arXiv:2603.01053v1 [cs.CR] 1 Mar 2026

从黑盒到白盒：数据集蒸馏泄漏

Huajie Chen[1], Tianqing Zhu[*], Yuchen Zhong[1], Yang Zhang[2], Shang Wang[3], Feng He[3], Lefeng Zhang[1], Jialiang Shen[4], Minghao Wang[1], Wanlei Zhou[1]

摘要

数据集蒸馏将大型真实数据集压缩为小型合成数据集，使基于合成数据训练的模型能够达到与基于真实数据训练的模型相当的性能。尽管合成数据集通常被认为具有隐私保护能力，但我们发现，现有蒸馏方法可能导致严重的隐私泄露，因为合成数据集会隐式编码蒸馏模型的权重轨迹，从而包含过多信息，易被攻击者利用。为揭示这一风险，我们针对最先进的蒸馏技术提出信息泄露攻击（IRA）。IRA 分为三个阶段：架构推断、成员推断和模型反演，每个阶段都会利用合成数据中的隐藏信息。在架构推断阶段，攻击者通过记录模型在合成数据集上训练时的损失，获得损失轨迹，并据此训练攻击模型，以预测蒸馏算法和模型架构。利用预测得到的架构，攻击者可以训练一个在结构和性能上都与受害者模型相仿的本地模型，从而有效地将黑盒场景转变为白盒场景，其能力超过了以往攻击中使用的影子模型。接下来，在成员推断阶段，攻击者利用对本地模型的完全访问权限，训练一个攻击模型，通过本地模型的隐藏层和最终层输出，判断某个样本是否属于真实数据集。最后，在模型反演阶段，攻击者尝试重建真实样本。我们提出了一种增强型双网络扩散框架，可对生成器施加约束；此外，我们还提出轨迹损失，通过利用合成数据集中蕴含的更深层信息，引导生成器逼近真实数据分布。实验表明，IRA 能够准确预测蒸馏算法和模型架构，并成功推断成员身份、恢复真实数据集中的敏感样本。我们的代码将在论文录用后发布。

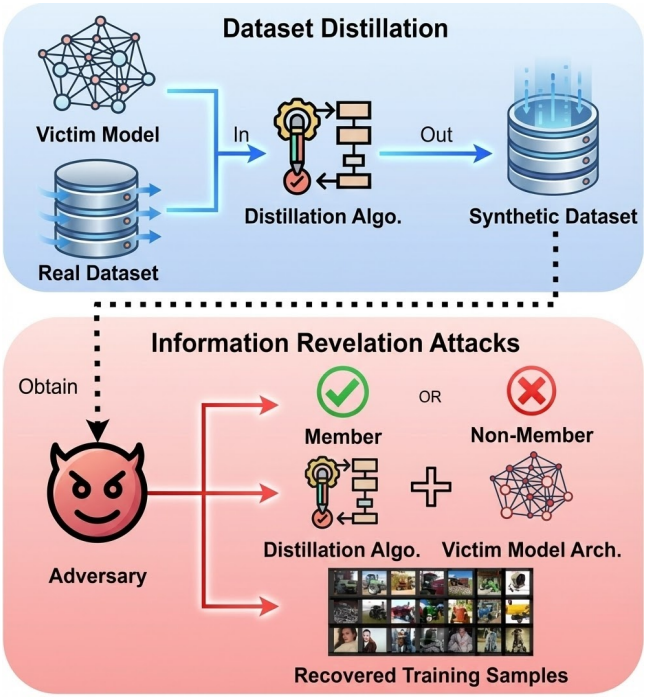


图 1：IRA 概览。受害者使用受害者模型和真实数据集，通过蒸馏算法生成公开可用的合成数据集。然而，攻击者可以利用该合成数据集发起 IRA，从而揭露敏感信息。

CCS 概念

•**安全与隐私**;

关键词

数据集蒸馏、隐私攻击、深度学习

ACM 参考格式：

Huajie Chen[1], Tianqing Zhu[*, 1], YuchenZhong[1], Yang Zhang[2], Shang Wang[3], Feng He[3], Lefeng Zhang[1], Jialiang Shen[4], Minghao Wang[1], Wanlei Zhou[1]。2018。从黑盒到白盒：数据集蒸馏泄漏。收录于《请根据版权信息填写正确的会议名称》论文集

确认邮件（会议缩写 'XX）。ACM，纽约州纽约，美国，22 页。
https://doi.org/XXXXXXX.XXXXXXX

*通讯作者。 [1]澳门城市大学。 [2]CISPA亥姆霍兹信息安全中心。 [3]悉尼科技大学。 [4]悉尼大学。会议缩写 'XX，纽约州伍德斯托克 © 2018 版权所有者/作者所有。出版权已授权给 ACM。ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 引言

数据集蒸馏 [1, 2]目前是一种热门技术，旨在利用信息量极高的合成数据集，提升深度学习模型的训练效率。数据集蒸馏旨在将大型真实数据集浓缩为一个包含较少样本、但性能与使用原始数据集训练的模型相近的合成数据集。例如，Wang等人 [1] 基于包含 50，000 个样本的 CIFAR10 数据集，合成了一个仅包含 100 张合成图像的数据集。使用该数据集训练的模型，其表现与使用真实数据集训练的模型相近。此外，合成样本有别于真实样本，因为合成样本包含浓缩信息的特征，因此往往呈现出类似噪声图像的外观。因此，人们认为数据集蒸馏可以成为一种可认证隐私保护方案，并将在提升深度学习模型的训练效率方面发挥重要作用 [3–6]。

