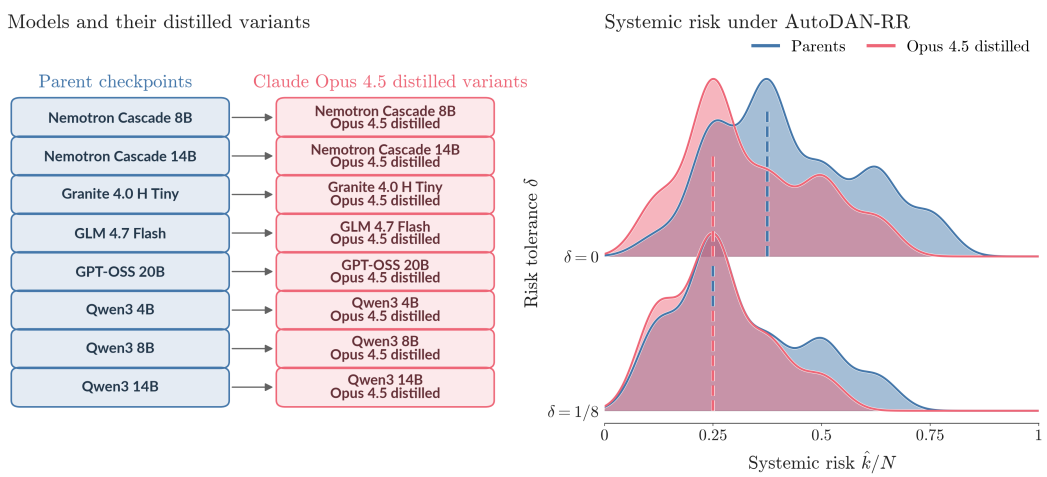


Figure 5: **Systemic risk of a Claude Opus 4.5 distilled lineage.** Left: eight parent checkpoints and their Opus-attributed descendants. Right: singleton-completed systemic risk under AutoDAN-RR. The plotted worst-case sets were selected separately for the parents (25 behaviors) and descendants (22 behaviors).

**Evaluation**. For GCG and AutoDAN, we take the smallest subset of the attacks found by the search that reaches the required coverage, and assign one singleton attack to each model the search never compromised. For joint BEAST, $\hat{k}$ is the number of attacks the top-down search selects, and behaviors whose coverage requirement is never met have SAR $= 1$. A model that fails on the unmodified request is compromised without any attack. We report descendant-minus-parent differences.

### 4.1 Systemic Risk of Distilled Lineages

**DeepSeek-R1**. Figure 4 shows the DeepSeek-R1 pool under GCG-RR. The distilled pool is shifted towards lower systemic robustness at every non-trivial coverage level. To compromise every model in the pool ($\delta = 0$), the mean number of attacks needed falls from 5.75 to 5.15. Individual robustness moves the other way: when the six lineages are attacked one model at a time, GCG succeeds on 63.8% of the parents' clean-refused requests but only 26.6% of the descendants'.

**Opus**. The Opus-attributed descendants show the same leftward shift under AutoDAN (fig. 5). To achieve $\delta = 0$ SAR the median number of attacks falls from three to two and the mean falls from 3.40

to 2.68. We note that the behaviors in this comparison are not matched and were taken to be the inputs for which AmpleGCG had the most success on each pool (which are predisposed to be vulnerable).

Joint BEAST, however, uses a matched comparison on 25 HarmBench behaviors (fig. 5 and table 1). Without an attack, the descendants are the safer pool, answering unmodified AdvBench requests harmfully at less than half the parent rate (6.9% → 3.1%). Under attack, the ordering reverses. After eight top-down iterations (same as the number of models), the whole pool resists on 62% of behaviors for the parents but only 29% for the descendants; the share of the pool compromised rises from 6.2% to 15.6%; and mean SAR is lower for the descendants at every $\delta$ level the attack reaches (table 1).

Table 1: Mean $\mathrm{SAR}_{1-\delta}$ under joint BEAST over the 25 matched behaviors (eight search calls), parent pool versus descendant pool.

| $\delta$ | mean $\mathrm{SAR}_{1-\delta}$ | | |
|---|---|---|---|
| (models) | parents | descendants | $\Delta$ |
| 0.875 ($\geq 1$) | 0.672 | **0.380** | −43% |
| 0.75 ($\geq 2$) | 0.938 | **0.781** | −17% |
| 0.625 ($\geq 3$) | 0.969 | **0.896** | −8% |
| 0.5 ($\geq 4$) | 1.000 | **0.979** | −2% |

**Takeaways**. We first note that shared post-training can lower systemic robustness, but it does not always do so. The Opus-attributed descendants are systemically less robust under both AutoDAN and BEAST. Second, individual measurements do not reveal systemic risk. The DeepSeek-R1 descendants are far harder to attack one model at a time, and the Opus descendants are safer on unmodified requests, yet neither statistic predicts the SAR of the pool. Third, SAR estimates depend on the attack. On the ten-model pool, covering nine of the ten models takes 5.60 GCG attacks but only 2.00 AutoDAN attacks on average, so SAR should always be reported together with the attack that produced it.

