

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

SoK: Colluding Adversaries in Machine Learning Pipelines

Vasisht Duddu, Lipeng He, and Asim Waheed, *University of Waterloo;*
N. Asokan, *University of Waterloo and KTH Royal Institute of Technology*

https://www.usenix.org/conference/usenixsecurity26/presentation/duddu

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

ARTIFACT EVALUATED
USENIX
AVAILABLE

SoK: Colluding Adversaries in Machine Learning Pipelines

Vasisht Duddu[1], Lipeng He[1], Asim Waheed[1], N. Asokan[1,2]
[1]*University of Waterloo,* [2]*KTH Royal Institute of Technology*
{vasisht.duddu, lipeng.he, asim.waheed}@uwaterloo.ca, asokan@acm.org

Abstract

Machine learning (ML) models are susceptible to various security, privacy, and fairness risks. Adversaries with different characteristics (i.e., objectives, knowledge, and capabilities) *can collude* by executing one attack to amplify others. Existing work lacks a systematic framework to explore collusion among adversaries, and to study the implications of the adversaries' characteristics. We present a framework covering collusion (a) between train- and inference-time adversaries, and (b) among inference-time adversaries[1]. Our framework accounts for *factors enabling collusion* between adversaries. We propose a guideline to conjecture about the potential for collusion using enabling factors. We use it to explain prior work, conjecture about unexplored collusions, and empirically validate five such cases. Finally, we discuss how adversaries' characteristics influence the potential for collusion.

1 Introduction

Machine learning (ML) models are susceptible to a wide range of risks to security [23, 32, 75, 80, 96], privacy [84, 86], and fairness [69]. Prior work explored attacks to exploit vulnerabilities associated with each of these risks. However, adversaries with different *characteristics* (i.e., objectives, knowledge, and capabilities) in the ML pipeline, either with different roles (e.g., model trainer and client) or with the same role (e.g., clients with different objectives), *can collude*. Collusion is the explicit coordination among adversaries to execute one attack and use the outcomes to improve another (e.g., [101]). Collusion among adversaries with different objectives is well documented in other domains: cryptocurrency pump-and-dump schemes where one adversary inflates a stock, which can be exploited by others for profit [38]; and cyber-crime markets where an adversary sells compromised data for downstream attacks like identity theft [47]. Similar collusion within ML pipelines is feasible, and merits investigation.

[1]**Terminology**: We hereafter use "test-time" instead of "inference-time".

For instance, consider two different adversaries: $\mathcal{A}dv1$ seeking to find an adversarial example against a target model $\mathcal{M}$ and $\mathcal{A}dv2$ seeking to build a shadow model of $\mathcal{M}$ either for some privacy attack [60], or to steal $\mathcal{M}$'s functionality [76]. $\mathcal{A}dv1$ sends multiple queries to $\mathcal{M}$ until it successfully finds an adversarial example [4], usually discarding all other query-response pairs. But, they are valuable for $\mathcal{A}dv2$. Therefore, $\mathcal{A}dv1$ can either sell the query-response data to $\mathcal{A}dv2$ or share the cost of querying $\mathcal{M}$. Hence, adversaries are incentivized to collude (a) via a marketplace for selling attack information, or (b) by sharing costs to avoid duplication of efforts.

Foreseeing all possible cases of collusion is challenging. A unified framework can help *practitioners* and *researchers* to identify new threats, design strong attacks to audit models, and design defenses against them. Ideally, the framework should be (i) *comprehensive* (captures an adversaries' characteristics *across multiple risks*) and (ii) *extensible* (allows inclusion of additional risks and characteristics).

We present the *first* such framework that includes various factors aiding the potential for collusion. Our framework covers two collusion types identified from prior work: (i) between train- and test-time adversaries ($\rightarrow$ "*train-test collusion*"), and (ii) among test-time adversaries ($\rightarrow$ "*test-time collusion*"). For both, we observe that a successful attack changes some factors as an outcome, which can influence the effectiveness of other attacks relying on those factors. However, these factors differ between both collusion types. For *train–test collusion*, poisoning changes overfitting and memorization which can impact the effectiveness of test-time attacks that rely on those factors. Here, the framework includes the factors related to overfitting and memorization, that can be manipulated by an adversary. For *test-time collusion*, knowledge inferred from the first attack is an outcome that satisfies the prerequisites of other attacks. Here, the framework includes factors related to the adversary's knowledge.

Using this framework, we propose a guideline for conjecturing about collusion potential by identifying factors that are shared across attacks. We empirically validate our conjectures for five unexplored cases of collusion. Finally, we

explore how an adversary's characteristics impact collusion potential and discuss implications for our guideline. We claim the following contributions: we

1. present the *first framework* to systematically explore collusion potential using various *factors*, and a *guideline* to conjecture about potential collusion; (§3)
2. use our guideline to *explain collusion* in prior work identified from a *literature survey*; (§4)
3. identify previously *unexplored cases of collusion*, use our guideline to conjecture about them, and empirically validate five such conjectured cases; (§5) and
4. discuss the *influence* of the adversary's characteristics on the collusion potential. (§6)

2 Background and Related Work

Notations: An ML model ($\mathcal{M}_\theta$) is a function parameterized by θ that takes an input x (e.g., images, tabular data, graphs, or text prompts) and produces $\mathcal{M}_\theta(x)$. For classification tasks, $\mathcal{M}_\theta(x)$ is a classification label, while for generative tasks, $\mathcal{M}_\theta(x)$ is an image or text. Hereafter, we denote $\mathcal{M}_\theta$ by simply writing $\mathcal{M}$. We assume training data ($\mathcal{D}_{tr}$) follows certain distributional properties (e.g., a specific female-to-male ratio). We train $\mathcal{M}$ by updating θ to minimize the loss between $\mathcal{M}(x)$ and the expected output y. After training, we evaluate the utility on a test dataset ($\mathcal{D}_{te}$), where we use accuracy for classifiers, and the quality of generated images (e.g., using Fréchet Inception Distance) and text (e.g., using perplexity) for generative models. We refer to predictions, intermediate activations, θ, and gradients as *model observables*.

Attacks against ML Models: Following the taxonomy in Duddu et al. [29], we consider an adversary ($\mathcal{A}dv$) executes an attack by exploiting vulnerabilities corresponding to security, privacy, and fairness risks:

- **Evasion (`Evsn`)** modifies the inputs to force $\mathcal{M}$ to misclassify (classifiers) [80,96], or generate unacceptable images/text (generative models) [82, 119]. In classifiers, these modified inputs are called adversarial examples ($x_{adv} = x + \delta$) obtained by adding perturbations δ to x where δ is computed to maximize $\mathcal{M}$'s loss. Similarly, in generative models, we generate adversarial prompts but to evade safeguards against unacceptable content (or jailbreaking).
- **Poisoning (`Poisn`)** trains $\mathcal{M}$ using malicious data records (*poisons*) injected in $\mathcal{D}_{tr}$ to degrade $\mathcal{M}$'s utility on $\mathcal{D}_{te}$ [23]. A variant of poisoning is **backdoors (`Bkdr`)** where $\mathcal{M}$ is trained with poisons embedded with a specific pattern (aka *trigger*), which maps an input to an $\mathcal{A}dv$-chosen output. Backdoored inputs will generate $\mathcal{A}dv$-chosen output, but clean inputs will result in normal behavior [2, 57, 128]. We refer to both of them as `Poisn` for brevity.
- **Unauthorized Model Ownership (`ModExt`)** occurs when $\mathcal{A}dv$ obtains either an identical copy of $\mathcal{M}$ or derives a surrogate model $\mathcal{M}_{sur}$ from $\mathcal{M}$ using model extraction attacks [14, 53, 76, 99]. We focus on the latter setting.
- **Membership Inference (`MemInf`)** identifies if an input was in $\mathcal{D}_{tr}$, by exploiting the difference in $\mathcal{M}$'s behavior on data records inside and outside $\mathcal{D}_{tr}$ [27, 28, 84].
- **Attribute Inference (`AttInf`)** identifies the value of an input's sensitive attribute, by exploiting the difference in $\mathcal{M}$'s behavior on inputs with different attribute values [49]. In generative models, this translates to extracting personally identifiable information [62].
- **Data reconstruction (`DtRecon`)** recovers inputs or data records from $\mathcal{D}_{tr}$ using access to $\mathcal{M}$ [41, 114, 124, 131]. For generative models, $\mathcal{M}$ can generate outputs which either replicate or closely resemble confidential $\mathcal{D}_{tr}$ [3, 10, 13].
- **Distribution Inference (`DistInf`)** identifies the distributional properties of $\mathcal{D}_{tr}$ by exploiting the difference in $\mathcal{M}$'s behavior when trained with different distribution values [70, 94, 95, 103, 130]. For instance, it differentiates between $\mathcal{M}$ trained on $\mathcal{D}_{tr}$ with a male-to-female ratio of 0.1 vs. $\mathcal{M}$ trained on $\mathcal{D}_{tr}$ with a ratio of 0.9.
- **Discriminatory Behavior (`Discr`)**[2] is observed when $\mathcal{M}$ behaves differently across demographic subgroups identified by attributes such as race or gender on $\mathcal{D}_{te}$ [40, 61, 69]. This is measured using the difference in accuracy or false positive/negative rates across different subgroups.

There are different attacks corresponding to each risk but for brevity, we use the notations for risks as attacks and categorize them as *train-time* (i.e., `Poisn`) and *test-time* (i.e., `Evsn`, `MemInf`, `AttInf`, `DistInf`, `DtRecon`, `ModExt`, and `Discr`).

Comparison to Prior Surveys/SoKs: Several surveys and SoKs study individual risks to ML models in isolation, including evasion [80, 96], unfairness [69], privacy [84], backdoors [2], poisoning [23], and unauthorized model ownership [75]. Salem et al. [86] relate multiple privacy risks using game-based formulations, though without considering collusion among adversaries. A separate line of work explores trade-offs among defenses and risks, where defenses designed for one risk may inadvertently amplify others [29, 36, 71, 93, 116]. Other works have explored trade-offs among defenses such as fairness, interpretability, privacy, and robustness across diverse settings [18, 35, 42], and explore how to combine defenses without conflict to mitigate multiple risks simultaneously [30, 97]. Prior works either focus on systematizing individual risks or trade-offs induced by the defender's choices. This includes unintended interactions of defenses with risks (e.g., [29, 71]) or conflicting interactions among defenses (e.g., [30, 97]). None of the prior works systematize how adversaries may collude with each other.

[2] Unlike other risks that involve an explicit $\mathcal{A}dv$, `Discr` does not require one. However, by assuming a test-time $\mathcal{A}dv$ who measures `Discr`, we can study collusion where a train-time $\mathcal{A}dv$ increases `Discr`, or a test-time $\mathcal{A}dv$ uses knowledge from `Discr` for other risks (§4).

3 Our Framework

Our goal is to develop a framework to explore *collusion potential among adversaries*, and study the influence of $\mathcal{A}dv$'s characteristics on the collusion potential. We identify the following requirements for an ideal framework: **R1 Comprehensive** (captures adversaries' characteristics *across multiple risks, model types, and settings*); and **R2 Extensible** (allows inclusion of additional risks and characteristics). We present our framework which includes: (i) factors aiding the collusion potential among adversaries (§3.1), (ii) relation between attack effectiveness and factors (§3.2), and (iii) a guideline to conjecture about the collusion potential (§3.3). We then study the influence of $\mathcal{A}dv$'s characteristics on collusion (§6). Later in §7, we discuss how our framework meets **R1** and **R2**.

3.1 Factors Aiding Collusion Potential

For illustration, we consider collusion among two adversaries ($\mathcal{A}dv1$ and $\mathcal{A}dv2$), and discuss extending beyond two adversaries in §7. Based on prior work (§4), we identify two collusion types and discuss the factors aiding their potential:

- **Train-test Collusion:** $\mathcal{A}dv1$ can specifically optimize the train-time attacks, to increase the effectiveness of a test-time attack (as $\mathbf{Att}_2$) executed by $\mathcal{A}dv2$.
- **Test-time Collusion:** $\mathcal{A}dv1$ first executes a test-time attack (as $\mathbf{Att}_1$), and uses the knowledge inferred to amplify another test-time attack (as $\mathbf{Att}_2$) executed by $\mathcal{A}dv2$.

Train-Test Collusion: We conjecture that train-time attacks, (`Poisn`) can be optimized to influence various factors underlying overfitting and memorization as an outcome (e.g., increase tail length of $\mathcal{D}_{tr}$'s distribution, distinguishability in model observables across subgroups or datasets). This can impact the effectiveness of test-time attacks that rely on those factors. Concretely, $\mathcal{A}dv1$ can optimize `Poisn` to increase memorization and thereby the distinguishability between (a) training and non-training data records (to increase effectiveness of `MemInf` [101]), (b) across subgroups (to increase effectiveness of `AttInf` [101]), or (c) across datasets with different distributions (to increase effectiveness of `DistInf` [16]). Hence, our framework includes factors which influence overfitting and memorization that can be manipulated by $\mathcal{A}dv1$, and exploited during test-time by $\mathcal{A}dv2$.

We use factors influencing overfitting and memorization identified by Duddu et al. [29] who explore an orthogonal problem of how defenses impact susceptibility to unrelated risks. From these, we select factors that $\mathcal{A}dv1$ can control by optimizing `Poisn`, and exclude hyperparameter-related ones[3] (e.g., number of attributes, $\mathcal{D}_{tr}$ size, model capacity) and those which are insufficiently explored (e.g., priority of learning stable attributes, distance to decision boundary). While we use the factors from Duddu et al. [29], our problem requires a fundamentally different analysis and their work cannot be extended to study collusion potential: (i) for `Poisn`, we focus on how `Poisn` affects the factors instead of how the factors affect `Poisn` as in Duddu et al. [29]; and (ii) factors related to test-time collusion (described later) are novel. The framework finally includes the following factors:

`Tr1-Tail` (Tail length of $\mathcal{D}_{tr}$'s distribution): Increasing the tail length of $\mathcal{D}_{tr}$'s distribution can increase the memorization of specific data records by $\mathcal{M}$, thereby increasing the effectiveness of attacks that exploit long-tailed distribution. $\mathcal{A}dv1$ controls this factor by injecting `Poisn` to extend the tail of $\mathcal{D}_{tr}$'s distribution and thus, influence other attacks [29].

`Tr2-Data` (Distinguishability across datasets): $\mathcal{M}$ behaves differently on data records inside and outside $\mathcal{D}_{tr}$ [101], or when trained on datasets with different distributions [15, 16]. This distinguishability can increase the effectiveness of other attacks (e.g., `MemInf` and `DistInf`). $\mathcal{A}dv1$ can control this distinguishability by optimizing `Poisn`.

`Tr3-Subgp` (Distinguishability across subgroups): $\mathcal{M}$ behaves differently on different demographic groups (e.g., race or sex)[4]. This distinguishability can increase the effectiveness of other attacks (e.g., `AttInf`, `DistInf`, `Discr`). $\mathcal{A}dv1$ can control this distinguishability by optimizing `Poisn` [90].