值得注意的是，最先进的数据集蒸馏算法能够生成高质量的合成数据集，其中包含源自使用真实数据集进行训练过程的权重轨迹信息 [7–9]。这类合成数据集使训练后的模型能够达到相同的性能，甚至超过使用原始数据集训练的模型。例如，Guo 等人 [8] 实现了无损数据集蒸馏，将 CIFAR-10 数据集蒸馏为一个包含 10，000 个样本的合成数据集（每类 1，000 张图像）。此外，该合成数据集甚至使模型达到 0。855 的测试准确率，超过了使用真实 CIFAR10 数据集训练的模型所达到的 0。848 测试准确率。

尽管数据集蒸馏带来了颇具前景的结果和优势，但数据集蒸馏目前正面临来自隐私攻击的严重威胁。在深度学习的隐私攻击的一般设置中，受害者模型通常对于攻击者而言是一个黑盒。这意味着攻击者只能查询受害者模型，以获取其输出供进一步使用。然而，如果攻击者能够正确推断出受害者模型的模型架构，那么攻击者就可以利用高质量的合成数据集训练一个与受害者模型具有相同架构和相似权重的本地模型。**这使攻击者能够将黑盒变成白盒**。我们认为，这是一个严重且尚未得到探索的问题，必然会导致隐私泄露，并使合成数据集遭受成员推断攻击 [10] 和模型反演攻击 [11]等安全和隐私攻击。

我们的工作。为揭示这一问题，我们提出了**首个针对数据集蒸馏方法的信息披露攻击（IRA）**，如图1所示。IRA包含三个阶段：架构推断、成员推断和模型反演。在架构推断阶段，攻击者旨在推断蒸馏过程中所使用的蒸馏算法和模型架构。攻击者首先使用模型架构和蒸馏算法的不同组合生成合成数据集。随后，利用这些合成数据集中的每一个，通过使用相同架构训练模型来生成损失轨迹。最终，利用损失轨迹训练一个攻击模型，该模型以损失轨迹为输入，预测蒸馏算法和模型架构。借助合成数据集和预测出的模型架构，攻击者现在可以为接下来的两个阶段训练一个本地模型。

在成员推断阶段，攻击者的目标是判断某个样本是否属于真实数据集。攻击者首先收集一个与真实数据集具有相同分布的辅助数据集。随后，利用本地模型在给定辅助数据集中的样本时其隐藏层和最终层的输出，训练一个攻击模型。接下来，对于任意未见样本，攻击模型可以尽其所能利用合成数据集中残留的信息，判断该样本是成员还是非成员。

在模型反演阶段，攻击者旨在恢复可视为真实数据集成员的数据样本。攻击者利用我们提出的双重扩散模型框架来学习真实数据集的分布。该框架使攻击者能够添加约束，迫使扩散模型的输出更贴近真实数据集中的样本。这些模型在辅助数据集上进行训练，并在本地模型的指导下进行训练。本研究提出的分类损失和轨迹损失旨在使模型利用本地模型和合成数据集中包含的信息。

为实现上述攻击目标，需要解决若干研究问题（RQ）：**RQ1：**为何能够识别蒸馏过程中所使用的模型架构？**RQ2：**攻击者如何收集有效的辅助数据集？**RQ3：**如何判断某个成员样本是否被成功恢复？

于是，我们有如下回答：**A1：**受Zhang等人 [12], 的启发，我们发现，模型架构可以通过模型在合成数据集上的训练过程中产生的损失轨迹加以区分。此外，我们进行了理论分析，讨论了合成数据集与相应损失轨迹之间的关系。随后，我们从理论上证明了架构推断攻击的可行性。**A2：**遵循成员推断攻击 [10], 的做法，攻击者可以访问Kaggle和Google Dataset Search等公共平台上的在线数据集。假设攻击者了解该任务，便能够收集与原始模型相似或承担相同任务的辅助数据集。**A3：**本文并不着重讨论评估任何模型反演攻击性能的指标。因此，我们沿用先前研究 [13, 14], , 将攻击准确率和KNN距离用作评估指标。若某个恢复样本能够被在真实数据集上训练的受害者模型分类到目标类别，则该样本被视为成员样本。同时，KNN距离较低的此类样本被认为在目标类别中更具代表性。

我们的贡献如下。

- 我们揭示，当前的数据集蒸馏方法既无法保护真实数据集的隐私，也无法保护蒸馏过程中所使用的模型。
- 我们提出了针对当前数据集蒸馏方法的首个信息揭示攻击。实验结果表明，IRA能够有效推断蒸馏过程中所使用的模型架构和蒸馏算法。攻击者利用合成数据集，以正确的模型架构训练本地模型，即可将黑盒受害者模型转变为白盒，从而助力成员推断和模型反演攻击。
- 我们对架构推断攻击进行了理论分析，其中深入讨论了合成数据集与损失轨迹之间的关系。我们从数学上证明了该攻击的可行性。

2 相关工作与预备知识

本文的主要符号说明列于表1。

2.1 数据集蒸馏

数据集蒸馏。 数据集蒸馏 [1]的目标是利用大型真实数据集 D_{real}创建合成数据集 D_{syn}，其中 $|D_{\text{syn}}| << |D_{\text{real}}|$。在 D_{syn} 上训练的模型，在测试集上的表现将与在 D_{real}上训练的模型相当。数据集蒸馏的首项工作 [1] 旨在直接最小化模型在