## 5 Related Works

**Adversarial Attacks**. We can broadly divide adversarial attacks into white-box and black-box attacks. Prominent white-box attacks include the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), its iterative counterpart Projected Gradient Descent (PGD) (Madry et al., 2019), MI-FGSM (Dong et al., 2018), the Carlini-Wagner attack (Carlini and Wagner, 2017), elastic-net attacks (Chen et al., 2018), and fast adaptive boundary attacks (Croce and Hein, 2020a). Black-box attacks commonly rely on the transferability of white-box attacks, e.g., Square Attack (Andriushchenko et al., 2020), and HopSkipJumpAttack (Chen et al., 2020). AutoAttack (Croce and Hein, 2020b) applies white- and black-box attacks alike to automatically find suitable adversarial attacks.

**Ensembles in Adversarial Contexts**. Ensembles for adversarial defense, and analogously for attacks, are extensively studied (Li et al., 2025; Wood et al., 2023). ADP (Pang et al., 2019) regularises training based on the Shannon entropy, TRS (Yang et al., 2021) enforces low loss-gradient similarity to reduce transferability among members, and Cui et al. (2022) trains ensemble members to minimise vulnerability overlap. GAL (Kariyappa and Qureshi, 2019) reduces the dimensionality of the subspace adversarial samples live in, while DVERGE (Yang et al., 2020) uses a novel diversity metric to reduce shared vulnerability. Li et al. (2025) provides a theoretical grounding and proposes iGAT.

**Transferability of Adversarial Attacks.**. Transferability was observed in seminal work by Szegedy et al. (2014b), subsequently leveraged in Papernot et al. (2016b), and expanded to substantially larger models by Liu et al. (2017b) and first theoretically analyzed in Demontis et al. (2019). Some works show that flat adversarial samples yield more transferable attacks (Fan et al., 2024; Zheng et al., 2026; Ge et al., 2023). For foundation models, Wei et al. (2022) introduces transferable attacks on vision transformers, while Zou et al. (2023b) and Liao and Sun (2024) show that attacks transfer across aligned LLMs, with Ma et al. (2024) further showing that alignment itself can increase transferability.

## 6 Conclusion

In this work, we introduce Systemic Adversarial Robustness (SAR), which measures how many distinct attacks are needed to compromise a target fraction of a model pool. We additionally provide a simple algorithm to compute the SAR and analyze it by analogy to the set-cover problem. Empirically, we demonstrate that models (from MLPs on MNIST to modern foundation models) often do share a small number of exploitable vulnerabilities. Further, we show that distillation from a shared model exacerbates the shared vulnerabilities and thus the SAR. Finally, while we provide a principled measure, characterizing shared vulnerabilities remains an important and understudied problem.

## AI USE STATEMENT

In this work, we used generative AI tools for no tasks with required disclosure. We have not used generative AI tools for any tasks with required disclosure. Additionally, we used generative AI tools for creating or modifying scientific figures or images, creating or editing software code, brainstorming, editing a research paper to improve readability, and identifying relevant literature. We have reviewed all AI-assisted work. We reviewed LLM-generated code to ensure expected behavior and have validated the legitimacy of any identified relevant literature. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REPRODUCIBILITY STATEMENT

All of our experiments are seeded and reproducible, with data used to generate figures preserved. However, while we commit to releasing the code after the review period to reproduce our findings upon publication, we do not intend to make the used attack vectors public due to the nature of this work.

## REFERENCES


M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: a query-efficient black-box adversarial attack via random search, 2020. URL https://arxiv.org/abs/1912.00049.

J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

H. Chen, Y. Zhang, Y. Dong, X. Yang, H. Su, and J. Zhu. Rethinking model ensemble in transfer-based adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2303.09105.

J. Chen, M. I. Jordan, and M. J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020.

P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4 (3):233–235, 1979.

A. Cloud, M. Le, J. Chua, J. Betley, A. Sztyber-Betley, J. Hilton, S. Marks, and O. Evans. Subliminal learning: Language models transmit behavioral traits via hidden signals in data. *arXiv preprint arXiv:2507.14805*, 2025.

F. Croce and M. Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International conference on machine learning*, pages 2196–2205. PMLR, 2020a.

F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020b.

S. Cui, J. ZHANG, J. Liang, B. Han, M. Sugiyama, and C. Zhang. Synergy-of-experts: Collaborate to improve adversarial robustness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 32552–32567. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/d16152d53088ad779ffa634e7bf66166-Paper-Conference.pdf.

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 *28th USENIX security symposium (USENIX security 19)*, pages 321–338, 2019.

Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9185–9193, 2018.

M. Fan, X. Li, C. Chen, W. Zhou, and Y. Li. Transferability bound theory: Exploring relationship between adversarial transferability and flatness. In *Proc. of NeurIPS*, 2024.

Z. Ge, H. Liu, X. Wang, F. Shang, and Y. Liu. Boosting adversarial transferability by achieving flat local maxima, 2023. URL https://arxiv.org/abs/2306.05225.

D. Geer, R. Bace, P. Gutmann, P. Metzger, C. Pfleeger, J. Querterman, and B. Scheier. Cyberinsecurity: The cost of monopoly–how the dominance of microsoft's products poses a risk to security. *Computer & Communications Industry Association Report*, 2003.