Test-time Collusion: For test-time collusion, $\mathcal{A}dv$ can increase their knowledge about $\mathcal{M}$ and $\mathcal{D}_{tr}$ as outcomes of executing an attack (e.g., distribution of $\mathcal{D}_{tr}$ from `DistInf`, or functionally similar "shadow model" $\mathcal{M}_{shdw}$[5] using `ModExt`), or some *side information* (e.g., adversarial examples as $\mathcal{A}dv$'s auxiliary dataset $\mathcal{D}^{aux}$ from `Evsn`). This knowledge acts as a prerequisite for later attacks (e.g., knowing $\mathcal{D}_{tr}$'s distribution or $\mathcal{M}$'s functionality can improve the effectiveness of other attacks such as `MemInf`, `AttInf`, `DistInf`). There is a potential for collusion if there exist a pair of attacks where the outcomes from $\mathbf{Att}_1$ can meet the prerequisites of $\mathbf{Att}_2$. The factors considered here focus on $\mathcal{A}dv$'s knowledge which differ from those for train–test collusion (and those in Duddu et al. [29]), which do not account for $\mathcal{A}dv$'s knowledge.

`Te1-Aux` (Quality of $\mathcal{A}dv$'s $\mathcal{D}^{aux}$): Attacks that require training $\mathcal{M}_{shdw}$ to mimic $\mathcal{M}$, or an attack model for a specific objective, benefit from having a high-quality $\mathcal{D}^{aux}$ that overlaps with $\mathcal{D}_{tr}$. In practice, $\mathcal{D}^{aux}$ typically does not overlap with $\mathcal{D}_{tr}$. If the overlap with $\mathcal{D}_{tr}$ or the quality of $\mathcal{D}^{aux}$ increases, and the attack transfers effectively to $\mathcal{M}$ (e.g., [73]).

`Te2-Dist` (Similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$): Attacks that require training $\mathcal{M}_{shdw}$ should ensure that its

[3] For `Poisn`, neither `DtOwnr` nor `DtProv` can access or modify model-related hyperparameters. Although `ModProv` may introduce architectural backdoors, model capacity should be maintained to avoid detection. Thus, $\mathcal{A}dv1$ cannot meaningfully manipulate hyperparameters without detection. For test-time attacks, `Clnt` acting as $\mathcal{A}dv1$, does not control any hyperparameters. Therefore, we omit hyperparameter-related factors.

[4] Distinguishability can be across any subgroups/classes but, we use "demographic groups" following prior work on `DistInf`, `Discr`, and `AttInf`.

[5] For some attacks (e.g., `Evsn`, `MemInf`, `AttInf`, `DistInf`, `DtRecon`), $\mathcal{A}dv$ trains $\mathcal{M}_{shdw}$ to mimic $\mathcal{M}$ using $\mathcal{A}dv$'s auxiliary data $\mathcal{D}^{aux}$ either independently or derived via `ModExt`, in which case $\mathcal{M}_{shdw}$ is $\mathcal{M}_{sur}$.

functionality mimics that of $\mathcal{M}$ to achieve better attack transferability. If the distribution similarity between $\mathcal{D}^{aux}$ and $\mathcal{D}_{tr}$ increases, the attack transfers effectively to $\mathcal{M}$ (e.g., [85]).

`Te3-Shdw` (Functional similarity between $\mathcal{M}_{sur}$ and $\mathcal{M}$): Some attacks require training $\mathcal{M}_{sur}$ that mimic $\mathcal{M}$'s functionality. Hence, if $\mathcal{A}dv$'s knowledge about $\mathcal{M}$'s functionality increases, $\mathcal{M}_{sur}$ will be more similar to $\mathcal{M}$, and the attack transfers effectively to $\mathcal{M}$ (e.g., [85]).

`Te4-Arch` (Architecture similarity between $\mathcal{M}_{sur}$ and $\mathcal{M}$): In real-world settings, $\mathcal{A}dv$ may not know $\mathcal{M}$'s architecture and as the knowledge increases, $\mathcal{M}_{sur}$ will be similar to $\mathcal{M}$ and the attack transfers effectively to $\mathcal{M}$ (e.g., [85]).

3.2 Relation between Attacks and Factors

We will now describe the relation between each attack and the various factors for train-test and test-time collusion: (i) how attack effectiveness leads to change in factors; and (ii) how changes in factors correlate with attack effectiveness.

- **Train-test Collusion:** Table 1 shows (i) how the effectiveness of the train-time attack (i.e., `Poisn`) leads to change in factors (left half), and (ii) how changes in factors correlate with the effectiveness of various attacks (right half). On the other hand, the correlation between changes in the factors and the effectiveness of various attacks is reproduced from Duddu et al. [29] (Table 1: right half).
- **Test-time Collusion:** Table 2 shows (i) how the effectiveness of test-time attacks leads to changes in factors (top), and (ii) how changes in factors correlate with the effectiveness of other test-time attacks (bottom).

In both collusion types, we use ↑ (or ↓) to indicate positive (negative) correlation (Tables 1 and 2). For each attack, we describe the relation between the attacks and the enabling factors for train-test and test-time collusion below.

Poisoning (`Poisn`)

This is a train-time attack, and hence, we only discuss the factors enabling train-test collusion. Below, we indicate how the effectiveness of `Poisn` correlates with a change in the factors (in Table 1: left half under "Correlation of attack effectiveness with change in factors").

- ***Train-Test Factors:*** $\mathcal{A}dv$ increases the tail length of $\mathcal{D}_{tr}$'s distribution by adding poisons and backdoors (**`Tr1-Tail`**: ↑) [120]. The attack can be optimized to increase the distinguishability of model observables between (a) data records inside and outside $\mathcal{D}_{tr}$ (**`Tr2-Data`**: ↑) [101], (b) different distributional properties (**`Tr2-Data`**: ↑) [15, 16], and (c) demographic subgroups (**`Tr3-Subgp`**: ↑) [90].

For the remaining attacks, in the case of train–test collusion, we indicate how changes in the factors (due to `Poisn`) correlate with attack effectiveness (Table 1: right half). For test-time collusion, we indicate both: (i) the correlation between attack effectiveness and changes in the factors (Table 2: top); and (ii) the correlation between changes in the factors and attack effectiveness (Table 2: bottom).

Table 1: **(Train-test Collusion) Relation of attacks and factors:** Rows for `Poisn` under "How effectiveness of attack (`Poisn`) leads to change in factors" (left half) indicates how `Poisn` influences a change in factors, instead of how change in factors influence effectiveness of `Poisn` (as in [29]). Rows under "How changes in factors correlate with attack effectiveness" (right half) are reproduced from [29]; only the factors that can be manipulated by $\mathcal{A}dv$ are considered. ↑ (or ↓) indicate positive (negative) correlation; Gray → not applicable; empty cells → no prior literature reported the correlations.

Factors → / Attacks ↓	How effectiveness of attack (`Poisn`) leads to change in factors			How changes in factors correlate with attack effectiveness		
	`Tr1-Tail` (Tail length)	`Tr2-Data` (Datasets)	`Tr3-Subgp` (Subgroups)	`Tr1-Tail` (Tail length)	`Tr2-Data` (Datasets)	`Tr3-Subgp` (Subgroups)
`Poisn`	↑ [120]	↑ [101]	↑ [90]			
`Evsn`				↑ [21, 121]		
`MemInf`				↑ [11, 12]	↑ [56, 132]	
`AttInf`				↑ [49]		↑ [1]
`DistInf`				↑ [15, 16]	↑ [95]	↑ [29]
`DtRecon`				↑ [58]	↑ [117]	↑ [29]
`ModExt`				↓ [112]		
`Discr`				↑ [69]		↑ [1]

Evasion (`Evsn`)

- ***Train-Test Factors:*** As the tail length of $\mathcal{D}_{tr}$ increases, models are more susceptible to adversarial examples particularly from the tail classes (**`Tr1-Tail`**: ↑) [21, 121]. Models perform well on well-represented (head) classes, but struggle to classify under-represented (tail) classes.
- ***Test-time Factors:*** In a blackbox setting where $\mathcal{A}dv$ is required to train $\mathcal{M}_{sur}$, the effectiveness of transferable adversarial examples is positively correlated with quality of $\mathcal{D}^{aux}$ (**`Te1-Aux`**: ↑), similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$ (**`Te2-Dist`**: ↑), and similarity of $\mathcal{M}_{sur}$'s functionality and architecture with $\mathcal{M}$ (**`Te3-Shdw`**: ↑, **`Te4-Arch`**: ↑).

Successful `Evsn` provides adversarial examples as *side knowledge* that can be used as $\mathcal{D}^{aux}$ for other attacks (**`Te1-Aux`**: ↑) [22, 52, 108]. Increasing the effectiveness of `Evsn` results in a better quality $\mathcal{D}^{aux}$.

Membership Test (`MemInf`)

- ***Train-Test Factors:*** As the tail length increases, outlier data records (from the tail classes) are memorized more, resulting in higher susceptibility to `MemInf` (**`Tr1-Tail`**: ↑) [11, 12]. Furthermore, susceptibility to `MemInf` increases with higher distinguishability in model observables across data records inside and outside $\mathcal{D}_{tr}$ (**`Tr2-Data`**: ↑) [87].

Table 2: **(Test-time Collusion) Relation of attacks and factors:** `Discr` is excluded as there are no attacks that $\mathcal{A}dv$ can exploit. ↑ (or ↓) to indicate positive (negative) correlation; Gray → not applicable; empty cells → no prior literature reported the correlations.

Factors → / Attacks ↓	How attack effectiveness leads to change in factors (as "outcomes"): Te1-Aux (Aux. Data)	Te2-Dist (Data Dstrb.)	Te3-Shdw (Model Fnct.)	Te4-Arch (Model Arch.)
Evsn	↑ [52, 132]			
MemInf	↑ [85, 87]			
AttInf		↑ [1]		
DistInf		↑ [94, 95]		
DtRecon	↑ [9, 107]			
ModExt			↑ [52, 76]	↑ [52, 76]

Factors → / Attacks ↓	How change in factors (as "prerequisites") correlate with attack effectiveness			
Evsn	↑ [25, 46, 110]	↑ [25, 46, 110]	↑ [25, 46, 110]	↑ [25, 46, 110]
MemInf	↑ [85, 87]	↑ [85, 87]	↑ [85, 87]	↑ [85, 87]
AttInf	↑ [49, 59]	↑ [49, 59]	↑ [59]	↑ [59]
DistInf	↑ [94, 95]		↑ [94, 95]	↑ [94, 95]
DtRecon	↑ [9, 107]	↑ [9, 107]	↑ [9, 107]	↑ [9, 107]
ModExt	↑ [52, 76]	↑ [52, 76]		

• ***Test-time Factors:*** The effectiveness of `MemInf` is positively correlated with quality of $\mathcal{D}^{aux}$ (**`Te1-Aux`**: ↑), similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$ (**`Te2-Dist`**: ↑), and similarity of $\mathcal{M}_{sur}$'s functionality and architecture with $\mathcal{M}$ (**`Te3-Shdw`**, **`Te4-Arch`**: ↑) [87].

After a successful attack, $\mathcal{A}dv$ identifies data records in $\mathcal{D}_{tr}$, which can act as $\mathcal{D}^{aux}$ for other attacks (**`Te1-Aux`**: ↑). $\mathcal{A}dv$ can accurately infer data records in $\mathcal{D}_{tr}$ as the effectiveness of `MemInf` increases.

Attribute Test (`AttInf`)

• ***Train-Test Factors:*** `AttInf` is more effective as $\mathcal{D}_{tr}$'s tail length increases (**`Tr1-Tail`**: ↑) [49]. `AttInf` exploits the distinguishability in model observables across subgroups. Hence, increasing distinguishability will also make `AttInf` more effective (**`Tr3-Subgp`**: ↑) [1].

• ***Test-time Factors:*** The effectiveness of `AttInf` is positively correlated with quality of $\mathcal{D}^{aux}$ (**`Te1-Aux`**: ↑), similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$ (**`Te2-Dist`**: ↑), and similarity of $\mathcal{M}_{sur}$'s functionality and architecture with $\mathcal{M}$ (**`Te3-Shdw`**, **`Te4-Arch`**: ↑). `AttInf` can help estimate $\mathcal{D}_{tr}$'s distribution when evaluated across several data records sampled i.i.d from $\mathcal{D}_{tr}$ (**`Te2-Dist`**: ↑). The estimate about $\mathcal{D}_{tr}$'s distribution is better with more effective `AttInf`.

Distribution Test (`DistInf`)

• ***Train-Test Factors:*** As the tail length increases, outlier data records (from the tail classes) are memorized more, resulting in higher susceptibility to `DistInf` (**`Tr1-Tail`**: ↑) [15, 16]. `DistInf` exploits the distinguishability of model observables when trained on datasets with different properties. Hence, increasing the distinguishability will also make `DistInf` effective (**`Tr2-Data`**: ↑) [95].

• ***Test-time Factors:*** The effectiveness of `DistInf` is positively correlated with quality of $\mathcal{D}^{aux}$ (**`Te1-Aux`**: ↑), and similarity of $\mathcal{M}_{sur}$'s functionality and architecture with $\mathcal{M}$ (**`Te3-Shdw`**, **`Te4-Arch`**: ↑).

After a successful attack, $\mathcal{A}dv$ learns $\mathcal{D}_{tr}$'s distribution (**`Te2-Dist`**: ↑). The estimate about $\mathcal{D}_{tr}$'s distribution is better with more effective `DistInf`.

Data Reconstruction (`DtRecon`)

• ***Train-Test Factors:*** Data records from tail classes are more susceptible (**`Tr1-Tail`**: ↑) [58]. Higher distinguishability across datasets and subgroups results in better attack effectiveness (**`Tr2-Data`**, **`Tr3-Subgp`**: ↑) [29, 117].

• ***Test-time Factors:*** The effectiveness of `DtRecon` is positively correlated with quality of $\mathcal{D}^{aux}$ (**`Te1-Aux`**: ↑), similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$ (**`Te2-Dist`**: ↑), and similarity of $\mathcal{M}_{sur}$'s functionality and architecture with $\mathcal{M}$ (**`Te3-Shdw`**, **`Te4-Arch`**: ↑).

After a successful attack, $\mathcal{A}dv$ infers data records in $\mathcal{D}_{tr}$, which can act as $\mathcal{D}^{aux}$ for other attacks (**`Te1-Aux`**: ↑). $\mathcal{A}dv$ can more accurately infer data records in $\mathcal{D}_{tr}$ as the effectiveness of `DtRecon` increases.

Unauthorized Model Ownership (`ModExt`)

• ***Train-Test Factors:*** As the tail length increases, $\mathcal{M}$ may not sufficiently capture the tail classes. Hence, training $\mathcal{M}_{sur}$ from $\mathcal{M}$ will exacerbate the utility drop on tail classes for $\mathcal{M}_{sur}$ (**`Tr1-Tail`**: ↓) [112].