表1：主要符号说明。

符号	定义
V, A	受害者/攻击者
f, h	受害者/本地模型
θ, τ	受害者/本地模型的权重
$D_{\text{real}}, D_{\text{syn}}$	真实/合成数据集
$D_{\text{aux}}, D_{\text{inv}}$	辅助/反演数据集
b, c	成员身份/目标类别标签
$\mathcal{F}$	模型架构集合
Γ	蒸馏算法集合
x, y, z	样本/标签/随机噪声
L	损失函数
α_t, β_t	预定义常数
ϵ, λ	高斯噪声/权重参数
C, E	类别数/训练轮数
T, l	采样步数/合成集数量
M, N	输入维度/批量大小
A_M	成员推理模型
A_A	架构推理模型
ϕ, ψ	双扩散模型
T^L, T^W	损失/权重轨迹

D_{syn} 和 D_{real}上的损失，其定义为

$$\begin{aligned} D^*_{\text{syn}} &= \arg\min_{D_{\text{syn}}} L(D_{\text{real}}, \theta), \\ \text{s.t. } \theta^* &= \arg\min_{\theta} L(D_{\text{syn}}, \theta), \end{aligned} \tag{2.1}$$

其中 θ 表示模型的参数； $L(\cdot,\cdot)$是损失函数。在每次迭代中，D_{syn} 通过优化进行更新

$$D_{\text{syn}} \leftarrow D_{\text{syn}} - \eta \nabla_{D_{\text{syn}}} L(D_{\text{real}}, \theta), \tag{2.2}$$

其中 η 表示学习率。

数据集凝缩。 数据集凝缩 [2] 不仅要求在 D_{syn} 上训练的模型具有与在 D_{real}上训练的模型相当的性能，还要求其权重与在 D_{real}上训练的模型的权重具有相似的收敛结果。令 $\theta_{D_{\text{syn}}}$ 和 $\theta_{D_{\text{real}}}$ 分别表示通过在 D_{syn} 和 D_{real}上训练得到的权重，因此目标定义为

$$\begin{aligned} D^*_{\text{syn}} &= \arg\min_{D_{\text{syn}}} \kappa(\theta_{D_{\text{syn}}}, \theta_{D_{\text{real}}}), \\ \text{s.t. } \theta_{D_{\text{syn}}} &= \arg\min_{\theta} L(D_{\text{syn}}, \theta), \end{aligned} \tag{2.3}$$

其中 $\kappa(\cdot,\cdot)$是一个定义为如下形式的距离函数

$$\begin{aligned} \kappa(\theta_{D_{\text{syn}}}, \theta_{D_{\text{real}}}) = \sum_{i=1}^{H} \iota\Big(& \nabla_{\theta^i_{D_{\text{syn}}}} L(D_{\text{syn}}, \theta_{D_{\text{syn}}}), \\ & \nabla_{\theta^i_{D_{\text{real}}}} L(D_{\text{real}}, \theta_{D_{\text{real}}}) \Big); \end{aligned} \tag{2.4}$$

H 表示层数，而 $\iota(\,,\,)$是诸如欧几里得距离之类的距离度量。D_{syn} 通过优化进行更新

$$D_{\text{syn}} \leftarrow D_{\text{syn}} - \eta \nabla_{D_{\text{syn}}} \kappa(\theta_{D_{\text{syn}}}, \theta_{D_{\text{real}}}) \tag{2.5}$$

轨迹匹配数据集蒸馏。当前的数据集蒸馏方法 [8, 15] 采用轨迹匹配来进行无损蒸馏。权重轨迹匹配的目标是

数据集蒸馏是使在 D_{syn} 上训练的模型的梯度下降轨迹与在 D_{real}上训练的模型的梯度下降轨迹相似，其定义如下：

$$D^*_{\text{syn}} = \arg\min_{D_{\text{syn}}} \frac{\|\hat{\theta}_{t+N} - \theta^*_{t+M}\|_2^2}{\|\theta^*_t - \theta^*_{t+M}\|}, \tag{2.6}$$

其中 θ^*_t表示在第t步使用 D_{real} 训练的模型权重； M和 N是两个预定义的超参数。$\hat{}\ \theta_t$ 是第 t步使用 D_{syn} 训练的模型权重，其获取方式如下：

$$\hat{\theta}_{t+i+1} = \hat{\theta}_{t+i} - \eta \nabla l(\hat{\theta}_{t+i}, D_{\text{syn}}), \text{ where } \hat{\theta}_t := \theta^*_t. \tag{2.7}$$

2.2 信息泄露攻击

架构推断攻击。 如先前研究所示 [12, 16, 17], , 在架构推断攻击（AIA）中，攻击者旨在推断目标模型 f的模型架构。训练过程中使用的模型架构乃至其他超参数，都可以利用侧信道信息进行推断，例如训练损失图和功耗侧信道。假设存在一个模型架构集合 $\mathbf{F}$，其大小表示为$l = |\mathbf{F}|$，攻击者的目标是确定 f使用的是哪种架构。将该攻击中获取的侧信道信息视为输入 $x \in \mathbb{R}^n$，攻击者试图寻找一个映射$A_A: \ \mathbb{R}^n \mapsto \mathbb{R}^l$，该映射即为推断模型。与 MIA 类似，训练一个可靠的 A_A 对于 AIA 至关重要。

成员推断攻击。 在成员推断攻击（MIA）中， [18], 拥有数据集的受害者 D 会在 D上训练一个分类器模型 f。攻击者的目标是确定样本$\mathbf{x}$ 是否属于 D，并访问 $f(\mathbf{x};\ \theta)$。攻击者可以拥有白盒访问权限或黑盒访问权限。黑盒访问权限又可进一步分为三个子类：完整置信度分数、Top-K 置信度分数和仅标签。MIA 中的一种常见方法 [10, 19] 是，在与目标分类器 $f(\mathbf{x};\ \theta)$ 相同（或相似）的分布上训练本地模型 h。随后，攻击者基于 h训练攻击模型 A_M，并利用其输出来改进 A。尤其是，构建有效的A_M在其中发挥着至关重要的作用。