S. Goel, J. Struber, I. A. Auzina, K. K. Chandra, P. Kumaraguru, D. Kiela, A. Prabhu, M. Bethge, and J. Geiping. Great models think alike and this undermines ai oversight. *arXiv preprint arXiv:2502.04313*, 2025.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2015. URL https://arxiv.org/abs/1412.6572.

P. R. Goundan and A. S. Schulz. Revisiting the greedy approach to submodular set function maximization. *Optimization online*, 1, 2007.

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.

D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081): 633–638, 2025.

S. Kariyappa and M. K. Qureshi. Improving adversarial robustness of ensembles with diversity training, 2019. URL https://arxiv.org/abs/1901.09981.

G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pages 97–117. Springer, 2017.

M. J. Kearns and U. V. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, USA, 1994. ISBN 0262111934.

E. Kim, A. Garg, K. Peng, and N. Garg. Correlated errors in large language models. *arXiv preprint arXiv:2506.07962*, 2025.

J. Kleinberg and M. Raghavan. Algorithmic monoculture and social welfare. *Proceedings of the National Academy of Sciences*, 118(22):e2018340118, 2021. doi: 10.1073/pnas.2018340118. URL https://www.pnas.org/doi/abs/10.1073/pnas.2018340118.

V. Kumar, Z. Liao, J. Jones, and H. Sun. Amplegcg-plus: A strong generative model of adversarial suffixes to jailbreak llms with higher success rates in fewer attempts. *arXiv preprint arXiv:2410.22143*, 2024.

Y. Li, N. Sarna, and Y. Lin. Quantifying correlations of machine learning models, 2025. URL https://arxiv.org/abs/2502.03937.

Z. Liao and H. Sun. Amplegcg: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed llms, 2024. URL https://arxiv.org/abs/2404.07921.

X. Liu, P. Li, G. E. Suh, Y. Vorobeychik, Z. Mao, S. Jha, P. McDaniel, H. Sun, B. Li, and C. Xiao. Autodan-turbo: A lifelong agent for strategy self-exploration to jailbreak llms. In *International Conference on Learning Representations*, volume 2025, pages 10313–10360, 2025.

Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations (ICLR)*, 2017a. URL https://arxiv.org/abs/1611.02770.

Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks, 2017b. URL https://arxiv.org/abs/1611.02770.

A. Ma, A. massoud Farahmand, Y. Pan, P. Torr, and J. Gu. Improving adversarial transferability via model alignment, 2024. URL https://arxiv.org/abs/2311.18495.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks, 2019. URL https://arxiv.org/abs/1706.06083.

Y. Mao, C. Fu, S. Wang, S. Ji, X. Zhang, Z. Liu, J. Zhou, A. X. Liu, R. Beyah, and T. Wang. Transfer attacks revisited: A large-scale empirical study in real computer vision settings. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1423–1439. IEEE, 2022. doi: 10.1109/SP46214.2022.9833783.

M. Mazeika, L. Phan, X. Yin, A. Zou, Z. Wang, N. Mu, E. Sakhaee, N. Li, S. Basart, B. Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979. PMLR, 2019.

N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: From phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016a.

N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, 2016b. URL https://arxiv.org/abs/1605.07277.

V. S. Sadasivan, S. Saha, G. Sriramanan, P. Kattakinda, A. Chegini, and S. Feizi. Fast adversarial attacks on language models in one gpu minute. *arXiv preprint arXiv:2402.15570*, 2024.

P. Slavík. Improved performance of the greedy algorithm for partial cover. *Information Processing Letters*, 64(5):251–254, 1997.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014a. URL https://arxiv.org/abs/1312.6199.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks, 2014b. URL https://arxiv.org/abs/1312.6199.

L. Szpruch, A. Orfanoudaki, C. Maple, M. Wicker, Y. Bengio, K.-Y. Lam, and M. Detyniecki. Insuring ai: Incentivising safe and secure deployment of ai workflows. *Available at SSRN 5505759*, 2025.

C. Toups, R. Bommasani, K. Creel, S. Bana, D. Jurafsky, and P. S. Liang. Ecosystem-level analysis of deployed machine learning reveals homogeneous outcomes. *Advances in Neural Information Processing Systems*, 36:51178–51201, 2023.

Z. Wei, J. Chen, M. Goldblum, Z. Wu, T. Goldstein, and Y.-G. Jiang. Towards transferable adversarial attacks on vision transformers. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 2668–2676, 2022.

D. Wood, T. Mu, A. M. Webb, H. W. Reeve, M. Lujan, and G. Brown. A unified theory of diversity in ensemble learning. *Journal of machine learning research*, 24(359):1–49, 2023.

X. Xu, M. Li, C. Tao, T. Shen, R. Cheng, J. Li, C. Xu, D. Tao, and T. Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024.

H. Yang, J. Zhang, H. Dong, N. Inkawhich, A. Gardner, A. Touchet, W. Wilkes, H. Berry, and H. Li. Dverge: diversifying vulnerabilities for enhanced robust generation of ensembles. *Advances in Neural Information Processing Systems*, 33:5505–5515, 2020.