• ***Test-time Factors:*** The effectiveness of `ModExt` is positively correlated with quality of $\mathcal{D}^{aux}$ (**`Te1-Aux`**: ↑), and similarity of $\mathcal{D}^{aux}$'s distribution with $\mathcal{D}_{tr}$ (**`Te2-Dist`**: ↑).

$\mathcal{A}dv$ can either learn $\mathcal{M}$'s architecture (**`Te4-Arch`**: ↑) [74, 102], or obtain a functionally similar $\mathcal{M}_{sur}$ (**`Te3-Shdw`**: ↑) [52, 76]. Effective attacks extract architecture and functionality more accurately.

Discriminatory Behavior (`Discr`) In train-test collusion, `Poisn` can be used to increase `Discr`. Here, we treat `Discr` (a risk without attacks) as $\mathbf{Att}_2$ to study such collusion. In test-time collusion, there are no attacks corresponding to `Discr` to obtain outcomes as $\mathbf{Att}_1$. Thus, we omit it.

• ***Train-Test Collusion:*** As the tail length increases (addition of new minority groups), the effectiveness of `Discr` also increases (**`Tr2-Data`**: ↑) [69]. Since `Discr` measures distinguishability across demographic groups, higher distinguishability implies higher `Discr` (**`Tr3-Subgp`**: ↑) [1].

3.3 Guideline to Explain/Predict Collusion

We conjecture that the potential for collusion can be predicted by analyzing the common factors between the two attacks, i.e., comparing "How attack effectiveness leads to change in factors" and under "How changes in factors correlate with attack effectiveness" in Tables 1 and 2. However, defining an exact algorithm is challenging due to the complex and only partially understood dynamics of various factors. Therefore, we present a guideline using expert insight to predict if there is potential for collusion.

We consider two adversaries, $\mathcal{A}dv1$ and $\mathcal{A}dv2$, executing $\texttt{Att}_1$ and $\texttt{Att}_2$, respectively. In train-test collusion, $\texttt{Att}_1$ is always Poisn and $\texttt{Att}_2$ is a test-time attack; in test-time collusion, both are test-time attacks.

G1 Identify the common factor(s) <f_c> by matching the non-empty entries across the two halves of Tables 1 and 2: compare columns where the row corresponds to $\texttt{Att}_1$ (under "How attack effectiveness leads to change in factors") with the same columns where the row corresponds to $\texttt{Att}_2$ (under "How changes in factors correlate with attack effectiveness"). See Tables 1 and 2 for train–test and test-time collusion.

G2 If <f_c> is available, indicate the common factors using a pair of arrows in the format "<f_c>: (<·>$_1$, <·>$_2$)" where <·>$_1$ corresponds to the arrow depicting "How attack effectiveness leads to change in factors" (for $\texttt{Att}_1$) while <·>$_2$ corresponds to the arrow depicting "How changes in factors correlate with attack effectiveness" (for $\texttt{Att}_2$).

- If arrows are aligned ($\uparrow, \uparrow$), it suggests that $\mathcal{A}dv1$ and $\mathcal{A}dv2$ can potentially collude as executing $\texttt{Att}_1$ increases $\texttt{Att}_2$.
- If arrows are not aligned (e.g., $\uparrow, \downarrow$), it suggests $\mathcal{A}dv1$ and $\mathcal{A}dv2$ are unlikely to collude (e.g., exploiting $\texttt{Att}_1$ either decreases $\texttt{Att}_2$ or does not meet prerequisites of $\texttt{Att}_2$).

G3 If <f_c> is not available, collusion cannot be concluded.

We use the guideline to explain collusion in prior work (§4), conjecture about unexplored ones, and empirically validate five cases (§5). We later discuss accounting for $\mathcal{A}dv$'s characteristics (§6), and extending beyond two adversaries (§7).

4 Systematization of Prior Work

We apply our guideline to explain collusion in prior work using factors from our framework (§4.1 and §4.2). We identify prior work through a literature survey and describe our methodology for collecting papers below.

Methodology for Collecting Papers: We surveyed papers covering collusion among adversaries. We collected an initial set of papers using keywords containing both risks. For instance, "poisoning/backdoors increases <$\texttt{Att}_2$>" in case of train-test collusions, and "$\texttt{Att}_1$ increases $\texttt{Att}_2$" for test-time collusions. We then reviewed the citations of these initial papers for comprehensive coverage. Overall, we collected 28 and 12 papers for train-test and test-time collusion respectively.

4.1 Train-Test Collusions

Table 3 shows train-test collusion (indicated as "$\texttt{Att}_1 \rightarrow \texttt{Att}_2$" where $\texttt{Att}_1$ is always Poisn). We indicate the common enabling factors identified using the guideline to explain the collusion indicated as ($\uparrow, \uparrow$) corresponding to (<·>$_1$, <·>$_2$) where <·>$_1$ corresponds to the arrow depicting the "How attack effectiveness leads to change in factors" (for $\texttt{Att}_1$) while <·>$_2$ corresponds to the arrow depicting the "How changes in factors correlate with attack effectiveness" (for $\texttt{Att}_2$).

Table 3: **Train-Test Collusions:** Collusion between $\mathcal{A}dv1$ (exploits Poisn) and $\mathcal{A}dv2$ (exploits test-time $\texttt{Att}_2$). **"Factors"** → common enabling factors (Tables 1 and 2) to identify potential for collusion ($\uparrow, \uparrow$). Gray → no prior work.

$\texttt{Att}_2$	References	Factors (<$\texttt{Att}_1$>, <$\texttt{Att}_2$>)	Collusion?
Evsn	[37, 50, 78]	Tr1-Tail ($\uparrow, \uparrow$)	Yes
ModExt	Our Work (§5.2)	Tr1-Tail ($\uparrow, \downarrow$)	No
MemInf	[19, 20, 101, 109, 123]	Tr1-Tail, Tr2-Data ($\uparrow, \uparrow$)	Yes
AttInf	[26, 66, 101]	Tr1-Tail, Tr3-Subgp ($\uparrow, \uparrow$)	Yes
DistInf	[15, 16, 64, 98, 105]	Tr1-Tail, Tr3-Subgp ($\uparrow, \uparrow$)	Yes
DtRecon	[34, , 77, 81, 91, 101, 126]	Tr1-Tail, Tr2-Data, Tr3-Subgp ($\uparrow, \uparrow$)	Yes
Discr	[39, 67, 68, 72, 90, 104, 129]	Tr1-Tail, Tr3-Subgp ($\uparrow, \uparrow$)	Yes

- **Poisn/Bkdr→Evsn**: Poisn increases the susceptibility to Evsn by increasing $\mathcal{D}_{tr}$'s tail length (**Tr1-Tail**: $\uparrow, \uparrow$) [37, 50, 78].
- **Poisn/Bkdr→MemInf**: Poisn increases susceptibility to MemInf by increasing $\mathcal{D}_{tr}$'s tail length, and distinguishability between data records inside and outside $\mathcal{D}_{tr}$ (**Tr1-Tail**, **Tr2-Data**: $\uparrow, \uparrow$) [19, 20, 101, 109, 123].
- **Poisn→AttInf**: Poisn increases susceptibility to AttInf by increasing $\mathcal{D}_{tr}$'s tail length, and distinguishability across demographic subgroups (**Tr1-Tail**, **Tr3-Subgp**: $\uparrow, \uparrow$) [26, 66, 101].
- **Poisn→DistInf**: Poisn increases susceptibility to DistInf by increasing $\mathcal{D}_{tr}$'s tail length, and distinguishability across datasets with different distributions (**Tr1-Tail**, **Tr2-Data**: $\uparrow, \uparrow$) [15, 16, 64, 98, 105].
- **Poisn/Bkdr→DtRecon**: Poisn/Bkdr make DtRecon more effective by increasing $\mathcal{D}_{tr}$'s tail length, and increasing the distinguishability across datasets and subgroups (**Tr1-Tail**, **Tr2-Data**, **Tr3-Subgp**: $\uparrow, \uparrow$) [34, , 77, 81, 91, 101, 126].
- **Poisn/Bkdr→Discr**: Poisn/Bkdr increases Discr by increasing in $\mathcal{D}_{tr}$'s tail length, and distinguishability across different subgroups (**Tr1-Tail**,**Tr3-Subgp**: $\uparrow, \uparrow$) [39, 67, 68, 72, 90, 104, 129].

Table 4: **Test-time Collusions:** Both adversaries exploit test-time risks: **Att**$_1$ and **Att**$_2$. **"Factors"** → common test-time enabling factors to see potential for collusion (prerequisite of **Att**$_2$ is satisfied by **Att**$_1$: ↑, ↑). Gray → no prior work.

Att$_1$	**Att**$_2$	**Factors** (<Risk1>, <Risk2>)	**Collusion?**
Evsn	**ModExt** [52] **MemInf** [60] **AttInf** **DistInf** [60] **DtRecon**	**Te1-Aux** (↑, ↑)	Yes
ModExt	**Evsn** [52, 79] **MemInf** [111] **AttInf** [17] **DistInf** **DtRecon**	**Te3-Shdw**, **Te4-Arch** (↑, ↑)	Yes
MemInf	**Evsn** **ModExt** [111] **AttInf** [118, 125] **DistInf** **DtRecon** [10, 13]	**Te1-Aux** (↑, ↑)	Yes
AttInf	**Evsn** **ModExt** **MemInf** **DistInf** **DtRecon**	**Te2-Dist** (↑, ↑)	Yes
DistInf	**Evsn** **ModExt** **MemInf** [60] **AttInf** [60] **DtRecon**	**Te2-Dist** (↑, ↑)	Yes
DtRecon	**Evsn** **ModExt** [43] **MemInf** **AttInf** **DistInf**	**Te1-Aux** (↑, ↑)	Yes

4.2 Test-Time Collusions

Table 4 shows test-time collusion (indicated as "**Att**$_1$ → **Att**$_2$"). We denote the common factors as ($<\cdot>_1$, $<\cdot>_2$) where $<\cdot>_1$ corresponds to the arrow depicting the "How attack effectiveness leads to change in factors" (for **Att**$_1$) while $<\cdot>_2$ corresponds to the arrow depicting the "How change in factors correlate with attack effectiveness" (for **Att**$_2$).

- **Evsn** → {**ModExt**, **MemInf**, **DistInf**}: ModExt, MemInf, and DistInf require $\mathcal{D}^{aux}$ to train $\mathcal{M}_{shdw}$ (**Te1-Aux**: ↑). Evsn can provide $\mathcal{D}^{aux}$ with adversarial examples to meet the prerequisites of ModExt, MemInf, and DistInf (**Te1-Aux**: ↑) [52, 60].
- **MemInf** → {**AttInf**, **DtRecon**, **ModExt**}: AttInf, DtRecon, and **ModExt** can be more effective when $\mathcal{D}^{aux}$ has some overlap with $\mathcal{D}_{tr}$ to increase the functional similarity of $\mathcal{M}_{shdw}$ with $\mathcal{M}$ (**Te1-Aux**: ↑). $\mathcal{A}dv1$ can exploit MemInf to identify data records in $\mathcal{D}_{tr}$ which can be added to $\mathcal{D}^{aux}$ to improve the functional similarity of $\mathcal{M}_{shdw}$ to $\mathcal{M}$. This can help design more effective (and transferrable) attacks: AttInf, DtRecon, and **ModExt** (**Te1-Aux**: ↑) [10, 13, 118].
- **DistInf** → {**AttInf**, **MemInf**}: AttInf and MemInf are more effective if $\mathcal{D}^{aux}$ has a similar distribution as $\mathcal{D}_{tr}$ (**Te2-Dist**: ↑). By exploiting DistInf, $\mathcal{A}dv1$ can infer $\mathcal{D}_{tr}$'s distribution, and $\mathcal{A}dv2$ use that knowledge to update $\mathcal{D}^{aux}$ to train $\mathcal{M}_{shdw}$ which is functionally more similar to $\mathcal{M}$. This can improve the effectiveness of AttInf and MemInf (**Te2-Dist**: ↑) [60].
- **DtRecon** → **ModExt**: Training $\mathcal{M}_{shdw}$ on $\mathcal{D}^{aux}$ which partially overlaps with $\mathcal{D}_{tr}$, increases ModExt's effectiveness (**Te1-Aux**: ↑). DtRecon can identify data records in $\mathcal{D}_{tr}$ which can be added to $\mathcal{D}^{aux}$. This increases the overlap with $\mathcal{D}_{tr}$ improving the functional similarity of $\mathcal{M}_{shdw}$ with $\mathcal{M}$, enabling a more effective ModExt (**Te1-Aux**: ↑) [43].
- **ModExt** → {**Evsn**, **MemInf**, **AttInf**}: Transferable adversarial examples (in a blackbox setting), MemInf, AttInf, and DtRecon require $\mathcal{M}_{shdw}$ which is functionally similar to $\mathcal{M}$ (**Te3-Shdw**, **Te4-Arch**: ↑). ModExt can help extract such $\mathcal{M}_{shdw}$ from $\mathcal{M}$ to meet the prerequisites of other attacks (**Te3-Shdw**, **Te4-Arch**: ↑) [17, 52, 79, 111].

5 Unexplored Collusions among Adversaries

We apply our guideline to conjecture about the potential of unexplored collusions (§5.1 and marked as gray in Tables 3 and 4), and empirically validate one train-test and four test-time collusions (§5.2, §5.3, and §5.4)[6].

5.1 Conjectures for Unexplored Collusions

Train-Test Collusions: There is only one unexplored train-test collusion (gray in Table 3) between $\mathcal{A}dv1$ who executes **Poisn**, and $\mathcal{A}dv2$ who executes **ModExt**. We see that **Tr1-Tail** is the common factor between both attacks. Since the arrows are opposing (↑ for Poisn and ↓ for ModExt), our guideline suggests that there is no potential for collusion (Poisn decreases the effectiveness of ModExt). We validate this conjecture in §5.2.

Test-time Collusions: We discuss various unexplored test-time collusions (gray in Table 4), and use our guideline to identify the potential for collusion, indicated as (↑, ↑).

- **Evsn** → {**AttInf**, **DtRecon**}: **Te1-Aux** is the common factor which suggests that Evsn generates adversarial examples which can be used as $\mathcal{D}^{aux}$ to train $\mathcal{M}_{shdw}$. Since the adversarial examples can characterize $\mathcal{M}$'s decision boundary, it can allow $\mathcal{M}_{shdw}$ to be functionally similar to $\mathcal{M}$. The resulting $\mathcal{M}_{shdw}$ can be used as a "shadow model" for other attacks such as AttInf, DtRecon, and Discr.
- **MemInf** → {**Evsn**, **DistInf**}: MemInf can help identify data records in $\mathcal{D}_{tr}$, which can then be included as part of $\mathcal{D}^{aux}$ (**Te1-Aux**). The resulting $\mathcal{M}_{shdw}$ trained on $\mathcal{D}^{aux}$ which overlaps with $\mathcal{D}_{tr}$, is likely to increase the functional similarity with $\mathcal{M}$, thereby improving the effectiveness of

[6]Code: https://zenodo.org/records/20429643

Evsn, ModExt, and DistInf.For instance, prior work on Evsn and DistInf have shown that increasing overlap between $\mathcal{D}^{aux}$ and $\mathcal{D}_{tr}$ can increase the effectiveness of the attacks [25, 95, 122].