模型反演攻击。 模型反演（MIV）是指部分或完全重建目标模型的训练数据集 [13, 14]。这可以表述为一个优化问题，旨在寻找能够最大化目标模型似然的敏感特征值。给定训练数据集 $D_{\text{train}} = \{(x^1,\ y^1)$ $.....(x^n,\ y^n)\}$以及一个在 D_{train}上训练的模型 $f(x;\ \theta)$，攻击者试图构建一个反演数据集 $D_{\text{inv}} = \{(\ \tilde{x}^1,\ \tilde{y}^1).....(\ \tilde{x}^m,\ \tilde{y}^m)\}$，$m \le n$，使得 $\forall i \in [1, m],\ \exists j \in [1, n],\ \|\tilde{x}^i - x^j\|^2 < \rho$，其中$\rho \in \mathbb{R}$是预先定义的误差预算。而且， $\forall i \in [1, m],\ f(\ \tilde{x}^i) = \tilde{y}^i$。

2.3 去噪扩散概率模型

去噪扩散概率模型（DDPMs） [20]是一类生成模型，受热力学扩散启发，通过迭代去噪过程学习生成数据。在前向过程中，干净图像 x_0 会逐渐加入随机高斯噪声，直至达到最大步数 T。这一前向过程由转移方程描述：

$$q(x_t|x_0) := \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I}), \tag{2.8}$$

其中 $\alpha_t := 1-\beta_t$； $\bar{\alpha}_t := \prod_{i=1}^{t}\alpha_i$； $\beta_t \in (0,1]$ 为预定义常数。前向过程可进一步简化为

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \tag{2.9}$$

其中 $\epsilon \in \mathbf{N}(\mathbf{0}, \mathbf{I})$。因此，可以轻松得到 x_t，其中有 x_0 和 $t \in [1,T]$。

在反向过程中，x_T 会按照反向转移方程逐步去噪，变为 x_0

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)). \tag{2.10}$$

训练去噪扩散概率模型时使用的损失函数可简化为

$$L_t := \frac{1}{2\beta_t^2}\|\tilde{\mu}_t(x_t,x_0,t) - \mu_\theta(x_t,t)\|^2, \tag{2.11}$$

其中

$$\tilde{\mu}_t(x_t,x_0,t) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t. \tag{2.12}$$

3 针对数据集蒸馏的信息泄露攻击

3.1 威胁模型

攻击者的目标。受害者 V 使用真实数据集 D_{real}、在 D_{real}上训练的受害者模型 f:$\mathbb{R}^M \mapsto \mathbb{R}^C$，以及某种数据集蒸馏算法 $\gamma \in \Gamma$，合成一个合成数据集 D_{syn}。这里，M、C和 Γ 分别表示展平后输入图像的维度、类别数，以及数据集蒸馏算法集合。攻击者 A 在能够访问 D_{syn}的情况下，旨在从 D_{real}中泄露敏感信息。敏感信息分为三类：架构信息、成员身份和 D_{real}中的训练样本。

为了揭示数据集蒸馏算法 γ 以及蒸馏过程中使用的 f的模型架构，A 需要结合所有可能的侧信道信息使用 AIA，以推断 γ 所用的算法（例如数据集蒸馏 [1] 或数据集凝缩 [2], ）以及 f的架构（例如 AlexNet）；为了揭示成员身份信息，A 需要发起成员推断攻击（MIA），以确定任意数据样本 $x \in D_{\text{real}}$是否属于成员；揭示训练样本则是要恢复一个数据集 D_{inv}，使得 $\forall x \in D_{\text{inv}}$，$x \in D_{\text{real}}$。具体而言，为了确定给定样本 $x \in D_{\text{inv}}$ 是否属于 D_{real}，我们遵循先前的研究 [13, 14]。如果受害者模型将带有目标标签的恢复样本分类为目标类别，则该恢复样本被视为成员。

攻击者能力。鉴于合成数据集 D_{syn} 由 V公开发布，A 可以完全访问 D_{syn}。A 可以访问公开可用数据集，收集辅助数据集 D_{aux}，其分布与 D_{real}相似。A还了解所有现有的模型架构和训练算法。因此，A 可以使用 D_{syn} 训练任意数量的本地模型 $h_i(\cdot;\tau_i)$，其中 i为索引，并访问任意层的权重或输出。A还能够记录训练过程中的中间信息。

攻击者面临的挑战。A 无法访问 D_{real}。尽管 A 可以访问所有公开可用的架构、算法和蒸馏方法，但 A并不知道 V 使用的模型架构、训练算法和蒸馏方法。首先，A 需要建立一个可靠的二元分类器 A_M，即成员推断攻击模型，以判断某个样本是否属于成员；其次，A 必须生成充足且有效的损失轨迹，以支持架构推断攻击模型 A_M的训练；最后，A 必须训练一个生成网络，使其能够在给定目标类别时生成可被受害者模型分类为该目标类别的样本。这些样本还应体现真实数据集中的感知特征，并且看起来自然。

攻击场景。我们设想 V 是数据所有者，他们使用真实训练数据集 D_{real}，通过数据集蒸馏算法 γ训练受害者模型 f，从而生成合成数据集 D_{syn}。随后，V会将 D_{syn} 免费发布到任意公开在线数据集平台，例如 Kaggle[1] 和 Hugging Face[2], ，也可以将其发布到任意市场用于商业用途。A 可以免费下载 D_{syn} 或购买该数据集，然后发起上述攻击。