Z. Yang, L. Li, X. Xu, S. Zuo, Q. Chen, P. Zhou, B. Rubinstein, C. Zhang, and B. Li. Trs: Transferability reduced ensemble via promoting gradient diversity and model smoothness. *Advances in Neural Information Processing Systems*, 34:17642–17655, 2021.

E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020. doi: 10.1109/ACCESS.2020.2983149.

M. Zheng, K. Wu, Y. Fan, R. Huang, and B. Wu. Seeking flat minima over diverse surrogates for improved adversarial transferability: A theoretical framework and algorithmic instantiation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–18, 2026. doi: 10.1109/TPAMI.2026.3679726.

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

A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023b. URL https://arxiv.org/abs/2307.15043.

# A Appendix

## A.1 Proofs

**Theorem 1** (Approximation ratio). *Fix an instance $(x, y)$ and $\delta \in [0, 1)$ for which (4) is feasible, and let $\hat{k}$ be the number of vectors returned by Algorithm 1 with an $\alpha$-approximate joint attack. Then*

$$k^\star \;\le\; \hat{k} \;\le\; \Big(1 + \frac{\ln \tau}{\alpha}\Big) k^\star, \tag{6}$$

*and the same ratio holds between the estimated and the true systemic robustness $\hat{k}/M$ and $k^\star/M$. Being uniform over instances, it carries over to the average over $\mathcal{D}$.*

*Proof.* The lower bound holds because $X$ satisfies the constraint in (4). For the upper bound let $X^\star$ attain $k^\star$ and set $U^\star := \bigcup_{x' \in X^\star} \mathcal{C}(x')$, so $|U^\star| \ge \tau$. Write $\mathcal{B}_t$ for the covered set after $t$ iterations and $r_t := \max\{0, \tau - |\mathcal{B}_t|\}$ for the remaining requirement, with $r_0 = \tau$. As noted above the fallback is never executed, so at iteration $t$ the uncovered set is $\mathcal{I}_{t-1} = [M] \setminus \mathcal{B}_{t-1}$ and contains at least $|U^\star| - |\mathcal{B}_{t-1}| \ge r_{t-1}$ elements of $U^\star$. These are covered by the $k^\star$ vectors of $X^\star$, so some $x^\star \in X^\star$ satisfies $|\mathcal{C}(x^\star) \cap \mathcal{I}_{t-1}| \ge r_{t-1}/k^\star$, and by Definition 2 the accepted vector covers $|F_t| \ge \alpha\, r_{t-1}/k^\star$ new models. Hence $r_t \le r_{t-1}(1 - \alpha/k^\star)$ and, iterating, $r_t \le \tau(1 - \alpha/k^\star)^t < \tau e^{-\alpha t/k^\star}$ for all $t \ge 1$. The right-hand side is at most 1 once $t \ge (k^\star/\alpha) \ln \tau$, at which point $r_t < 1$ and, being an integer, $r_t = 0$. The loop therefore stops at $\hat{k} \le \max\{1, \lceil (k^\star/\alpha) \ln \tau \rceil\} \le k^\star + (k^\star/\alpha) \ln \tau$, using $k^\star \ge 1$. □

## A.2 Systemic Risk Evaluation Algorithms

**Joint PGD for Image Classification Models**.

For an image $x \in [0, 1]^d$, with pixel intensities scaled to $[0, 1]$, the admissible perturbation set is the $\ell_\infty$ ball $\mathcal{B}_\varepsilon(x)$ within the valid pixel range,

$$\mathcal{B}_\varepsilon(x) = \big\{\, x' \in [0, 1]^d : \|x' - x\|_\infty \le \varepsilon \,\big\}. \tag{7}$$

**Objective.** Let $z_j(x') \in \mathbb{R}^{|\mathcal{Y}|}$ denote the logits of $f_j$, so that $f_j(x') = \arg\max_c z_j(x')_c$. Given an instance $x$ and a set of uncovered models $\mathcal{I} \subseteq [M]$, joint PGD seeks a single admissible input that compromises every uncovered model at once, by maximizing their mean cross-entropy with respect to their own clean predictions:

$$\max_{x' \in T_x} \mathcal{L}_\mathcal{I}(x'), \qquad \mathcal{L}_\mathcal{I}(x') = \frac{1}{|\mathcal{I}|} \sum_{j \in \mathcal{I}} \mathrm{CE}\big(z_j(x'),\, f_j(x)\big). \tag{8}$$

This is an untargeted, white-box attack: every model in $\mathcal{I}$ contributes an equal share of the gradient, and models outside $\mathcal{I}$, which are already covered, contribute nothing.