- **AttInf** $\rightarrow$ {**Evsn**, **MemInf**, **DistInf**, **DtRecon**, **ModExt**}: AttInf can help estimate distribution of $\mathcal{D}_{tr}$ allowing $\mathcal{A}dv$ to subsequently update $\mathcal{D}^{aux}$ (**Te2-Dist**). This can improve the effectiveness of Evsn, ModExt, MemInf, DtRecon, and DistInf, by training $\mathcal{M}_{shdw}$ which is functionally more similar to $\mathcal{M}$.
- **DistInf** $\rightarrow$ {**Evsn**, **DtRecon**, **ModExt**}: DistInf can infer $\mathcal{D}_{tr}$'s distribution allowing $\mathcal{A}dv$ to subsequently update $\mathcal{D}^{aux}$ (**Te2-Dist**). The resulting $\mathcal{M}_{shdw}$ is likely to be more similar to $\mathcal{M}$, thus, likely improving the effectiveness of Evsn, ModExt, DtRecon, and DistInf.
- **DtRecon** $\rightarrow$ {**Evsn**, **MemInf**, **AttInf**, **DistInf**}: **Te1-Aux** is the common factor among the attacks. DtRecon can help partially identify data records in $\mathcal{D}_{tr}$ which can augment $\mathcal{D}^{aux}$ (**Te1-Aux**). This can result in training $\mathcal{M}_{shdw}$ to be similar to $\mathcal{M}$, thus, improving the effectiveness of Evsn, MemInf, AttInf, and DistInf.
- **ModExt** $\rightarrow$ {**DistInf**, **DtRecon**}: ModExt can help train $\mathcal{M}_{shdw}$ that is similar to $\mathcal{M}$ (**Te3-Shdw**, **Te4-Arch**), which can be used as "shadow models" for effective DistInf and DtRecon.

As an illustrative example, we validate the conjecture for four test-time collusions: ModExt $\rightarrow$ DistInf (§5.3) and DtRecon $\rightarrow$ {MemInf, AttInf, DistInf } (§5.4).

Takeaway: Unexplored collusions are gaps in the current literature, which are left as directions for future work.

5.2 Train-Test Collusion: Poisn $\rightarrow$ ModExt

We now validate our conjecture that there is no potential for collusion between Poisn and ModExt, marked as ($\uparrow$, $\downarrow$) in Table 3. Since Poisn increases the tail length of $\mathcal{D}_{tr}$ (**Tr1-Tail**), we conjecture that $\mathcal{M}_{sur}$ may fail to learn the under-represented data records, thereby reducing the effectiveness of ModExt.

Experimental Setup: We run our experiments on two image datasets: CIFAR10/100. The dataset contains 60,000 images (50,000 for training and 10,000 for testing) where each data point is a 32 $\times$ 32 colored image. The images are clustered into 10 and 100 classes for CIFAR10 and CIFAR100 respectively, representing different objects. We use 25,000 records for $\mathcal{D}_{tr}$ and the remaining 25,000 records as $\mathcal{D}^{aux}$ for training $\mathcal{M}_{sur}$ via ModExt. Hence, the accuracy of $\mathcal{M}$ is not the same as the state-of-the-art on these datasets. We use ResNet models as $\mathcal{M}$ and VGG16 as $\mathcal{M}_{sur}$. We check that $\mathcal{M}$ has been successfully poisoned using *accuracy on poisoned test dataset*[7]. We report $\mathcal{M}_{sur}$'s *accuracy* and *fidelity* (the percentage of outputs that match between $\mathcal{M}$ and $\mathcal{M}_{sur}$) on a clean test dataset. We use backdoors with a 5 x 5 white patch as the trigger, and knockoff-nets for ModExt by training $\mathcal{M}_{sur}$ using $\mathcal{M}$'s predictions as the ground truth [76]. We consider different query budgets for training $\mathcal{M}_{sur}$ including {10%, 25%, 50%, 100%} of the 25,000 data records in $\mathcal{D}^{aux}$.

[7]We empirically verify this, but do not report the numbers for brevity.

Table 5: **Poisn** $\rightarrow$ **ModExt**: Surrogate model's accuracy ("Surr. Acc.") and fidelity ("Surr. Fid.") for 0% poison are the baseline. For a given budget, green $\rightarrow$ higher than baseline, orange $\rightarrow$ similar to baseline (within std. dev.), and red $\rightarrow$ lower than baseline. We report target accuracy to show that ModExt is effective for different poisoning rates.

Budget	CIFAR10 Surr. Acc.	CIFAR10 Surr. Fid.	CIFAR100 Surr. Acc.	CIFAR100 Surr. Fid.
0% Poison (Baseline)				
	Target Acc.: $87.37_{\pm 0.16}$		**Target Acc.**: $62.24_{\pm 0.47}$	
2500	$76.83_{\pm 0.71}$	$78.54_{\pm 0.54}$	$46.99_{\pm 1.59}$	$54.75_{\pm 1.16}$
6250	$83.41_{\pm 0.23}$	$87.19_{\pm 1.09}$	$53.04_{\pm 1.07}$	$62.63_{\pm 0.92}$
12500	$85.05_{\pm 0.27}$	$88.82_{\pm 0.63}$	$54.67_{\pm 0.84}$	$65.40_{\pm 0.70}$
25000	$85.91_{\pm 0.43}$	$89.58_{\pm 1.07}$	$56.09_{\pm 0.88}$	$66.55_{\pm 0.90}$
5% Poison				
	Target Acc.: $78.19_{\pm 0.60}$		**Target Acc.**: $33.69_{\pm 1.36}$	
2500	$75.30_{\pm 1.12}$	$79.84_{\pm 1.71}$	$33.25_{\pm 0.29}$	$37.24_{\pm 0.60}$
6250	$78.04_{\pm 0.42}$	$83.31_{\pm 0.85}$	$35.00_{\pm 0.13}$	$39.90_{\pm 0.59}$
12500	$78.30_{\pm 0.25}$	$83.31_{\pm 0.31}$	$35.34_{\pm 1.21}$	$40.81_{\pm 1.22}$
25000	$78.93_{\pm 0.26}$	$84.39_{\pm 0.24}$	$34.80_{\pm 0.55}$	$40.14_{\pm 1.41}$
10% Poison				
	Target Acc.: $77.05_{\pm 0.49}$		**Target Acc.**: $33.57_{\pm 1.66}$	
2500	$74.60_{\pm 0.60}$	$78.96_{\pm 0.58}$	$32.54_{\pm 1.34}$	$36.96_{\pm 0.99}$
6250	$75.86_{\pm 0.23}$	$80.59_{\pm 0.85}$	$35.56_{\pm 1.85}$	$40.49_{\pm 1.43}$
12500	$78.29_{\pm 0.51}$	$83.95_{\pm 0.69}$	$35.00_{\pm 3.05}$	$40.11_{\pm 3.20}$
25000	$78.00_{\pm 0.30}$	$83.87_{\pm 0.53}$	$34.76_{\pm 1.34}$	$40.47_{\pm 0.87}$
15% Poison				
	Target Acc.: $75.90_{\pm 0.56}$		**Target Acc.**: $33.51_{\pm 1.95}$	
2500	$75.29_{\pm 0.61}$	$79.16_{\pm 0.22}$	$33.25_{\pm 1.23}$	$38.20_{\pm 0.87}$
6250	$75.99_{\pm 1.56}$	$81.17_{\pm 1.72}$	$34.23_{\pm 1.71}$	$39.72_{\pm 1.38}$
12500	$76.33_{\pm 1.55}$	$81.63_{\pm 0.73}$	$34.89_{\pm 1.11}$	$41.32_{\pm 0.78}$
25000	$77.11_{\pm 0.86}$	$82.54_{\pm 1.43}$	$34.99_{\pm 1.63}$	$40.98_{\pm 0.96}$
20% Poison				
	Target Acc.: $74.70_{\pm 0.40}$		**Target Acc.**: $30.38_{\pm 0.39}$	
2500	$73.03_{\pm 1.32}$	$76.86_{\pm 1.78}$	$29.78_{\pm 0.57}$	$35.25_{\pm 1.25}$
6250	$75.22_{\pm 0.70}$	$80.20_{\pm 1.74}$	$31.65_{\pm 0.97}$	$37.83_{\pm 1.27}$
12500	$75.41_{\pm 0.70}$	$81.44_{\pm 0.71}$	$31.67_{\pm 1.69}$	$38.34_{\pm 1.34}$
25000	$75.65_{\pm 0.95}$	$82.27_{\pm 0.44}$	$31.69_{\pm 1.83}$	$37.51_{\pm 1.17}$

Results: Table 5 shows the results of extracting the target model $\mathcal{M}$ using ModExt on CIFAR10/100. For a given budget, we compare the surrogate accuracy ("Surr. Acc.") and fidelity ("Surr. Fid.") when $\mathcal{M}$ is trained with poisons {0.05%, 0.1%, 0.15%, 0.2%}, with the case where $\mathcal{M}$ is trained without poisons (0% poison). We select a low poison rate to evaluate ModExt's effect, without significant utility degradation from poisoning. For a specific budget, to compare the baseline accuracy and fidelity (i.e., 0% poison), we use green for higher than baseline, orange for similar (within standard deviation), and red for lower than baseline.

Across different poison rates and budgets, we see that the surrogate accuracy and fidelity are lower than the baseline for CIFAR100 (red). For CIFAR10, the surrogate accuracy and fidelity are similar to the baseline (orange) for the low budget of 2500 queries, where $\mathcal{M}_{sur}$ generalizes well. However, for higher budgets, the accuracy / fidelity are lower than the baseline (red). Overall, these results suggest that `Poisn` reduces the effectiveness of `ModExt`, confirming our conjecture that there is no increase in potential for collusion.

5.3 Test-time Collusion: `ModExt` →`DistInf`

We validate our guideline's conjecture on the potential for collusion between `ModExt` and `DistInf`, marked as (↑, ↑) in Table 4. Here, the common factors are **`Te3-Shdw`** and **`Te4-Arch`** suggesting that `ModExt` increases the knowledge about $\mathcal{M}$ for better shadow models for `DistInf`. We validate by comparing `DistInf` attack accuracy when its shadow models are trained independently on $\mathcal{D}^{aux}$ versus extracted from $\mathcal{M}$ using `ModExt`.

Experimental Setup: We assume both $\mathcal{A}dv1$ and $\mathcal{A}dv2$ have blackbox access to $\mathcal{M}$. $\mathcal{A}dv1$ performs `ModExt` where they steal the functionality of $\mathcal{M}$ by sending queries to $\mathcal{M}$ and using the corresponding outputs to train the surrogate models.

For `ModExt`, we use KnockoffNets [76]. For `DistInf`, we use the blackbox attack from Suri et al. [94] where we train an attack classifier to distinguish models trained on datasets with ratios α_1 and α_2. For example, the classifier predicts if the proportion of males in the training data is $\alpha_1 = 0.1$ or $\alpha_2 = 0.9$. Following Suri et al. [94], we consider two examples ratios where α_1 is close to α_2 ($\alpha_1 = 0.45$ vs. $\alpha_2 = 0.55$; and $\alpha_1 = 0.475$ vs. $\alpha_2 = 0.525$).

We use two image datasets with sensitive attributes as required for `DistInf`. CELEBA contains 202,599 celebrity face images with 40 binary attributes, and UTKFACE contains 23,705 in-the-wild face images with age, sex, and race attributes. We split each dataset 50/50 into a train and a test partition, then split each partition 50/50 (stratified jointly on the label and the sensitive attribute: sex for CELEBA, race for UTKFACE) into target and $\mathcal{A}dv$ share. We report the attack accuracy of `DistInf` as the mean and standard deviation across five runs. We confirm that fidelity and accuracy of the stolen shadow models are high and omit them for brevity.

Results: We report three settings in Table 6: *baseline* where the shadow models for `DistInf` are trained independently without querying $\mathcal{M}$; *Cross-Arch* where the shadow models have a different architecture than $\mathcal{M}$ (i.e., VGG11) and obtained using `ModExt`; and *Same-Arch* where the shadow model has the same architecture as $\mathcal{M}$, obtained using `ModExt`.

As expected, we find that the attack accuracy of `DistInf` is significantly higher for cross-arch and same-arch settings (where shadow models are obtained using `ModExt`) compared to the case where the shadow models are trained independently on $\mathcal{D}^{aux}$ (in green). This confirms our conjecture that `ModExt` improves `DistInf`.

Table 6: **`ModExt` → `DistInf`:** *Baseline* → shadow models trained independently without querying $\mathcal{M}$ (target); *Cross-Arch* → VGG11 shadow model extracted from $\mathcal{M}$; *Same-Arch* → ResNet34 shadow model extracted from $\mathcal{M}$. Green → higher than baseline, orange → similar to baseline (within std. dev.), red → lower than baseline.

Setting	CELEBA		UTKFACE	
	$\alpha_1 = 0.45$ $\alpha_2 = 0.55$	$\alpha_1 = 0.475$ $\alpha_2 = 0.525$	$\alpha_1 = 0.45$ $\alpha_2 = 0.55$	$\alpha_1 = 0.475$ $\alpha_2 = 0.525$
Baseline.	$70.00_{\pm 14.58}$	$55.00_{\pm 18.11}$	$55.00_{\pm 3.06}$	$45.50_{\pm 10.81}$
Cross-Arch	$98.00_{\pm 3.26}$	$96.50_{\pm 4.18}$	$91.50_{\pm 4.54}$	$88.00_{\pm 7.79}$
Same-Arch	$99.00_{\pm 1.37}$	$98.50_{\pm 3.35}$	$93.00_{\pm 7.37}$	$92.50_{\pm 4.68}$

5.4 Test-time Collusion: `DtRecon` → {`MemInf`, `AttInf`, `DistInf`}

We validate our conjecture on collusion between `DtRecon` and three privacy attacks: `MemInf`, `AttInf`, and `DistInf`, marked as (↑, ↑) in Table 4. The common factor is **`Te1-Aux`** suggesting that `DtRecon` improves the quality of $\mathcal{D}^{aux}$ to be similar to $\mathcal{D}_{tr}$. This helps train better shadow models for `MemInf`, `AttInf`, and `DistInf`. We validate the conjecture by comparing the attack accuracy of `MemInf`, `AttInf`, and `DistInf` in two settings: (i) when $\mathcal{D}^{aux}$ has no overlap with $\mathcal{D}_{tr}$, and (ii) when $\mathcal{D}^{aux}$ includes records extracted using `DtRecon`, and used to train the shadow models. We evaluate using the mean and standard deviation over five runs unless stated otherwise.