3.2 信息泄露攻击的初始化

获得合成数据集 D_{syn}, A 后，A 便开始攻击的第一步，即训练一个本地模型 h： $\mathbb{R}^M \mapsto \mathbb{R}^C$。超参数、模型架构和训练算法有多种选择。$A$ 甚至可以训练多个本地模型组成集成。在这里，我们考虑最简单的方式，即在一种主流架构（如 ResNet18）上训练一个本地模型。默认情况下，A 使用交叉熵损失训练 h

$$L_{\text{CE}}(y,h(x)) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\log(h(x)_{i,j}), \tag{3.1}$$

其中，N是批量大小；y是真实标签。A 使用 SGD优化器优化权重。在训练过程中，A会在每个训练轮次结束时记录损失和模型权重，以创建本地模型的损失轨迹 $T^L = \{L_1, \ldots L_E\}$。

接下来，A 收集一组模型架构 $\mathbf{F} = \{f_1, \ldots\ldots, f_v\}$，其中 $v = |\mathbf{F}|$。给定辅助数据集 D_{aux}，A 使用 $\gamma_i \in \Gamma$ 和 $f_j \in \mathbf{F}$ 合成 l个合成数据集，记为 $D^{i,\,j\,k,}_{\text{syn}}$，其中 i是 $\gamma_i \in \Gamma$的索引；j表示 $f_j \in \mathbf{F}$ 的索引；$k \in [1,l]$是特定合成数据集的索引。然后，对于每个 $D^{i,\,j\,k,}_{\text{syn}}$，$A$ 在 $D^{i,\,j\,k,}_{\text{syn}}$ 上训练一个相同架构的模型，训练 E 个训练轮次，并记录损失轨迹 $\mathbf{T}^{i,\,j,\,k} \in \mathbb{R}^E$。最终，$A$ 获得一个集合 $\mathbf{T}$，其中包含由 $\gamma_i \in \Gamma$和 $f_j \in \mathbf{F}$ 的所有组合生成的不同合成数据集的损失轨迹。

3.3 架构推断阶段

以匹配梯度下降轨迹或数据集分布为目标，先前研究提出了不同的方法来合成合成数据集 D_{syn}。这导致本地模型训练过程中出现各种损失轨迹模式。其中，当一个本地模型 h 在由针对某个真实数据集 D_{real}的特定蒸馏算法生成的 D_{syn} 上训练时，h 在训练过程中的损失轨迹是独特的。也就是说，当每个 h_i 都在各自的 D^i_{syn}上进行训练时，

[1]https://www.kaggle.com/datasets/amarhaiqal/aspen-hysys-distillation-column-data[2]https://huggingface.co/datasets/lemon-mint/bge_蒸馏_训练_数据集_v1

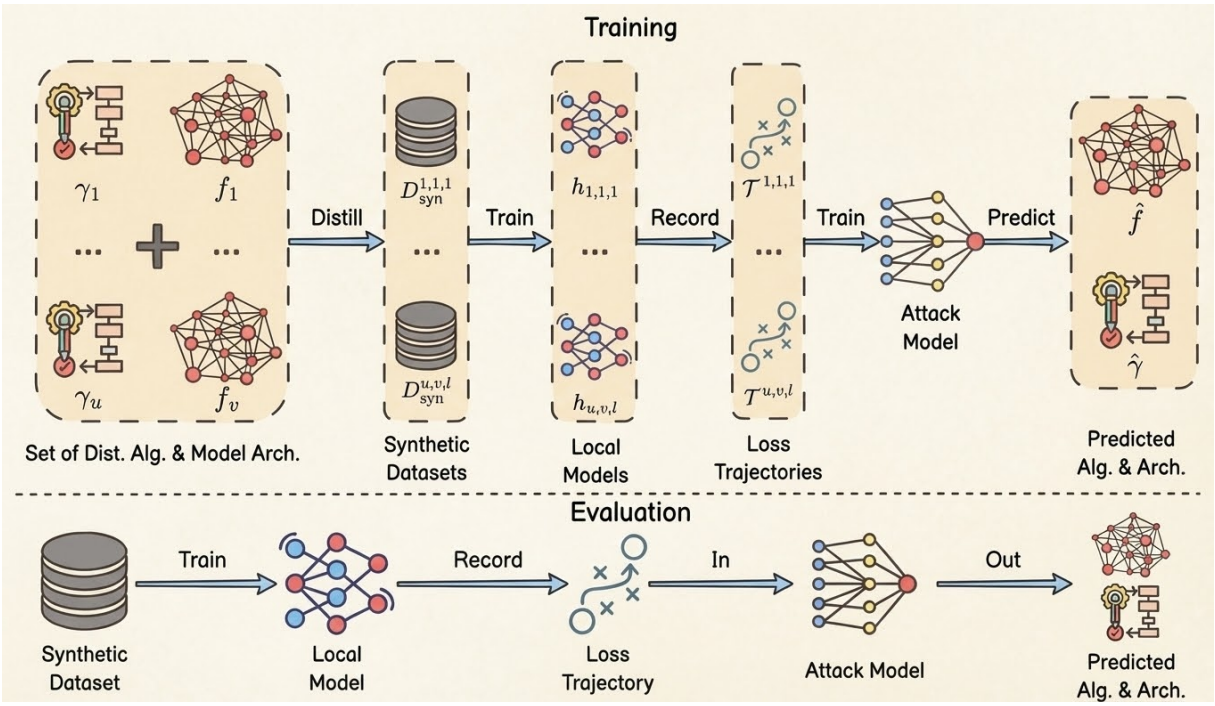


图 2：架构推断阶段。对于 AIA，A 将不同的蒸馏算法 γ_i 与模型架构 f_j 相结合，以生成 l 合成数据集。然后，分别使用这些合成数据集训练 $u \times v \times l$ 本地模型，从而生成损失轨迹集合 $\mathcal{T}$。随后使用 $\mathcal{T}$ 训练架构攻击模型 A_A，以预测数据集蒸馏采用了哪种算法和架构。