**Optimization.** We solve (8) with $\ell_\infty$ projected gradient ascent. Let $\Pi_{\mathcal{B}_\varepsilon(x)}$ be the projection onto $\mathcal{B}_\varepsilon(x)$, which clips each coordinate $x'_i$ to $[-\varepsilon, \varepsilon]$ around $x_i$ and then to $[0, 1]$. Both constraints are coordinate-wise intervals that intersect, so these two clips give the exact projection. Starting from a uniform random point $x'^{(0)} = \Pi_{\mathcal{B}_\varepsilon(x)}(x + u)$ with $u \sim \mathcal{U}([-\varepsilon, \varepsilon]^d)$, each step is

$$x'^{(t+1)} = \Pi_{\mathcal{B}_\varepsilon(x)}\Big(x'^{(t)} + \eta\, \mathrm{sign}\big(\nabla_{x'}\, \mathcal{L}_\mathcal{I}(x'^{(t)})\big)\Big), \qquad t = 0, \ldots, n_{\mathrm{it}} - 1. \tag{9}$$

We run $n_{\mathrm{rs}}$ independent random restarts and return the final iterate of the restart with the highest objective $\mathcal{L}_\mathcal{I}(x'^{(n_{\mathrm{it}})})$ (Algorithm 2). MNIST attack hyperparameters are shown in Table 2.

**Joint BEAST for Large Language Models**.

For LLMs, an instance $x$ is a harmful request from AdvBench, paired with an affirmative target response $y_x$ (e.g. "Sure, here is …"). The attacker appends an adversarial suffix $s$ to the request, written $x \oplus s$ while adhering to an attack budget $L$. A suffix $s$ is conforming with the budget constraint iff $|s| \le L$ under a fixed canonical tokenizer.

**Algorithm 2** Joint PGD attack $\mathcal{A}_{\mathcal{I}}(x) = \textsc{JointPGD}(x, \mathcal{I})$

**Require:** instance $x \in [0,1]^d$; active models $\mathcal{I} \subseteq [M]$; budget $\varepsilon$; steps $n_{\text{it}}$; step size $\eta$; restarts $n_{\text{rs}}$

$\ell^\star \leftarrow -\infty$
**for** $r = 1, \dots, n_{\text{rs}}$ **do**
  $x' \leftarrow \Pi_{T_x}(x + u), \quad u \sim \mathcal{U}([-\varepsilon, \varepsilon]^d)$
  **for** $t = 1, \dots, n_{\text{it}}$ **do**
    $x' \leftarrow \Pi_{T_x}\big(x' + \eta \ \text{sign}(\nabla_{x'} \mathcal{L}_{\mathcal{I}}(x'))\big)$
  **if** $\mathcal{L}_{\mathcal{I}}(x') > \ell^\star$ **then**
    $x^\star \leftarrow x'; \quad \ell^\star \leftarrow \mathcal{L}_{\mathcal{I}}(x')$
**return** $x^\star$

Table 2: Hyperparameters of joint PGD on MNIST.

| Hyperparameter | Value |
|---|---|
| Steps $n_{\text{it}}$ | 40 |
| Step size $\eta$ | $2.5\,\varepsilon/n_{\text{it}} = \varepsilon/16$ |
| Random restarts $n_{\text{rs}}$ | 3 |

**Objective.** Each model $f_j$ is an autoregressive LLM with its own tokenizer. Let $p_j(\cdot \mid \cdot)$ be its next-token distribution. Following BEAST (Sadasivan et al., 2024), we optimise a surrogate: the per-token negative log-likelihood of the target response,

$$\ell_j(x') = -\frac{1}{|y_x|_j} \sum_{t=1}^{|y_x|_j} \log p_j\big(y_{x,t} \,\big|\, c_j(x'),\, y_{x,<t}\big), \tag{10}$$

where $y_x$ is tokenized with $f_j$'s own tokenizer and $|y_x|_j$ is its length. Given an instance $x$ and a set of uncovered models $\mathcal{I} \subseteq [M]$, joint BEAST seeks a suffix that makes every uncovered model likely to begin its answer with the target, by minimizing their mean target negative log likelihood (NLL).

**Optimization.** We optimize for the NLL with a gradient-free beam search over suffix tokens in which every active model proposes candidates (Algorithm 3). The beam starts from $B$ seed suffixes of $L_0$ tokens, generated for the request $x$ by AmpleGCG (Liao and Sun, 2024). The search then extends every suffix by one token per step for $L - L_0$ steps. At each step, every pair of beam suffix $s$ and active model $j \in \mathcal{I}$ proposes $n_{j,s}$ distinct tokens, sampled without replacement from

$$q_{j,s} = \text{softmax}\big(z_j(x \oplus s)/T\big), \tag{11}$$

where $z_j(x \oplus s)$ are the logits of $f_j$ at the last position of $c_j(x \oplus s)$, that is, where its response would begin. Each proposal is decoded to text, re-encoded with the canonical tokenizer, appended to $s$, and truncated to $L_0 + t$ canonical tokens. The quotas $n_{j,s}$ are split as evenly as possible over all $B\,|\mathcal{I}|$ source pairs, so that no single model dominates the candidate set. Every candidate is then scored on every active model, and the $B$ candidates with the lowest mean NLL $\mathcal{L}_{\mathcal{I}}$ form the next beam. After the final step, the attack returns all $B$ suffixes in the beam, not only the best one.

Hyperparameter details for all jointBEAST experiments can be found in Table 3.