Experimental Setup: To validate the conjecture, we generate reconstructed data by executing `DtRecon` on a fraction of $\mathcal{D}_{tr}$. We then replace the fraction of $\mathcal{D}^{aux}$ with these reconstructed $\mathcal{D}_{tr}$ records (aka replacement ratio, we use {25%, 50%, 75%}). For `DtRecon`, we use generative gradient inversion [31]. We assume $\mathcal{A}dv1$ has whitebox access to $\mathcal{M}$ (e.g., in a federated learning setting where $\mathcal{A}dv$ can access the model gradients), and uses this to reconstruct data records in $\mathcal{D}_{tr}$. These reconstructed data records are then shared with $\mathcal{A}dv2$ to train shadow models for `MemInf`, `AttInf`, and `DistInf`.

For `MemInf`, we use the likelihood ratio attack [11] on CIFAR10/100 (§5.2). Each dataset's 50k training images are split equally into a target pool (members) and $\mathcal{A}dv$'s non-member pool ($\mathcal{D}^{aux}$). $\mathcal{M}$ is trained on a random 12.5k subset of the target pool, while $\mathcal{A}dv$ trains 64 shadow models (50 epochs each) on random halves of a 15k-record $\mathcal{D}^{aux}$. $\mathcal{D}_{te}$ is the standard 10k records from the datasets. We evaluate on a fixed disjoint set of 2k records (1k members, 1k non-members). Overall, we have $|\mathcal{D}_{tr}| = 12{,}500, |\mathcal{D}_{te}| = 10{,}000, |\mathcal{D}^{aux}| = 25{,}000$, and a membership evaluation set of size 2,000. All models are ResNet34.

For `AttInf`, we use the attack from Aalmoes et al. [1] on

CELEBA and UTKFACE (§5.4). For CELEBA, the target task is smile prediction and the sensitive attribute is sex; for UTKFACE, the target task is predicting sex and the sensitive attribute is race. Stratified on the label and the sensitive attribute, the target trains on $\mathcal{D}_{tr}$ (1,000 members, of which $\mathcal{A}dv$ reconstructs 500 and the other 500 are held-out), $\mathcal{A}dv$ holds $\mathcal{D}^{aux}$ (1,500 non-members), and evaluation queries the 500 held-out members and 500 held-out non-members. Overall, we have $|\mathcal{D}_{tr}| = 1{,}000, |\mathcal{D}^{aux}| = 1{,}500$, and an evaluation set of size 1,000. Both $\mathcal{M}$ and attack model use ResNet18.

Finally, for `DistInf`, we use the blackbox attack of Suri et al. [94] on CELEBA and UTKFACE (§5.3), with sex as the sensitive attribute. $\mathcal{M}$ is trained at ratio α_1, and the attack asks whether $\mathcal{M}$'s ratio is α_1 or an alternative ratio α_2. We train 128 shadow models (64 with subgroup ratio α_1, 64 with α_2), each on 200 records for 12 epochs. Through each shadow model, we pass a fixed set of 500 query images (half male, half female), and record its loss on each which are used by a logistic-regression attack classifier to predict whether $\mathcal{M}$ was trained with α_1 or α_2. We report average accuracy over three runs. $\mathcal{M}$ and shadow models are ResNet18.

Results (`DtRecon` → `MemInf`): We report the TPR@0.01 FPR for different percentages of replaced records. For the first dataset, the baseline TPR of 1.50 increases to 2.50 at 25% replacement and further to 3.50 at 50%, before dropping to 2.10 at 75% replacement. Similarly, for the second dataset, the baseline TPR of 2.50 improves to 2.70 at 25% replacement and 2.80 at 50%, but decreases to 2.60 at 75% replacement. Overall, the TPR@0.01 FPR improves with moderate levels of replacement, while a higher replacement ratio of 75% leads to a decline in TPR for both datasets. We conjecture that the benefit of adding member-like records to the shadow training set saturates after a few hundred records, while each inexact reconstruction acts as a slightly mislabeled member whose error accumulates as more are replaced. Thus, for moderate ratios, we find that `DtRecon` can improve `MemInf`, confirming our conjecture for the collusion potential.

Results (`DtRecon` → `AttInf`): We report the AUC score under ROC curve. For CELEBA, the baseline AUC at 0% replacement is 69.25% (average value), which increases to 71.45% at 25% replacement, 73.07% at 50%, and reaches 73.95% at 75% replacement. Similarly, for UTKFACE, the baseline AUC of 62.94% improves consistently with higher replacement ratios, increasing to 64.85% at 25%, 66.03% at 50%, and 67.42% at 75% replacement. We find that the improvement for 50% and 75% replacement gives significant performance improvement when compared to the baseline. Overall, for both datasets, increasing the percentage of replaced records leads to an improvement in AUC score. This confirms our conjecture that there is potential for collusion where `DtRecon` can help improve `AttInf`.

Results (`DtRecon` → `DistInf`): We report the attack accuracy results for `DistInf` under two ratio configurations: α_1=0.5 vs. α_2=0.1, and α_1=0.5 vs. α_2=0.9 (following results in Suri and Evans [95]). For UTKFACE with α_2=0.1, the baseline attack accuracy at 0% replacement is 62.66%, which increases to 74.06% at 25%, $\sim$ 76% at 50% and 75% replacement. For α_2=0.9, the attack accuracy improves from a baseline of 60.31% to 72.50% at 25%, 72.81% at 50%, and further to 75.31% at 75% replacement.

For CELEBA with α_2=0.1, the baseline attack accuracy of 60.47% increases consistently to 64.53% at 25% replacement, 68.91 at 50%, and 71.09% at 75%. The strongest trend is observed for CELEBA with α_2=0.9, where the baseline accuracy of 63.44% increases substantially to 80.78% at 25% replacement, 82.81% at 50%, and reaches 87.66% at 75% replacement. Overall, across both datasets and ratio configurations, increasing the percentage of reconstructed records in $\mathcal{D}^{aux}$ consistently improves `DistInf`, supporting the conjecture that `DtRecon` can increase collusion potential.

6 Accounting for Adversary's Characteristics

So far, we did not explicitly take $\mathcal{A}dv$'s characteristics into account but implicitly assumed that each $\mathcal{A}dv$ possesses the minimal set of characteristics needed to execute the attack. We now extend our framework to consider $\mathcal{A}dv$'s characteristics by discussing the impact of changing them on the collusion potential. We present Tables 8 and 9 showing the impact of collusion potential on varying $\mathcal{A}dv$'s characteristics. The notations for $\mathcal{A}dv$'s characteristics are based on our proposed streamlined threat model in Appendix A. The notations are summarized in Table 7. We consider the following two cases depending on which $\mathcal{A}dv$ is changed:

- $\mathcal{A}dv$**1** → **Change;** $\mathcal{A}dv$**2** → **Same**: Table 8 shows the minimum characteristics for each risk that we implicitly assumed so far (indicated as ★). Increasing $\mathcal{A}dv$1's characteristics will increase the effectiveness of $\mathbf{Att}_1$ and the correlations with underlying factors (Tables 1 and 2). This, in turn, increases the potential for collusion and facilitates $\mathbf{Att}_2$. From Table 8, we can determine whether the potential for collusion increases based on increasing $\mathcal{A}dv$1's characteristics beyond the minimum requirements (indicated as ↑) when going from left (weak) to right (strong).
- $\mathcal{A}dv$**1** → **Same;** $\mathcal{A}dv$**2** → **Change**: Recall that the guideline identifies a potential for collusion when the *factors* required for $\mathbf{Att}_2$ are satisfied by the *outcomes* of $\mathbf{Att}_1$. Strengthening $\mathcal{A}dv$2's characteristics can increase the effectiveness of $\mathbf{Att}_2$, though not necessarily by increasing collusion potential. In some cases, stronger characteristics may directly satisfy prerequisites required for $\mathbf{Att}_2$, independent of $\mathbf{Att}_1$'s outcomes, thereby reducing or eliminating the need for collusion. Therefore, collusion should be evaluated only over the factors for $\mathbf{Att}_2$ that remain unmet after accounting for $\mathcal{A}dv$2's characteristics. Table 9 summarizes such cases, where ↓ indicates reduced collusion potential under a specific characteristic, and ✗ indicates that collusion with $\mathbf{Att}_1$ becomes unnecessary. We illustrate how

Table 7: $\mathcal{A}dv$**'s Characteristics and their descriptions**. Detailed descriptions provided in Appendix A.

Category	Notation	Description
Role ($\overline{\texttt{Role}}$)	DtOwnr	Data Owner
	DtProv	Data Provider
	CodeProv	Code Provider
	ModProv	Model Provider
	ModTrnr	Model Trainer
	ModOwnr	Model Owner
	SrvProv	Service Provider
	Clnt	Client
Objective ($\overline{\texttt{Obj}}$)	Evsn	Evasion
	Poisn	Poisoning
	Bkdr	Backdoor
	MemInf	Membership Inference
	AttInf	Attribute Inference
	DistInf	Distribution Inference
	DtRecon	Data Reconstruction
	ModExt	Unauth. Model Ownership
	Discr	Discriminatory Behavior
Capability ($\overline{\texttt{Cap}}$)	HbC	Honest-but-Curious $\mathcal{A}dv$
	Mal	Malicious $\mathcal{A}dv$
Optimization ($\overline{\texttt{Opt}}$)	Adpt	Adaptive
	nonAdpt	Non-Adaptive
Model Knowledge ($\overline{\texttt{K}_\texttt{M}}$)	NoAccess	No Access
	BBox	Blackbox
	GBox	Graybox
	WBox	Whitebox
	hardLbl	Hard labels
	topK	Top-K Predictions
	fullPred	Full prediction vector
Interaction ($\overline{\texttt{I}}$)	NoAccess	No Access
	1Shot	One-Shot (Single Query)
	KShot	K-Shot (k Queries)
	Unlmtd	Unlimited Queries
Data Knowledge ($\overline{\texttt{K}_\texttt{D}}$)	NoAccess	No Access
	NoOvlap	No-overlap with $\mathcal{D}_{tr}$
	PartOvlap	Partial overlap
	FullOvlap	Full overlap
	NoBlind	Non-Blind
	Blind	Blind

strengthening $\mathcal{A}dv2$'s characteristics removes or reduces the need for collusion, using Table 9.

- **Model knowledge:** Granting $\mathcal{A}dv2$ whitebox access to $\mathcal{M}$ (WBox) satisfies both the functionality and architecture requirements (**Te3-Shdw**, **Te4-Arch**), which ModExt would otherwise provide. As a result, attacks such as MemInf, AttInf, DistInf, and DtRecon can be carried out effectively *without* colluding with $\mathcal{A}dv1$ for ModExt (✗ in Table 9). Granting only graybox access (GBox) satisfies the architecture requirement (**Te4-Arch**) but not functionality; this *reduces* but does not remove the need for collusion with $\mathcal{A}dv1$ for ModExt (↓).
- **Data knowledge ($\mathcal{D}^{aux}$):** MemInf, Evsn, and DtRecon produce $\mathcal{D}^{aux}$ as an outcome (**Te1-Aux**), which can strengthen other attacks. However, **Te1-Aux** may be satisfied directly by increasing $\mathcal{A}dv2$'s knowledge about $\mathcal{D}_{tr}$: full overlap (FullOvlap) removes the need to collude with $\mathcal{A}dv1$ for MemInf, Evsn, and DtRecon (✗), while partial overlap (PartOvlap) reduces it (↓).
- **Data knowledge (distribution):** AttInf and DistInf provide $\mathcal{D}_{tr}$'s distribution as an outcome (**Te2-Dist**), which strengthens attacks that require **Te2-Dist**, such as AttInf, Evsn, MemInf, DtRecon, and ModExt. **Te2-Dist** can instead be satisfied by increasing $\mathcal{A}dv2$'s knowledge about $\mathcal{D}_{tr}$: FullOvlap removes the need for collusion while PartOvlap reduces it.

Based on the above discussion, we need to refine the guideline to account for the cases where strengthening $\mathcal{A}dv2$'s characteristics satisfies $\texttt{Att}_2$'s prerequisites without requiring collusion. Specifically, after identifying common factors between $\texttt{Att}_1$ and $\texttt{Att}_2$ from Tables 1 and 2, we use Tables 8 and 9 to determine whether $\mathcal{A}dv2$'s characteristics already meet the prerequisites for $\texttt{Att}_2$. Hence, when assessing collusion potential, we refine our guideline (**G1**) to consider only the common *unmet* factors between risks *after* accounting for the adversary's characteristics.

Takeaway: We revise our guideline to account for the setting where $\mathcal{A}dv2$ has characteristics which meet the prerequisite, removing the need for collusion.

7 Discussion and Summary

Framework and Requirements: We evaluate our framework against the two stated requirements: comprehensiveness and extensibility.

R1 Comprehensive: Our framework is comprehensive with respect to both the risks considered for collusion and the factors used to relate them.

- *Comprehensiveness of risks:* Our framework covers security, privacy, and fairness risks that are sufficiently studied in prior work to support systematic reasoning about collusion across the ML pipeline. These include both train-time and test-time risks for which interactions can be characterized through shared underlying factors.
  Risks that lack sufficient prior analysis to enable such characterization are outside the scope of this work. For example:
  - *Side-channel attacks [7, 48, 113]* operate under a blackbox $\mathcal{M}$ (BBox) where a Clnt or SrvProv exploits physical or system-level leakage to extract model architecture, and partial model parameters.
  - *Fault injection [65, 83, 115]* occurs when a Clnt or SrvProv exploits hardware vulnerabilities to induce targeted faults during execution or weight storage, enabling accuracy degradation or targeted misclassification.
  - *Adversarial initialization [44, 45]* is a train-time attack where a ModProv or ModTrnr initialize the model such that standard training on benign data converges to $\mathcal{A}dv$-

Table 8: **Impact of $\mathcal{A}dv$'s Characteristics on Collusion Potential ($\mathcal{A}dv$1 → Change; $\mathcal{A}dv$2 → Same):** Rows correspond to $\texttt{Att}_1$; we vary $\mathcal{A}dv$1's characteristics while $\mathcal{A}dv$2 is fixed; values range from weak (left) to strong (right). ★ → minimum requirement for $\mathcal{A}dv$1 to execute $\texttt{Att}_1$; ↑ → collusion potential with $\texttt{Att}_2$ increases by strengthening the characteristic from ★; empty cell → insufficient, unrealistic, or unnecessary. Refer to Appendix A for description of notations. The table markings are based on prior work covering each of the attacks [23, 32, 75, 80, 84, 96].