这些 D_{real} 由同一算法蒸馏而来，但采用了不同的蒸馏算法，其损失轨迹 TS 可以区分。类似地，对于固定的蒸馏算法，每个 D^i_{syn} 都是使用特定模型架构的模型 f_i 和相同的 D_{real} 合成的。当各个本地模型 h_iS 分别在各自的 D^i_{syn} 上训练时，它们的损失轨迹 TS 也可以区分。

基于这一发现，我们设计了图 2 所示的架构推断攻 击。在初始化步骤生成损失轨迹集合 T 后，对于每个 d_i 和 f_j，A 有 l 条损失轨迹 $\mathrm{T}^{i,\ j}$。这使得 A 能够训练架构推断攻击模型 A_A： $\mathbb{R}^E \mapsto \mathbb{R}^{u\times v}$。 A 可以使用 SGD 优化器和 L_{CE}（如公式 3.1 所示）来训练 A_A，使得给定损失轨迹 $T^L \in \mathbb{R}^E$ 时， $A_A(T^L) \in \mathbb{R}^{u\times v}$ 是对 V 在数据集蒸馏过程中所使用的蒸馏算法和模型架构的预测。攻击过程如算法 1 所示。

在获得训练好的 A_A 后， A 将准备好的预训练本地模型损失轨迹 T^L 输入 A_A，从而得到预测的蒸馏算法 $\hat{}\gamma$ 和模型架构 $\hat{}f$。有了 D_{syn} 和 $\hat{}f$， A 现在可以训练一个本地模型 h，该模型不仅与受害者模型 f 具有相同的架构，而且权重和性能也相似。这表明**攻击者完成了将黑盒转变为白盒**，并已准备好进入下一阶段。

为进一步探究架构推断攻击的机制，我们进行了理论分析，讨论了合成数据集与其对应的损失轨迹之间的关系，这些轨迹是在本地模型的训练过程中生成的。当使用相同的数据集 [21]训练时，同一模型架构的不同神经网络中的权重往往会收敛到一阶驻点。我们知道，使用相同算法和模型架构生成的合成数据集彼此相似，而使用不同算法或模型架构生成的合成数据集则彼此不同。因此，我们希望进一步证明，当两个模型分别使用两个相似的数据集进行优化时，具有相同模型架构的两个模型的训练损失往往也彼此相似。

以下定理表明，当两个相同架构的神经网络分别在两个相似数据集上训练时，每个训练轮次的权重距离可以根据训练轮次数、学习率以及两个数据集之间的距离进行估算。因此，只要数据集之间的距离足够小，这两个网络在每个训练轮次产生的损失就会彼此接近。

定理 3.1。假设数据集 $\mathcal{D}_2$ 是另一个数据集 $\mathcal{D}_1$,的扰动版本，即

$$\mathcal{D}_2 = \{(\boldsymbol{x}_n + \boldsymbol{h}_n, y_n) : \|\boldsymbol{h}_n\|_2 < \delta, n = 1, 2, \ldots, N\}, \tag{3.2}$$

其中 $\delta > 0$ 是一个常数。我们将 $\mathcal{D}_2$ 上的训练过程描述为

$$\begin{aligned} &\boldsymbol{\theta}_2(0) = \boldsymbol{\theta}_0, \\ &\boldsymbol{\theta}_2(t+1) = \boldsymbol{\theta}_2(t) - \eta(t)\nabla\mathcal{L}_{\mathcal{D}_2}(\boldsymbol{\theta}_2(t)), \end{aligned} \tag{3.3}$$