**Joint GCG for Large Language Models**.

For joint GCG, an instance $x$ is a harmful request from HarmBench. As for joint BEAST, the attacker appends an adversarial suffix $s$ to the request, written $x \oplus s$. Unlike joint BEAST, the suffix length is not constrained explicitly: it is set by the initial suffix, and each optimization step substitutes a single token.

**Objective.** Joint GCG optimises an untargeted surrogate. Instead of pulling each model towards an affirmative target, it pushes each model away from a fixed refusal $r$ ("I cannot assist with that request."). For model $f_j$, the surrogate is the per-token negative log-likelihood of the refusal. Given an instance $x$ and a set of uncovered models $\mathcal{I} \subseteq [M]$, joint GCG seeks a suffix that makes every

**Algorithm 3** Joint BEAST attack $\mathcal{A}_{\mathcal{I}}(x) = \textsc{JointBeast}(x, \mathcal{I})$

**Require:** instance $x$ with target $y_x$; active models $\mathcal{I} \subseteq [M]$; seed suffixes $s_1, \ldots, s_B$ of $L_0$ tokens; suffix length $L$; query budget $Q$; temperature $T$

$\mathcal{S} \leftarrow \{s_1, \ldots, s_B\}$; score each on every $j \in \mathcal{I}$
**for** $t = 1, \ldots, L - L_0$ **do**
  $C_t \leftarrow$ number of candidates the budget allots to step $t$
  $\mathcal{K} \leftarrow \emptyset$
  **for** each $s \in \mathcal{S}$ and $j \in \mathcal{I}$ **do**
    draw $n_{j,s}$ distinct tokens $v \sim q_{j,s}$ ▷ (11)
    add $s \oplus v$, truncated to $L_0 + t$ tokens, to $\mathcal{K}$
  remove duplicates, keeping $C_t$ candidates in $\mathcal{K}$
  $\mathcal{S} \leftarrow$ the $B$ suffixes $s \in \mathcal{K}$ with the lowest $\mathcal{L}_{\mathcal{I}}(x \oplus s)$
**return** $\{ x \oplus s : s \in \mathcal{S} \}$

Table 3: Hyperparameters of joint BEAST.

| Hyperparameter | Value |
|---|---|
| Beam width $B$ | 10 |
| Seed length $L_0$ | 20 tokens |
| Suffix length $L$ | 30 tokens (as per canonical tokenizer) |
| Seed generator and canonical tokenizer | AmpleGCG-llama2-sourced-llama2-7b-chat |
| Temperature $T$ | 1 |
| Query budget per call $Q$ | 15,000 |
| Maximum calls $n_{\text{calls}}$ | 4 (60,000 search queries) |
| Samples | 25 AdvBench behaviors |
| Judge | HarmBench-Llama-2-13b-cls |

uncovered model unlikely to refuse, by maximizing their mean refusal NLL:

$$\max_{s} \ \mathcal{L}_{\mathcal{I}}(x \oplus s), \qquad \mathcal{L}_{\mathcal{I}}(x') = \frac{1}{|\mathcal{I}|} \sum_{j \in \mathcal{I}} \ell_j(x'). \tag{12}$$

**Optimization.** We optimize (12) with greedy coordinate gradient (GCG) search (Zou et al., 2023a), adapted to a pool of models that do not share a tokenizer (Algorithm 4). Suffixes are therefore exchanged between models as text and never as token ids. At each step, one model $j \in \mathcal{I}$ proposes candidates, and the proposing model rotates over $\mathcal{I}$ so that every active model proposes for an (almost) equal number of steps. The proposing model tokenizes the current suffix $s$ with its own tokenizer as $(v_1, \ldots, v_n)$ and computes the gradient of its surrogate $\tilde{\ell}_j$ with respect to the one-hot encoding $e_i \in \{0,1\}^{|\mathcal{V}_j|}$ of each suffix token $v_i$. Each of $n_{\text{cand}}$ candidates is formed by choosing a position $i$ uniformly at random and replacing $v_i$ with a token drawn uniformly from the $n_{\text{top}}$ tokens with the largest gradient entries at that position. The candidate with the highest mean refusal NLL $\mathcal{L}_{\mathcal{I}}$ replaces the current suffix if it improves on the best objective found so far.

In all joint GCG experiments we use the hyperparameter settings shown in Table 4.

**Joint AutoDAN for Large Language Models**.

For joint AutoDAN, an instance $x$ is a harmful request from HarmBench. Unlike joint BEAST and joint GCG, the attack does not append a suffix. Instead, an attacker LLM $g$ rewrites the request into a complete natural-language prompt $u$ that preserves its objective. A prompt is admissible if it is at most 6,000 characters long, reads as ordinary natural language (at least five words and at least 85% ASCII characters), and does not duplicate an earlier prompt up to whitespace and case.