$\texttt{Att}_1$	Role	Cap		Opt		$\overline{\texttt{K}_\texttt{M}}$							$\overline{\texttt{I}}$				$\overline{\texttt{K}_\texttt{D}}$					
		HbC	Mal	nonAdpt	Adpt	NoAccess	BBox	GBox	WBox	hardLbl	topK	fullPred	NoAccess	1Shot	KShot	Unlmtd	NoAccess	NoOvlap	PartOvlap	FullOvlap	Blind	NoBlind
Poisn	DtOwnr/DtProv		★	★		★							★						★	↑	★	↑
	ModTrnr		★	★	↑				★			★				★				★		★
	ModProv		★	★				★									★					
	CodeProv		★	★		★							★				★					
Evsn	Clnt		★	★	↑		★	↑	↑	★	↑	↑		★	↑	↑		★	↑	↑	★	↑
ModExt	Clnt	★		★	↑		★			★	↑	↑			★			★	↑	↑	★	↑
MemInf	Clnt	★		★	↑		★	↑	↑	★	↑	↑		★	↑	↑		★	↑	↑	★	↑
AttInf	Clnt	★		★	↑		★	↑	↑	★	↑	↑		★	↑	↑		★	↑	↑	★	↑
DistInf	Clnt	★		★	↑		★	↑	↑	★	↑	↑		★	↑	↑		★	↑	↑		★
DtRecon	Clnt	★		★	↑		★	↑	↑	★	↑	↑		★	↑	↑		★	↑	↑	★	↑

desired behavior (e.g., backdoors, increased training time, or biased outputs).

– *Data ordering attacks [51, 88]* is a specific form of Poisn in which a DtOwnr or DtProv poisons the model by reordering data during training.
– *Sponge examples [24, 89, 106]* violates availability where Clnt sends carefully crafted inputs that increase energy consumption and latency.

As research on these risks matures, they can be incorporated into our framework by reasoning about $\mathcal{A}dv$'s characteristics and any additional factors required for the risk.

- *Comprehensiveness of factors:* Our framework identifies a set of train-test and test-time factors that underlie collusion among risks. These factors are drawn from prior work and are common across the risks considered in this paper (§2). We show how *each risk* relates to *all* identified factors, thus enabling systematic reasoning about collusion (Tables 1 and 2). While we do not claim that the identified factors are complete, our framework allows adding more factors.

Overall, our framework is comprehensive within its defined scope of risks in §2 (meets **R1**) and can be extended to incorporate new risks and factors.

R2 Extensible: Our framework is designed to support additional factors, risks, and adversary characteristics. To add new factors, we can specify how each factor relates to existing risks, as illustrated in Tables 1 and 2. For new risks, we characterize how the risk relates to the existing factors (e.g., Tables 1 and 2 and §3.2). Once these relations are defined, the guideline can be used to study the potential collusion. Overall, our framework satisfies the extensibility requirement (**R2**).

Takeaway: Our proposed framework is comprehensive (**R1**) and extensible (**R2**).

Extending Beyond Two Adversaries: Our framework extends naturally to collusions with more than two adversaries. We first discuss two cases where $\mathcal{A}dv$1 colludes with multiple adversaries in *parallel* (we use "⇒" for "colludes with"):

- **Train $\mathcal{A}dv$1 ⇒ Multiple Test $\mathcal{A}dv$s**
  $\mathcal{A}dv$1 can optimize train-time risks to increase susceptibility to multiple test-time risks, enabling collusion with several test-time adversaries.
- **Test $\mathcal{A}dv$1 ⇒ Multiple Test $\mathcal{A}dv$s**
  $\mathcal{A}dv$1 executes a test-time attack and uses the knowledge learned as a prerequisite to increase the effectiveness of other test-time attacks. This will allow $\mathcal{A}dv$1 to collude with multiple test-time adversaries.

For the above two cases, we can apply our guideline (from §3.3) between $\mathcal{A}dv$1 and each of the other test-time adversaries in parallel. The potential for overall collusion will be determined independently by the potential for each pairwise collusion. We now present a case for *sequential collusion*:

- **Test or Train $\mathcal{A}dv$1 ⇒ Test $\mathcal{A}dv$2 ⇒ Test $\mathcal{A}dv$3 ⇒ ⋯**
  $\mathcal{A}dv$1 colludes with $\mathcal{A}dv$2 to increase effectiveness of $\texttt{Att}_2$ executed by $\mathcal{A}dv$2, whose success provides knowledge en-

Table 9: **Impact of $\mathcal{A}dv$'s Characteristics on Collusion Potential ($\texttt{Att}_2$):** Rows correspond to $\texttt{Att}_2$; we vary $\mathcal{A}dv2$'s characteristics while $\mathcal{A}dv1$ is fixed. Strengthening $\mathcal{A}dv2$ raises effectiveness directly (not via collusion, hence omitted); we instead mark its effect on collusion potential with the listed $\texttt{Att}_1$ (s): ↓ → collusion potential decreases with characteristic; ✗ → collusion not required as characteristic meets the prerequisite factor. Columns under $\overline{\texttt{K}_\texttt{M}}$ are left empty for ModExt, as they are the outcomes of ModExt.

	$\overline{\texttt{K}_\texttt{D}}$			$\overline{\texttt{K}_\texttt{M}}$		
$\texttt{Att}_2$	$\texttt{Att}_1$	PartOvlap	FullOvlap	$\texttt{Att}_1$	GBox	WBox
Evsn	MemInf, DtRecon, AttInf, DistInf	↓	✗	ModExt	↓	✗
ModExt	MemInf, Evsn, DtRecon, AttInf, DistInf	↓	✗	—	—	—
MemInf	Evsn, DtRecon, AttInf, DistInf	↓	✗	ModExt	↓	✗
AttInf	MemInf, Evsn, DtRecon, DistInf	↓	✗	ModExt	↓	✗
DistInf	MemInf, Evsn, DtRecon, AttInf	↓	✗	ModExt	↓	✗
DtRecon	MemInf, Evsn, AttInf, DistInf	↓	✗	ModExt	↓	✗

abling downstream test-time collusion.

Here, we can apply our guideline sequentially to each pairwise collusion. The potential for overall collusion will be determined by the potential for pairwise collusions. More complex combinations of the parallel and sequential cases may exist, but our guideline can be readily extended to predict potential collusion.

Adaptive Strategies: One possible setting involves multi-round collusion, where $\mathcal{A}dv1$ iteratively adapts their attack strategy based on the success of $\mathcal{A}dv2$'s attack in the previous round. In this scenario, the effectiveness and collusion potential are expected to increase across rounds. Specifically, improvements in $\texttt{Att}_1$ by $\mathcal{A}dv1$ based on feedback from previous round, will further improve the effectiveness of $\texttt{Att}_2$ executed by $\mathcal{A}dv2$. This iterative amplification is captured by our guideline through an increase in correlation between $\texttt{Att}_1$ and the factors across successive rounds, consequently explaining the increase in effectiveness of $\texttt{Att}_2$.

Exceptions to Guideline: We do not claim our framework is perfect; rather, it is a first step toward systematically understanding and predicting the potential for collusion among adversaries. Our guideline cannot assess collusion when there are no common factors. While we did not encounter any such cases, further work would be needed to either uncover more relations between risks and factors to apply the guideline, or identify additional missing factors that can enable the collusion. This can then allow us to apply our guideline to evaluate collusion among adversaries.

Application of Framework and Guideline: Various stakeholders in the ML pipeline can be impacted by the increased effectiveness of colluding adversaries. For instance, *data owners*, *model owners*, and *service providers* become more susceptible to security, privacy, and fairness risks under collusion. Our framework helps identify possible sources of collusion, the enabling factors, and the potential impact of such collusion. *ML security researchers and practitioners* can use our framework to study previously unexplored collusion settings, develop stronger attacks for auditing, design defenses, and analyze enabling factors. Practitioners can further account for these factors to reduce collusion potential (see below), and use our guideline to efficiently assess collusion risks.

Potential Mitigations: By explicitly identifying the enabling factors through our framework and guideline, our work can help researchers and practitioners design defenses that disrupt collusion. For train-test collusion, defenses such as detecting and filtering poisoned samples in $\mathcal{D}_{tr}$ [92], modifying training objectives to reduce the influence of poisons [127], can reduce the impact of Poisn that enable subsequent attacks. Additionally, specific factors can be explicitly targeted to reduce correlations (e.g., reduce subgroup distinguishability using fairness constraints [1]). For test-time collusion, defenders can apply various transformations or filters on inputs and outputs to restrict the information that $\mathcal{A}dv$ can infer through interactions with $\mathcal{M}$. More broadly, combining complementary defenses that mitigate different risks can further reduce collusion potential [30].

Limitations and Future Work: We identify the following limitations and directions for future work:

- Our framework focuses on correlations rather than causation, since the limited interpretability of model behavior and the poor understanding of factor–attack dynamics make it difficult to causally identify enabling factors. As a result, our framework is conceptual and the proposed guideline is heuristic in nature, similar to Duddu et al. [29]. Developing a principled guideline with causal analysis is an important direction for future work.
- Our evaluation empirically validates the heuristic guideline without providing theoretical guarantees. We evaluate the framework by (a) demonstrating that it explains collusion in prior work and (b) empirically validating it on five illustrative case studies. However, the evaluation remains limited in establishing the general effectiveness of the guideline. Providing theoretical justification for different collusion settings and their corresponding conditions is an important future research direction.
- The scope of attacks explored in this work is limited to well-studied security, privacy, and fairness risks with sufficient prior literature to establish correlations and validate the guideline. Additional attacks are possible, as discussed in §7 under "Framework and Requirements: **R1** Comprehensive". However, further empirical analysis is required to establish the corresponding correlations and apply our guideline.
- Although not empirically evaluated, we analyzed prior work spanning different model types (e.g., generative models), settings (e.g., federated learning and retrieval-augmented generation), and modalities (e.g., text, images, and tabular

data). This suggests that the identified factors may generalize across diverse models and settings. However, our work does not provide a comprehensive evaluation of such generalization, which we leave for future work. Additionally, other generative-model-specific factors, currently excluded due to unexplored attack correlations, can be naturally incorporated into the framework.

Summary: Existing work lacks a systematic framework for exploring colluding adversaries in ML pipelines. We introduce the *first framework* by identifying factors which enable collusions. We propose a guideline to identify potential collusion using the factors, explain prior work, conjecture about unexplored ones, and empirically validate five such cases. We then study how adversaries' characteristics influence the potential for collusion, based on which we update our guideline.

Ethical Considerations

We discuss ethical considerations based on impact to various stakeholders, risks to ethics during the research and publication process, various mitigations to reduce ethical risks, justification for conducting research and publishing.

Impacts to Stakeholders: The primary stakeholders impacted by this work are: *data subjects* whose susceptibility to risks (e.g., to privacy or fairness) may worsen with colluding adversaries; *model owners and service providers* who face increased security and privacy failures when threats are evaluated as interactions among risks instead of isolated risks; *ML security researchers and practitioners* who may adopt our framework for auditing and designing defenses; and *society* as colluding adversaries can undermine trust in ML systems used in high-stakes domains.

Risks and Benefits of Research: We do not collect new data, involve human subjects, or target real-world systems. All empirical results validate conjectures using standard datasets and models from prior work. As such, the research process itself poses minimal direct risk to data subjects or institutions.

By explicitly characterizing train–test collusion (where training-time manipulation increases susceptibility to inference-time attacks) and test-time collusion (where outcomes of one inference attack satisfy the prerequisites of another), our framework may lower the barrier for adversaries to identify effective attack sequences.

However, the primary ethical benefit is defensive: our analysis reveals that evaluating risks in isolation can substantially underestimate real-world harm. In particular, our systematization shows that mitigations targeting a single risk may fail when adversaries collude across stages of the ML pipeline. This can help model designers account for such risks and design more effective defenses.

Mitigations: We do the following to limit potential harm:

- **Using Existing Attacks:** Our framework relies on previously published risks and attacks and show how existing attacks can be amplified under collusion.
- **Defensive Framing:** Collusions are presented as evaluation blind spots, intended to inform auditing and defense rather than to maximize adversarial effectiveness.
- **Scoped Empirical Validation:** Experiments are limited to validating conjectured relationships between risks, not to demonstrating real-world exploitability.

There is still risk that adversaries could use our framework for attack design, but this is an inherent trade-off in our work.

Decision to Conduct and Publish: We justify publishing this work under the principles of *beneficence and public interest*. Colluding adversaries already exist in practice; failing to make these interactions explicit leaves defenders unprepared and disproportionately harms data subjects. Therefore, we conclude that the benefits of publication outweigh the risks.

Open Science

The code for our experiments will be made publicly available upon publication. To align with USENIX Security's open science policy, the code is available at: https://zenodo.org/records/20429643. This includes all relevant datasets, scripts, binaries, and source code.

Acknowledgments

This work is supported in part by the Wallenberg Visiting Professor Program, the Natural Sciences and Engineering Research Council of Canada (grant number RGPIN-2026-04826), and the Government of Ontario (RE011-038). Vasisht and Lipeng are supported by David R. Cheriton Scholarship. Vasisht is supported by an IBM PhD Fellowship and Mastercard's Cybersecurity and Privacy Excellence Graduate Scholarship. Views expressed in the paper are those of the authors and do not reflect the position of the funding agencies. The authors thank Anudeep Das, Adam Caulfield, Prach Chantasantitam, and Yihan Wang for their feedback.

References

[1] Jan Aalmoes, Vasisht Duddu, and Antoine Boutet. On the alignment of group fairness with attribute privacy. In *WISE*, pages 333–348, 2025.

[2] Gorka Abad et al. Sok: A systematic evaluation of backdoor trigger characteristics in image classification. In *arXiv:2302.01740*, 2023.

[3] Michael Aerni et al. Measuring non-adversarial reproduction of training data in large language models. In *arXiv:2411.10242*, 2024.

[4] Maksym Andriushchenko et al. Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV*, pages 484–501, 2020.

[5] Gilad Asharov et al. Static vs. adaptive security in perfect MPC: A separation and the adaptive security of BGW. In *Cryptology ePrint Archive, Paper 2022/758*, 2022.

[6] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *USENIX Security*, pages 1505–1521, 2021.

[7] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *USENIX Security*, pages 515–532, August 2019.

[8] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs #1. In *CRYPTO*, pages 1–12, 1998.

[9] Vincenzo Carletti et al. Sok: Gradient inversion attacks in federated learning. In *USENIX Security Symposium (USENIX SEC)*, 2025.

[10] Nicholas Carlini et al. Extracting training data from large language models. In *USENIX Security*, pages 2633–2650, 2021.

[11] Nicholas Carlini et al. Membership inference attacks from first principles. In *SP*, pages 1897–1914, 2022.

[12] Nicholas Carlini et al. The privacy onion effect: Memorization is relative. In *NeurIPS*, 2022.

[13] Nicholas Carlini et al. Extracting training data from diffusion models. In *USENIX Security*, 2023.

[14] Nicholas Carlini et al. Stealing part of a production language model. In *arXiv:2403.06634*, 2024.

[15] Melissa Chase et al. Property inference from poisoning. In *SP*, 2022.

[16] Harsh Chaudhari et al. Snap: Efficient extraction of private properties with poisoning. In *SP*, pages 400–417, 2023.

[17] Chen Chen et al. Killing one bird with two stones: model extraction and attribute inference attacks against bert-based apis. In *arXiv:2105.10909*, 2021.

[18] Huiqiang Chen et al. Privacy and fairness in federated learning: On the perspective of tradeoff. *ACM Comput. Surv.*, 56, sep 2023.

[19] Yufei Chen et al. Amplifying membership exposure via data poisoning. In *NeurIPS*, pages 29830–29844, 2022.