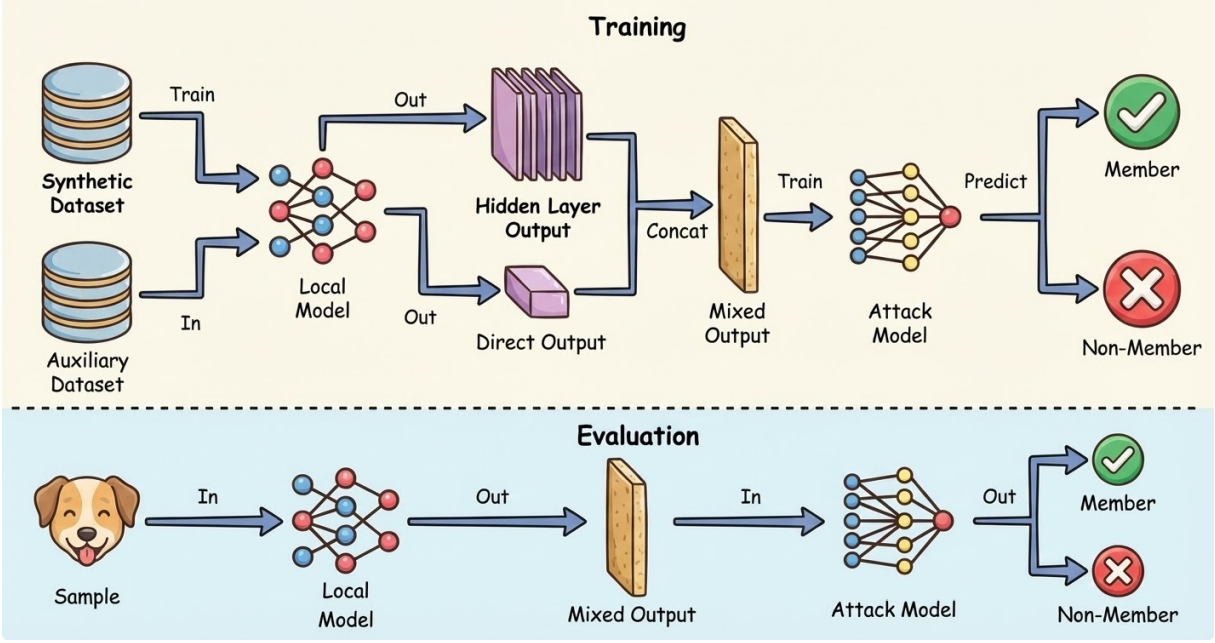


图3：成员身份推断阶段。在成员推断攻击中，A 利用本地模型每一层的输出 A_M 训练成员推断攻击模型，以确定本地模型 h 的层是否包含任意给定样本 x。

其中 $\theta_i(t)$ 表示模型 i 在第 t 步的权重；$\eta(t)$ 是第 t 步的学习率；$L_{\mathcal{D}_i}(\cdot)$ 是模型在 $\mathcal{D}_i$ 上训练得到的损失。

对于任意 $\varepsilon > 0$，有

$$\|\theta_1(t) - \theta_2(t)\| < \varepsilon,\ t \in \{1, 2, \ldots, T_0\}, \tag{3.4}$$

当 δ 足够小时。

由此可得以下推论：

Corollary 3.1. *For any $\varepsilon > 0$, one has*

$$|\mathcal{L}_{\mathcal{D}_1}(\theta_1(t)) - \mathcal{L}_{\mathcal{D}_2}(\theta_2(t))| < \varepsilon,\ t \in \{1, 2, \ldots, T_0\}, \tag{3.5}$$

当 δ 足够小时。

我们在附录 A 中证明定理 3.1 和推论 3.1。

根据推论 3.1，当本地模型在两个相似的数据集上训练时，往往会产生相似的损失轨迹。因此，这解释了攻击者为何能够利用损失轨迹训练分类器网络，以识别受害者模型所使用的数据集蒸馏方法和模型架构。

3.4 成员推断阶段

如图3所示，借助训练充分的本地模型 $h(\cdot\ \tau)$，A 开始训练成员推断攻击模型 A_M。为了最大限度地利用 D_{syn} 中保留的信息，A_M 的输入不仅包括输出 $h(x)$，还包括 h 中每一层的中间输出。这些输出来自 h 中的各层，记为 $h_o(x) = \{h^1(x)\ h^2(x) \ldots, h(x)\}$。$h^i(x)$ 是 h 中第 i 层的输出，而 $h(x)$ 表示 h 中最终层的输出。在 A_M 的训练过程中，A 从成员集和非成员集中随机抽取样本，从而组成一个大小为 N。如果样本 x_i 属于成员集，则将其与标签 $b_i = 1$ 配对；否则与 $b_i = 0$ 配对。A 随后使用二元交叉熵损失训练 A_M

$$L_{\text{BCE}}(b, A_M(h_o(x))) = -\frac{1}{N}\sum_{i=1}^{N} b_i \cdot \log(A_M(h_o(x))) + (1 - b_i) \cdot \log(1 - A_M(h_o(x))) \tag{3.6}$$

在评估中，如果 $A_M(h_o(x)) < 0.5$，我们就将 x 视为非成员样本；当 $A_M(h_o(x)) > 0.5$ 时，则判定 x 为成员。攻击过程如算法 2 所示。

3.5 模型反演阶段

在这种模型反演攻击中，我们使用扩散模型作为生成器网络。使用扩散模型发动此攻击的主要挑战在于，向扩散模型的损失函数中添加额外约束无法取得理想结果。原因是，去噪扩散概率模型输出的是预测噪声，而非预测的干净图像。因此，这些约束无法将预测噪声与原始干净图像配对。为解决这一问题，我们采用并改进了双网络框架[22]，其中两个网络 ϕ 和 ψ 共同构成生成网络，如图4所示。ϕ 用于预测噪声 ϵ，而 ψ 用于预测干净图像 x_0。因此，这使得 A 能够向 ψ 的损失函数中添加约束。

在训练过程中，给定通过方程 2.9 生成的带噪图像 x_t、预定义标签 c 以及均匀采样的时间步 t，ϕ 将它们作为输入，并预测噪声 $\epsilon_\phi(x_t,\ t,\ c)$。$c$ 被设为 $\emptyset$，以便以概率 p^u 执行无条件采样。

算法 1： 架构推断阶段

数据： D_{syn}：一个合成数据集；
结果： $\hat{i}$，$\hat{j}$：蒸馏算法和模型架构的索引；
/* 训练攻击模型 */
轨迹集 $\leftarrow \emptyset$；
/* 对于每种蒸馏算法。 */
为 $\gamma_i \in \Gamma$ **执行**
  /* 对于每种模型架构 */
  对于 $f_j \in \mathcal{F}$ **执行**。
    /* 生成 l 损失轨迹样本。 */
    对 $k \in [1, l]$ **执行**
      合成一个 $D^{i,j,k}_{\text{syn}}$，训练一个本地模型并记录其损失轨迹 $\mathbf{T}^{ijk}$；
      将 $(\mathbf{T}^{ijk})$ 添加到轨迹集；
    end
  end
end
当未收敛时 **执行**
  使用轨迹集训练 A_A；
end
/* 评估 */
使用 D_{syn} 训练本地模型，并记录其损失轨迹 $\mathcal{T}$；
$\hat{i}$，$\hat{j} \leftarrow A_A(\mathbf{T})$； /* 使.....的预测 $\hat{i}$ 蒸馏算法和 $\hat{j}$ 模型架构 */
返回 $\hat{i}, \hat{j}$；