**Objective.** The attack treats the target models as black boxes and uses only their text responses. For each prompt $u$, every model in the pool generates a greedy response $f_j(u)$, which the HarmBench classifier labels with a binary outcome $b_j(u) \in \{0,1\}$ and a confidence $\sigma_j(u) \in [0,1]$. The confidence is the classifier's normalised probability of answering "yes" or "no" at its first output token. Given an instance $x$ and a set of models $\mathcal{I} \subseteq [M]$, joint AutoDAN seeks a prompt that compromises

**Algorithm 4** Joint GCG attack $\mathcal{A}_{\mathcal{I}}(x) = \textsc{JointGCG}(x, \mathcal{I})$

**Require:** instance $x$; active models $\mathcal{I} \subseteq [M]$; initial suffix $s_0$; steps $n_{\text{it}}$; candidates per step $n_{\text{cand}}$; top-$n_{\text{top}}$ substitutions

$s \leftarrow s_0$; $\quad \ell^\star \leftarrow -\infty$
**for** $t = 1, \dots, n_{\text{it}}$ **do**
  $j \leftarrow$ proposing model for step $t$ ▷ *rotates over* $\mathcal{I}$
  tokenize $s$ with $f_j$'s tokenizer as $(v_1, \dots, v_n)$
  $g_i \leftarrow \nabla_{e_i} \tilde{\ell}_j(x \oplus s)$ for $i = 1, \dots, n$
  $\mathcal{K} \leftarrow \{s\}$
  **for** $c = 1, \dots, n_{\text{cand}}$ **do**
    $i \sim \mathcal{U}(\{1, \dots, n\})$; $\quad v \sim \mathcal{U}\big(\text{top}_{n_{\text{top}}}(g_i)\big)$
    add the text of $(v_1, \dots, v_{i-1}, v, v_{i+1}, \dots, v_n)$ to $\mathcal{K}$
  $s' \leftarrow \arg\max_{u \in \mathcal{K}} \tilde{\mathcal{L}}_{\mathcal{I}}(x \oplus u)$
  **if** $\tilde{\mathcal{L}}_{\mathcal{I}}(x \oplus s') > \ell^\star$ **then**
    $s \leftarrow s'$; $\quad \ell^\star \leftarrow \tilde{\mathcal{L}}_{\mathcal{I}}(x \oplus s')$
**return** $x \oplus s$

Table 4: Hyperparameters of joint GCG.

| Hyperparameter | Value |
|---|---|
| Steps per call $n_{\text{it}}$ | 20 |
| Candidates per step $n_{\text{cand}}$ | 64 |
| Top-$n_{\text{top}}$ substitutions | 256 |
| Samples | 25 HarmBench behaviors |
| Judge | HarmBench-Llama-2-13b-cls |

as many models in $\mathcal{I}$ as possible. It ranks prompts lexicographically by

$$R_{\mathcal{I}}(u) = \Big( \sum_{j \in \mathcal{I}} b_j(u), \quad \frac{1}{|\mathcal{I}|} \sum_{j \in \mathcal{I}} \sigma_j(u), \quad q_{0.2}\big(\{\sigma_j(u)\}_{j \in \mathcal{I}}\big) \Big), \tag{13}$$

that is, by the number of compromised models, then the mean confidence, then the lower 20% quantile of the confidences. The mean and the quantile break ties between prompts that compromise the same number of models, and favour prompts that move the least-compromised models.

**Optimization.** We optimize (13) with a strategy-guided search adapted from AutoDAN-Turbo (Liu et al., 2025), in which the attacker revises prompts using feedback from one *anchor* model at a in a round robin manner (Algorithm 5). The attacker is guided by a frozen library of abstract jailbreak strategies $\Sigma$ from the AutoDAN-Turbo warm-up library. The search first evaluates the unmodified request $x$ on every model. The attacker then produces a cold-start rewrite of $x$ without strategy guidance, which becomes the first candidate. The anchor then cycles once over all $M$ models in the pool. For anchor $a$, two parents are selected from all candidates found so far: the *systemic* parent $u^{\text{sys}}$, which maximizes $R_{\mathcal{I}}$, and the *specialist* parent $u^{\text{spec}}$, which is best on the anchor alone, ranked by $(b_a, \sigma_a, R_{\mathcal{I}})$. The strategies in $\Sigma$ are ranked by lexical cosine similarity between their definitions and the request, the systemic parent, and the anchor's response to it. The attacker then generates $n_{\text{p}}$ proposals. Proposals alternate between the systemic and the specialist parent. Each of the first $n_{\text{p}} - 1$ proposals applies one of the top-ranked strategies, and the last combines the top two. For every proposal, the attacker receives the request, the parent prompt, the anchor's response to the systemic parent, whether that response was judged harmful, and the strategy definitions. It returns a revised prompt, which is resampled up to three times if it is not admissible. Every proposal is evaluated on every model in the pool and added to the candidates.

We show further hyperparameter details used in joint AutoDAN experiments in Table 5.