[20] Zitao Chen and Karthik Pattabiraman. A method to facilitate membership inference attacks in deep learning models. In *NDSS*, 2025.

[21] Seungju Cho et al. Long-tailed adversarial training with self-distillation. In *ICLR*, 2025.

[22] Christopher A. Choquette-Choo et al. Label-only membership inference attacks. In *ICML*, pages 1964–1974, 2021.

[23] Antonio Emanuele Cinà et al. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Comput. Surv.*, 55, 2023.

[24] Antonio Emanuele Cinà et al. Energy-latency attacks via sponge poisoning. *Information Sciences*, 702:121905, 2025.

[25] Ambra Demontis et al. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *USENIX Security*, pages 321–338, 2019.

[26] Ruyi Ding et al. Vertexserum: Poisoning graph neural networks for link inference. In *ICCV*, pages 4532–4541, 2023.

[27] Jinhao Duan et al. Are diffusion models vulnerable to membership inference attacks? In *ICML*, 2023.

[28] Michael Duan et al. Do membership inference attacks work on large language models? In *arXiv:2402.07841*, 2024.

[29] Vasisht Duddu et al. Sok: Unintended interactions among machine learning defenses and risks. In *SP*, pages 2996–3014, 2024.

[30] Vasisht Duddu et al. Combining machine learning defenses without conflicts. In *TMLR*, 2025.

[31] Hao Fang et al. Gifd: A generative gradient inversion method with feature domain optimization. In *ICCV*, pages 4967–4976, 2023.

[32] Lucas Fenaux and Florian Kerschbaum. SoK: Analyzing adversarial examples: A framework to study adversary knowledge. In *arXiv 2402.14937*, 2024.

[33] Ryan Feng et al. Stateful defenses for machine learning models are not yet secure against black-box attacks. In *CCS*, pages 786–800, 2023.

[34] Shanglun Feng and Florian Tramèr. Privacy backdoors: Stealing data with corrupted pretrained models. In *ICML*, 2024.

[35] Julien Ferry et al. Sok: Taming the triangle–on the interplays between fairness, interpretability and privacy in machine learning. *arXiv:2312.16191*, 2023.

[36] Ferdinando Fioretto et al. Differential privacy and fairness in decisions and learning tasks: A survey. In *IJCAI*, pages 5470–5477, 2022.

[37] Liam Fowl et al. Adversarial examples make strong poisons. In *NeurIPS*, pages 30339–30351, 2021.

[38] Honglin Fu et al. Perseus: Tracing the masterminds behind cryptocurrency pump-and-dump schemes. *arXiv preprint arXiv:2503.01686*, 2025.

[39] Nicholas Furth et al. Un-fair trojan: Targeted backdoor attacks against model fairness. In *SDS*, pages 1–9, 2022.

[40] Isabel O Gallegos et al. Bias and fairness in large language models: A survey. *Computational Linguistics*, pages 1–79, 2024.

[41] Jonas Geiping et al. Inverting gradients - how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems*, pages 16937–16947, 2020.

[42] Alex Gittens et al. An adversarial perspective on accuracy, robustness, fairness, and privacy: Multilateral-tradeoffs in trustworthy ml. *IEEE Access*, 10:120850–120865, 2022.

[43] Xueluan Gong et al. Inversenet: Augmenting model extraction attacks with training data inversion. In *IJCAI*, pages 2439–2447, 2021.

[44] Kathrin Grosse et al. Adversarial initialization - when your network performs the way i want -. *ArXiv e-prints*, February 2019.

[45] Kathrin Grosse et al. On the security relevance of initial weights in deep neural networks. In *ICANN*, pages 3–14, Cham, 2020. Springer International Publishing.

[46] Jindong Gu et al. A survey on transferability of adversarial examples across deep neural networks. *Transactions on Machine Learning Research (TMLR)*, 2024.

[47] Halcyon. What is an initial access broker (iab)? Accessed 2026-05-18.

[48] Weizhe Hua et al. Reverse engineering convolutional neural networks through side-channel information leaks. In *DAC*, 2018.

[49] Bargav Jayaraman and David Evans. Are attribute inference attacks just imputation? In *CCS*, pages 1569–1582, 2022.

[50] Wenbo Jiang et al. Adversarial robustness poisoning: Increasing adversarial vulnerability of the model via data poisoning. In *GLOBECOM*, pages 4286–4291, 2024.

[51] Qixuan Jin et al. TOGA: Trigger optimization for clean data ordering backdoor attack, 2026.

[52] Mika Juuti et al. Prada: protecting against dnn model stealing attacks. In *EuroS&P*, pages 512–527, 2019.

[53] Kalpesh Krishna et al. Thieves of sesame street: Model extraction on bert-based apis. In *ICLR*, 2020.

[54] Nicolas Küchler et al. Architectural backdoors for within-batch data stealing and model inference manipulation. In *arXiv 2505.18323*, 2025.

[55] Harry Langford et al. Architectural Neural Backdoors from First Principles . In *SP*, pages 60–60, 2025.

[56] Hao Li et al. Enhanced label-only membership inference attacks with fewer queries. In *USENIX Security*, 2025.

[57] Yiming Li et al. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35:5–22, 2022.

[58] Ziang Li et al. From head to tail: Efficient black-box model inversion attack via long-tailed learning. In *CVPR*, pages 29288–29298, 2025.

[59] Yugeng Liu et al. {ML-Doctor}: Holistic risk assessment of inference attacks against machine learning models. In *USENIX Security*, pages 4525–4542, 2022.

[60] Yugeng Liu et al. Amplifying machine learning attacks through strategic compositions. In *arXiv 2506.18870*, 2025.

[61] Sasha Luccioni et al. Stable bias: Evaluating societal representations in diffusion models. In *NeurIPS*, 2024.

[62] Nils Lukas et al. Analyzing leakage of personally identifiable information in language models. In *SP*, pages 346–363, 2023.

[63] Nils Lukas et al. Leveraging optimization for adaptive attacks on image watermarks. In *ICLR*, 2024.

[64] Xinjian Luo et al. Exploring privacy and fairness risks in sharing diffusion models: An adversarial perspective. *IEEE TIFS*, 2024.

[65] Yukui Luo et al. Deepstrike: Remotely-guided fault injection attacks on dnn accelerator in cloud-fpga. In *DAC*, page 295–300, 2022.

[66] Mohammad Malekzadeh et al. Honest-but-curious nets: Sensitive attributes of private inputs can be secretly coded into the classifiers' outputs. In *CCS*, pages 825–844, 2021.

[67] Syed Irfan Ali Meerza and Jian Liu. Eab-fl: Exacerbating algorithmic bias through model poisoning attacks in federated learning. In *IJCAI*, pages 458–466, 2024.

[68] Ninareh Mehrabi et al. Exacerbating algorithmic bias through fairness attacks. In *AAAI*, pages 8930–8938, 2021.

[69] Ninareh Mehrabi et al. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54:1–35, 2021.

[70] Luca Melis et al. Exploiting unintended feature leakage in collaborative learning. In *SP*, pages 691–706, 2019.

[71] Xiangtao Meng et al. From defender to devil? unintended risk interactions induced by llm defenses. *arXiv:2510.07968*, 2025.

[72] Ali Naseh et al. Backdooring bias into text-to-image models. In *arXiv:2406.15213*, 2024.

[73] Milad Nasr et al. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *SP*, pages 739–753, 2019.

[74] Seong Joon Oh et al. Towards reverse-engineering black-box neural networks. In *ICLR*, 2018.

[75] Daryna Oliynyk et al. I know what you trained last summer: A survey on stealing machine learning models and defences. *ACM Comput. Surv.*, 55, July 2023.

[76] Tribhuvanesh Orekondy et al. Knockoff nets: Stealing functionality of black-box models. In *CVPR*, pages 4954–4963, 2019.

[77] Ashwinee Panda et al. Teach llms to phish: Stealing private information from language models. In *ICLR*, 2024.

[78] Ren Pang et al. A tale of evil twins: Adversarial inputs versus poisoned models. In *CCS*, pages 85–99, 2020.

[79] Nicolas Papernot et al. Practical black-box attacks against machine learning. In *AsiaCCS*, pages 506–519, 2017.

[80] Nicolas Papernot et al. SoK: Security and privacy in machine learning. In *EuroS&P*, pages 399–414, 2018.

[81] Yuefeng Peng et al. Data extraction attacks in retrieval-augmented generation via backdoors. In *arXiv:2411.01705*, 2024.

[82] Minh Pham et al. Circumventing concept erasure methods for text-to-image generative models. In *ICLR*, 2024.

[83] Adnan Siraj Rakin et al. Deep-Dup: An adversarial weight duplication attack framework to crush deep neural network in Multi-Tenant FPGA. In *USENIX Security*, pages 1919–1936, 2021.

[84] Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *ACM Comput. Surv.*, 56, 2023.

[85] Ahmed Salem et al. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *NDSS*. The Internet Society, 2019.

[86] Ahmed Salem et al. SoK: Let the privacy games begin! a unified treatment of data inference privacy in machine learning. In *SP*, pages 327–345, 2023.

[87] Reza Shokri et al. Membership inference attacks against machine learning models. In *SP*, pages 3–18, 2017.

[88] Ilia Shumailov et al. Manipulating sgd with data ordering attacks. In *NeurIPS*, 2021.

[89] Ilia Shumailov et al. Sponge examples: Energy-latency attacks on neural networks. In *EuroS&P*, pages 212–231, 2021.

[90] David Solans et al. Poisoning attacks on algorithmic fairness. In *ECML-KDD*, pages 162–177, 2020.

[91] Congzheng Song et al. Machine learning models that remember too much. In *CCS*, pages 587–601, 2017.

[92] Ezekiel Soremekun et al. Towards backdoor attacks and defense in robust machine learning models. *Computers & Security*, 127:103101, 2023.

[93] M. Strobel and R. Shokri. Data privacy and trustworthy machine learning. In *IEEE Security & Privacy*, pages 2–7, 2022.

[94] Anshuman Suri et al. Dissecting distribution inference. In *SaTML*, pages 150–164, 2023.

[95] Anshuman Suri and David Evans. Formalizing and estimating distribution inference risks. *PETS*, 2022.

[96] Fnu Suya et al. SoK: Pitfalls in evaluating black-box attacks. In *SaTML*, pages 387–407, 2024.

[97] Sebastian Szyller and N Asokan. Conflicting interactions among protection mechanisms for machine learning models. In *AAAI*, pages 15179–15187, 2023.

[98] Yulong Tian et al. Manipulating Transfer Learning for Property Inference . In *CVPR*, pages 15975–15984, 2023.

[99] Florian Tramèr et al. Stealing machine learning models via prediction apis. In *USENIX Security*, page 601–618, 2016.

[100] Florian Tramer et al. On adaptive attacks to adversarial example defenses. In *NeurIPS*, pages 1633–1645, 2020.

[101] Florian Tramèr et al. Truth serum: Poisoning machine learning models to reveal their secrets. In *CCS*, pages 2779–2792, 2022.

[102] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *IEEE SP*, pages 36–52, 2018.

[103] Lijin Wang et al. Property existence inference against generative models. In *USENIX Security*, pages 2423–2440, 2024.

[104] Linlin Wang et al. Bias amplification in rag: Poisoning knowledge retrieval to steer llms. In *arXiv 2506.11415*, 2025.

[105] Zhibo Wang et al. Poisoning-assisted property inference attack against federated learning. *IEEE Transactions on Dependable and Secure Computing*, 20:3328–3340, 2023.

[106] Zijian Wang et al. Energy-latency attacks to on-device neural networks via sponge poisoning. In *SecTL Workshop*, 2023.

[107] Rui Wen et al. Sok: Data reconstruction attacks against machine learning models: Definition, metrics, and benchmark. In *USENIX SEC*, pages 5601–5620, 2025.

[108] Yuxin Wen et al. Canary in a coalmine: Better membership inference with ensembled adversarial queries. In *ICLR*, 2023.

[109] Yuxin Wen et al. Privacy backdoors: Enhancing membership inference through poisoning pre-trained models. In *NeurIPS*, 2024.

[110] Lei Wu and Zhanxing Zhu. Towards understanding and improving the transferability of adversarial examples in deep neural networks. In *ACML*, pages 837–850, 2020.

[111] Yaxin Xiao et al. Mexmi: Pool-based active model extraction crossover membership inference. In *NeurIPS*, 2022.

[112] Yi Xie et al. Same: Sample reconstruction against model extraction attacks. In *AAAI*, volume 38, pages 19974–19982, 2024.

[113] Mengjia Yan et al. Cache telepathy: Leveraging shared resource attacks to learn DNN architectures. In *USENIX Security*, pages 2003–2020, August 2020.

[114] Ziqi Yang et al. Neural network inversion in adversarial setting via background knowledge alignment. In *CCS*, page 225–240, 2019.

[115] Fan Yao et al. DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *USENIX Security*, pages 1463–1480, 2020.

[116] Kai Yao and Marc Juarez. Sok: What makes private learning unfair? In *SaTML*, pages 841–857, 2025.

[117] Jiayuan Ye et al. Leave-one-out distinguishability in machine learning. In *ICLR*, 2024.

[118] Samuel Yeom et al. Privacy risk in machine learning: Analyzing the connection to overfitting. In *CSF*, pages 268–282, 2018.

[119] Sibo Yi et al. Jailbreak attacks and defenses against large language models: A survey. In *arXiv:2407.04295*, 2024.

[120] Peng Yifeng et al. Swallowing the poison pills: Insights from vulnerability disparity among llms. In *arXiv:2502.18518*, 2025.

[121] Xinli Yue et al. Revisiting adversarial training under long-tailed distributions. In *CVPR*, pages 24492–24501, 2024.

[122] Wanrong Zhang et al. Leakage of dataset properties in Multi-Party machine learning. In *USENIX Security*, pages 2687–2704, 2021.

[123] Yanjun Zhang et al. Agrevader: Poisoning membership inference against byzantine-robust federated learning. In *WWW*, pages 2371–2382, 2023.

[124] Yuheng Zhang et al. The secret revealer: Generative model-inversion attacks against deep neural networks. In *CVPR*, 2020.

[125] B. Zhao et al. On the (in)feasibility of attribute inference attacks on machine learning models. In *EuroS&P*, pages 232–251, 2021.

[126] Joshua C. Zhao et al. Loki: Large-scale Data Reconstruction Attack against Federated Learning through Model Manipulation . In *SP*, pages 1287–1305, 2024.

[127] Qi Zhao and Christian Wressnegger. Adversarially robust anti-backdoor learning. In *AISec*, pages 77–88, 2024.

[128] Shuai Zhao et al. Weak-to-strong backdoor attack for large language models. In *arXiv:2409.17946*, 2024.

[129] Mengxin Zheng et al. Trojfair: trojan fairness attacks. In *LAMPS*, 2023.

[130] Junhao Zhou et al. Property inference attacks against gans. In *arXiv:2111.07608*, 2021.

[131] Ligeng Zhu et al. Deep leakage from gradients. In *NeurIPS*, 2019.