**Algorithm 5** Joint AutoDAN attack $\mathcal{A}_{\mathcal{I}}(x) = \textsc{JointAutoDAN}(x, \mathcal{I})$

**Require:** instance $x$; objective models $\mathcal{I} \subseteq [M]$; attacker $g$; strategy library $\Sigma$; proposals per anchor $n_{\mathrm{p}}$; systemic archive size $n_{\mathrm{sys}}$

evaluate $x$ on every $f_j$, $j \in [M]$
$a_1, \ldots, a_M \leftarrow$ cyclic order of $[M]$
$u_0 \leftarrow g\big(x, f_{a_1}(x)\big)$; evaluate $u_0$ on every $f_j$; $\quad \mathcal{P} \leftarrow \{u_0\}$
**for** $m = 1, \ldots, M$ **do**
  $u^{\mathrm{sys}} \leftarrow \arg\max_{u \in \mathcal{P}} R_{\mathcal{I}}(u)$
  $u^{\mathrm{spec}} \leftarrow \arg\max_{u \in \mathcal{P}} \big(b_{a_m}(u), \sigma_{a_m}(u), R_{\mathcal{I}}(u)\big)$
  $\sigma_1, \sigma_2, \cdots \leftarrow$ strategies in $\Sigma$ ranked by similarity to $\big(x, u^{\mathrm{sys}}, f_{a_m}(u^{\mathrm{sys}})\big)$
  **for** $i = 1, \ldots, n_{\mathrm{p}}$ **do**
    $\bar{u} \leftarrow u^{\mathrm{sys}}$ if $i$ is odd, else $u^{\mathrm{spec}}$
    $S \leftarrow \{\sigma_i\}$ if $i < n_{\mathrm{p}}$, else $\{\sigma_1, \sigma_2\}$
    $u \leftarrow g\big(x, \bar{u}, f_{a_m}(u^{\mathrm{sys}}), b_{a_m}(u^{\mathrm{sys}}), S\big)$
    evaluate $u$ on every $f_j$, $j \in [M]$; $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{u\}$
**return** the top $n_{\mathrm{sys}}$ prompts in $\mathcal{P}$ under $R_{\mathcal{I}}$, and the best $u \in \mathcal{P}$ for each $j \in \mathcal{I}$

Table 5: Hyperparameters of joint AutoDAN.

| Hyperparameter | Value |
|---|---|
| Attacker model $g$ | Qwen3.8-27B |
| Sampling temperature | 0.8 |
| Top-$p$ | 0.95 |
| Maximum new tokens | 768 |
| Resampling attempts | 3 |
| Proposals per anchor $n_{\mathrm{p}}$ | 4 |
| Systemic archive size $n_{\mathrm{sys}}$ | 2 |
| Samples | 15 HarmBench behaviors |
| Judge | HarmBench-Llama-2-13b-cls |

### A.3 Further Experimental Details

#### A.3.1 MNIST

For MNIST experiments, the base model pools are trained once, their configurations persisted and then reused across all experiments. All models, irrespective of pool, are gradient based and thus trained via standard gradient descent, using Adam, specifically. All models are trained to convergence via an early stopping mechanism, triggered when a held out validation loss starts to plateau (with a patience of 10 train steps). After early stopping is triggered, we recover the best performing epoch checkpoint.

Further details on the MNIST model pool composition can be found in Table 6.

#### A.3.2 Large Language Models

Our experimental evaluation for large language models focuses on two distinct model pools. Both with roughly similar complexity in terms of model size but varying underlying foundation models. The *Opus* pool consists of the following eight models: Nemotron-Cascade-8B-Thinking, Nemotron-Cascade-14B-Thinking, granite-4.0-h-tiny, GLM-4.7-Flash, gpt-oss-20b, Qwen3-4B-Thinking-2507-unsloth-bnb-4bit, Qwen3-8B-unsloth-bnb-4bit, Qwen3-14B. The *Deepseek* pool is comprised of the below six models: Qwen2.5-Math-1.5B, Qwen2.5-Math-7B, Qwen2.5-14B, Qwen2.5-32B, Qwen3-8B-Base, Llama-3.1-8B. See Table 7 for an overview. As the pool names suggest, the former is distilled on Opus 4.5 and the latter on DeepSeek-R1 to evaluate the impact of distillation on systemic robustness.

Table 6: MNIST model pools. Each pool has 20 models, and each diversity level keeps what varies in the level below and adds one more factor. Every model is trained with early stopping and the best-validation weights are kept.

| Pool | Architecture | Training data and regularisation | Varies across models |
|---|---|---|---|
| Low | MLPs, one hidden layer (256 units) | Shared training splits in fixed order, no augmentation or regularisation | Initialisation seed |
| Medium | MLPs, one hidden layer (256 units) | diverse resampling of training split, random affine augmentation, weight decay | Seed, training regime |
| High | 6 two-layer CNNs, 6 three-layer CNNs, 4 one-hidden-layer MLPs, 4 *diverse* CNN variants | Same as Medium | Seed, training regime, architecture family |

Table 7: LLM pools. Parameters are the minimum–maximum over the pool, with the median in parentheses.

| Pool | $M$ | Params (B) | Post-training |
|---|---|---|---|
| Opus parents | 8 | 4.0–31.2 (11.5) | 8 instruction- or reasoning-tuned |
| Opus descendants | 8 | 4.0–31.2 (11.5) | 8 distilled from reasoning traces attributed to Claude Opus 4.5 |
| DeepSeek-R1 parents | 6 | 1.5–32.8 (8.1) | 6 base, not safety-aligned |
| DeepSeek-R1 descendants | 6 | 1.8–32.8 (8.1) | 6 distilled from DeepSeek-R1 |