[132] Tianqing Zhu et al. Label-only model inversion attacks: Attack with the least information. *IEEE TIFS*, 18:991–1005, 2023.

Appendix

A Streamlined Threat Model

We represent an adversary model as **TM** which is a tuple of sets representing different characteristics of $\mathcal{A}dv$. To define the threat model, we identify various $\mathcal{A}dv$'s characteristics: objectives, capabilities, knowledge, and access to $\mathcal{M}$ and $\mathcal{D}_{tr}$. Formally, we represent the threat model as a tuple of seven sets (denoted as $\overline{<.>}$), corresponding to the questions below: (i) what role(s) does $\mathcal{A}dv$ play? ($\overline{\texttt{Role}}$) (ii) what are $\mathcal{A}dv$'s objectives? ($\overline{\texttt{Obj}}$) (iii) what are $\mathcal{A}dv$'s capabilities? ($\overline{\texttt{Cap}}$) (iv) what is $\mathcal{A}dv$'s interaction with $\mathcal{M}$? ($\overline{\texttt{I}}$) (v) what attack optimizations can $\mathcal{A}dv$ use? ($\overline{\texttt{Opt}}$) (vi) what is $\mathcal{A}dv$'s knowledge about $\mathcal{M}$? ($\overline{\texttt{K}_\texttt{M}}$); and (vii) what is $\mathcal{A}dv$'s knowledge about data? ($\overline{\texttt{K}_\texttt{D}}$) Given these characteristics, we denote a threat model **TM** describing $\mathcal{A}dv$ as a tuple of the seven sets:

$$\texttt{TM} = \left(\overline{\texttt{Obj}}, \overline{\texttt{Role}}, \overline{\texttt{Cap}}, \overline{\texttt{Opt}}, \overline{\texttt{K}_\texttt{M}}, \overline{\texttt{I}}, \overline{\texttt{K}_\texttt{D}} \right)$$

The elements of each of the seven sets constituting **TM** are drawn from their respective spaces of all possible values. We denote these spaces with the first letter capitalized: $\overline{\texttt{Obj}}$ as the space for $\overline{\texttt{Obj}}$, $\overline{\texttt{Role}}$ for $\overline{\texttt{Role}}$, $\overline{\texttt{Cap}}$ for $\overline{\texttt{Cap}}$, $\overline{\texttt{Opt}}$ for $\overline{\texttt{Opt}}$, $\overline{\texttt{K}_\texttt{M}}$ for $\overline{\texttt{K}_\texttt{M}}$, $\overline{\texttt{I}}$ for $\overline{\texttt{I}}$, and $\overline{\texttt{K}_\texttt{D}}$ for $\overline{\texttt{K}_\texttt{D}}$. We describe the possible values in each space characterizing $\mathcal{A}dv$. This representation also allows us to define an operation on the tuples to combine adversary models for capturing colluding adversaries. We outline each set and their possible values, provide an overview of the ML pipeline and involved entities (Figure 1).

Role of $\mathcal{A}dv$ ($\overline{\texttt{Role}}$) specifies who $\mathcal{A}dv$ is. This includes data owner (DtOwnr), data provider (DtProv), code provider (CodeProv), model provider (ModProv), model owner (ModOwnr), model trainer (ModTrnr), service provider (SrvProv), and client (Clnt).

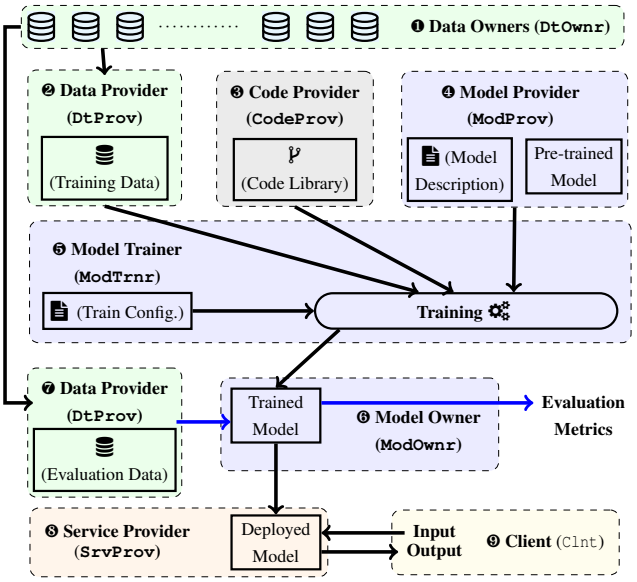


Figure 1: **ML Pipeline.** Raw data from data owners (DtOwnr) is aggregated by a data provider (DtProv) and processed into $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$. A model trainer (ModTrnr) uses architecture from a model provider (ModProv), code from a code provider (CodeProv), and a training configuration to train or fine-tune $\mathcal{M}$. The resulting $\mathcal{M}$ is owned by a model owner (ModOwnr), who evaluates it using $\mathcal{D}_{te}$. Finally, a service provider (SrvProv) deploys the model for clients (Clnt).

❶ **Data Owner** (DtOwnr) can share data taken from several sources (e.g., scraping from the Internet, personal data, software repositories, etc.). This is later used as $\mathcal{D}_{tr}$ (see ❷) or $\mathcal{D}_{te}$ (see ❼). As $\mathcal{A}dv$, DtOwnr can add poisons/backdoors to $\mathcal{D}_{tr}$ to degrade utility or give $\mathcal{A}dv$-chosen outputs.

❷, ❼ **Data Provider** (DtProv) processes data from data owners to get $\mathcal{D}_{tr}$ (for training) and $\mathcal{D}_{te}$ (for evaluation). As $\mathcal{A}dv$, DtProv can add poisons or backdoors to $\mathcal{D}_{tr}$ for degrading utility or giving $\mathcal{A}dv$-chosen outputs.

❸ **Code Provider** (CodeProv) includes third-party libraries (e.g., PyTorch, or code repositories on GitHub) used for training and evaluating $\mathcal{M}$. As $\mathcal{A}dv$, CodeProv can inject malicious code to influence $\mathcal{M}$'s behavior (e.g., [6]).

❹ **Model Provider** (ModProv) shares the architecture of $\mathcal{M}$ or pre-trained model (e.g., HuggingFace), which can be downloaded for further use (e.g., fine-tuning). As $\mathcal{A}dv$, ModProv can share a backdoored $\mathcal{M}$ (e.g. [55]).

❺ **Model Trainer** (ModTrnr) uses an architecture or a pre-trained model from ModProv, source code from CodeProv, and $\mathcal{D}_{tr}$ from DtProv, to train or fine-tune $\mathcal{M}$. As $\mathcal{A}dv$, ModTrnr can tamper the training configuration to influence $\mathcal{M}$'s behavior (e.g., [6]).

❻ **Model Owner** (ModOwnr) owns the trained $\mathcal{M}$ from ModTrnr, and gives to SrvProv for deployment. As $\mathcal{A}dv$,

ModOwnr can tamper $\mathcal{M}$ to influence its behavior.

❽ **Service Provider** (SrvProv) offers the trained $\mathcal{M}$ as a service with an API to send queries and receive outputs. As $\mathcal{A}dv$, SrvProv mounts various inference attacks (MemInf, AttInf, DistInf, DtRecon) to leak information in $\mathcal{D}_{tr}$, measure its bias (Discr), or deploy a tampered model.

❾ **Client** (Clnt) as $\mathcal{A}dv$ can mount various privacy attacks (MemInf, AttInf, DistInf, DtRecon), steal $\mathcal{M}$'s functionality (ModExt), and measure its bias (Discr). Alternatively, Clnt can download a pre-trained $\mathcal{M}$ (e.g., language models from Hugging Face) and query it locally. In this case, ModExt does not apply, as Clnt already has full access to $\mathcal{M}$.

$$\overline{\texttt{Role}} = \left\{ \begin{array}{l} \texttt{DtOwnr}, \texttt{DtProv}, \texttt{CodeProv}, \texttt{ModProv}, \\ \texttt{ModTrnr}, \texttt{ModOwnr}, \texttt{SrvProv}, \texttt{Clnt} \end{array} \right\}$$

$\mathcal{A}dv$'s Objectives ($\overline{\texttt{Obj}}$) includes the risks from §2.

$$\overline{\texttt{Obj}} = \left\{ \begin{array}{l} \texttt{Evsn}, \texttt{Poisn}, \texttt{Bkdr}, \texttt{MemInf}, \texttt{AttInf}, \\ \texttt{DistInf}, \texttt{DtRecon}, \texttt{ModExt}, \texttt{Discr} \end{array} \right\}$$

$\mathcal{A}dv$'s Capability ($\overline{\texttt{Cap}}$) specifies whether or not $\mathcal{A}dv$ deviates from the computation prescribed for their role. This includes **honest-but-curious** (HbC) or **malicious** (Mal), which have also been called "passive" and "active" respectively (e.g., [70, 73]). HbC cannot deviate from prescribed procedure, but can observe (and use) all visible data. Mal can deviate to do arbitrary computations on top.

$$\overline{\texttt{Cap}} = \{\texttt{HbC}, \texttt{Mal}\}$$

Optimization ($\overline{\texttt{Opt}}$) indicates the type of attack optimization performed by $\mathcal{A}dv$. An **adaptive** $\mathcal{A}dv$ (Adpt) modifies their subsequent attack queries to $\mathcal{M}$, based on the feedback from $\mathcal{M}$'s previous outputs[8]. Alternatively, **non-adaptive** $\mathcal{A}dv$ (nonAdpt) use a fixed set of attack queries or optimize them locally without feedback from $\mathcal{M}$.

$$\overline{\texttt{Opt}} = \{\texttt{Adpt}, \texttt{nonAdpt}\}$$

Knowledge of Model ($\overline{\texttt{K}_\texttt{M}}$) includes **whitebox** (WBox), **blackbox** (BBox), **graybox** (GBox), and **no access** (NoAccess). For both BBox and GBox, we have **full predictions** (fullPred), **top-k predictions** (topK), and **hard labels** (hardLbl).

- **Whitebox** (WBox): $\mathcal{A}dv$ has full knowledge of $\mathcal{M}$'s architecture, parameters, training process and training configuration (e.g., downloaded publicly available pre-trained models). Since $\mathcal{A}dv$ has access to $\mathcal{M}$'s parameters, they can compute gradients and intermediate activations.
- **Graybox** (GBox): $\mathcal{A}dv$ knows the architecture and training configuration but not the parameters.
- **Blackbox** (BBox): $\mathcal{A}dv$ does not know $\mathcal{M}$'s architecture, parameters, training process and configuration. For both BBox and GBox, we have the following cases:
  - **Hard-Label** (hardLbl): $\mathcal{A}dv$ only receives the final classification for a given input, such as malware classifiers producing labels (e.g., malware or not).
  - **Top-K Predictions** (topK): $\mathcal{A}dv$ obtains top-k confidence scores from the prediction vector ($1 \leq k \leq N$) with N classes (e.g., ImageNet models output scores for a subset of classes).
  - **Full Predictions** (fullPred): $\mathcal{A}dv$ receives confidence scores for all classes (e.g., in high-stakes ML models like credit approval).
- **No access** (NoAccess): $\mathcal{A}dv$ cannot query $\mathcal{M}$ and can access the outputs for some inputs.

$$\overline{\texttt{K}_\texttt{M}} = \left\{ \begin{array}{l} \texttt{NoAccess}, (\texttt{BBox}, \texttt{GBox}) \\ \times (\texttt{hardLbl}, \texttt{topK}, \texttt{fullPred}), \texttt{WBox}, \end{array} \right\}$$

We use (BBox, GBox) × (hardLbl, topK, fullPred) to denote their six possible combinations. WBox corresponds to fullPred (WBox.fullPred), which we omit for brevity.

Interaction ($\overline{\texttt{I}}$) indicates how $\mathcal{A}dv$ interacts with $\mathcal{M}$: (i) **no access (NoAccess)**; (ii) **one-shot (1Shot)** where $\mathcal{A}dv$ can only query once (e.g., evading a malware classifier on a target machine); (iii) **k-Shot (KShot)** where $\mathcal{A}dv$ can send k queries to $\mathcal{M}$ (e.g., in case of budget constraints); (iv) **unlimited (Unlmtd)** where $\mathcal{A}dv$ has no limit on the number of queries to $\mathcal{M}$ (e.g., with local access to $\mathcal{M}$).

$$\overline{\texttt{I}} = \{\texttt{NoAccess}, \texttt{1Shot}, \texttt{KShot}, \texttt{Unlmtd}\}$$

Knowledge of Data ($\overline{\texttt{K}_\texttt{D}}$) indicates the assumptions about $\mathcal{A}dv$'s auxiliary data $\mathcal{D}^{aux}$ for designing ($\mathcal{D}^{aux}_{tr}$) and evaluating ($\mathcal{D}^{aux}_{te}$) the attack. For $\mathcal{D}^{aux}_{tr}$, we indicate whether $\mathcal{A}dv$ has **no access** (NoAccess), **no overlap** (NoOvlap), **partial overlap** (PartOvlap) or **full overlap** (FullOvlap) with $\mathcal{D}_{tr}$. For $\mathcal{D}^{aux}_{te}$, we indicate if $\mathcal{A}dv$ has ground truth: **non-blind** (NoBlind) when available, and **blind** (Blind) otherwise (e.g., unavailable as labeling is expensive).

- **No Access** (NoAccess): Some $\mathcal{A}dv$ in ML pipeline will not have access to $\mathcal{D}_{tr}$ (e.g., CodeProv or ModProv).
- **No Overlap** (NoOvlap): $\mathcal{A}dv$ knows the underlying distribution of $\mathcal{D}_{tr}$ from which $\mathcal{A}dv$ samples $\mathcal{D}^{aux}_{tr}$ but has no overlap with $\mathcal{D}_{tr}$. Also, $\mathcal{D}^{aux}_{tr}$ can include natural data or synthetically generated data.
- **Partial Overlap** (PartOvlap): $\mathcal{D}^{aux}_{tr}$ has partial overlap with $\mathcal{D}_{tr}$ (e.g., part of $\mathcal{D}_{tr}$ is public).
- **Complete Overlap** (FullOvlap): $\mathcal{D}^{aux}_{tr}$ is the same as $\mathcal{D}_{tr}$ including features and labels (e.g., $\mathcal{D}_{tr}$ is public for GPT2).

We denote the possible combinations as {NoOvlap, PartOvlap, FullOvlap} × {NoBlind, Blind}.

$$\overline{\texttt{K}_\texttt{D}} = \left\{ \begin{array}{l} (\texttt{NoOvlap}, \texttt{PartOvlap}, \texttt{FullOvlap}) \\ \times (\texttt{NoBlind}, \texttt{Blind}) \end{array} \right\}$$

[8] **Terminology:** ML literature misuses "adaptive" to describe $\mathcal{A}dv$ aware of defenses [32, 33, 63, 100], whereas in security literature, such knowledge is assumed by default. We call adversaries without this knowledge "naïve". Following standard security terminology [5, 8], we define "adaptive" adversaries as those who update their attacks based on previous responses, and assume $\mathcal{A}dv$ knows the defenses by default.