算法 2： 成员推断阶段

数据： X：任意数据样本；h：训练好的本地模型
结果： 0/1：非成员/成员；
/* 训练攻击模型 */
训练集 $\leftarrow \emptyset$；
对 $(x, b) \in D_{aux}$ **执行**
  将 $(h_o(x)\ b)$ 加入训练集；
end
当尚未收敛时 **执行**
  使用训练集训练 A_M；
end
/* 评估 */
$X \sim P(D)$； /* 采样 */
如果 $A_M(X) > 0.5$ **那么**
  返回 1； /* 成员 */
else
  返回 0； /* 非成员 */
end

因此，x_0 的估计可推导如下，

$$x_0^* = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\phi(x_t, t, c)). \tag{3.7}$$

算法3： 针对数据集蒸馏的模型反演攻击tion

数据： D_{syn}：合成数据集；
结果： x_0：恢复的训练样本；
/* 训练 */
当 L 未收敛 **执行**
  $x_t, c \sim D_{\text{aux}}$； /* 采样 */
  $x_t \leftarrow \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$； /* 前向过程 */
  $\hat{\epsilon} \leftarrow \phi(x_t, t, c)$； /* 噪声预测 */
  $x_0^* \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon})$； /* 反向过程 */
  $\hat{x}_0, r_t \leftarrow \psi(x_0^*, t, c)$； /* 直接输出 */
  $\tilde{\mu}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t$； /* 估计 */
  $\mu_{\phi,\psi}(x_t, t, c) = r_t \cdot \hat{x}_0 + (1-r_t)\cdot x_0^*$； /* 联合输出 */
  $L \leftarrow \lambda_1 \cdot L_{\text{MSE}}(\epsilon, \hat{\epsilon}) + \lambda_2 \cdot L_{\text{MSE}}(x_0, \hat{x}_0) + \lambda_3 \cdot L_{\text{MSE}}(\tilde{\mu}_t, \mu_{\phi,\psi}(x_t, t, c)) + \lambda_4 \cdot L_{\text{cls}} + \lambda_5 \cdot L_{\text{traj}}$； /* 计算损失 */
  更新 ϕ，ψ；
end
/* 评估 */
$x_T \leftarrow \mathrm{N}(\mathbf{0}, \mathbf{I})$； /* 初始化噪声 */
$c \leftarrow \text{random}(0, C)$； /* 分配类别 */
对于 $t, \{T, \ldots\ldots, 1\}$ **执行**
  $\hat{\epsilon} = \phi(x_t, t, c)$；
  $\mu^S_{\hat{\epsilon}} = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} + \sqrt{1-\bar{\alpha}_{t-1}}\cdot\hat{\epsilon}$；
  $x_0^* \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\phi(x_t, t, c))$；
  $\hat{x}_0, r_t \leftarrow \psi(x_0^*, t, c)$；
  $\mu^S_{\hat{x}_0} = \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{x_t - \sqrt{\bar{\alpha}_t}\hat{x}_0}{\sqrt{1-\bar{\alpha}_t}}$；
  $x_{t-1} \leftarrow r_t \cdot \mu^S_{\hat{x}_0} + (1-r_t)\cdot\mu^S_{\hat{\epsilon}}$；
end
返回 x_0

基于 $\epsilon_\phi(x_t, t, c)$，A 获得了 $p_\phi(x_{t-1}|x_t)$ 的均值估计 $\mu_\phi(x_t, t, c)$，记作

$$\mu_\phi(x_t, t, c) = \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon_\phi(x_t, t, c) \tag{3.8}$$

ψ 随后输入 x_0^*、t 和 c，以输出 $\hat{x}_0$ 和动态值 r_t，记为 $(\hat{x}_0, r_t) = \psi(x_0^*, t, c)$。$\hat{x}_0 \in \mathbb{R}^M$ 是预测的干净图像；$r_t \in \mathbb{R}^{\frac{M}{3}}$ 用于平衡 ϕ 和 ψ 对最终输出 $\mu_{\phi,\psi}(x_t, t)$ 的影响。$\frac{M}{3}$ 表示 r_t 可以视为具有与 $\hat{x}_0$ 相同宽度和高度、但仅有 1 个通道的特征图。然后，A 可以基于 $\hat{x}_0$ 进一步估计 $\mu_\phi(x_t, t, c)$，记为

$$\mu_{\hat{x}_0}(x_0^*, t) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t. \tag{3.9}$$

最终，$\mu_{\phi,\psi}(x_t, t, c)$ 的最终估计由以下内容得出

$$\mu_{\phi,\psi}(x_t, t, c) = r_t \cdot \mu_{\hat{x}_0}(x_t, t, c) + (1-r_t)\cdot\mu_{\hat{\epsilon}}(x_t, t, c) \tag{3.10}$$

为了训练 ϕ 和 ψ，A 最小化损失

$$\begin{aligned} L = {} & \lambda_1 \cdot L_{\text{MSE}}(\epsilon, \epsilon_\phi(x_t, t, c)) + \lambda_2 \cdot L_{\text{MSE}}(x_0, \hat{x}_0) \\ & + \lambda_3 \cdot L_{\text{MSE}}(\tilde{\mu}_t, \mu_{\phi,\psi}(x_t, t, c)) + \lambda_4 \cdot L_{\text{cls}} + \lambda_5 \cdot L_{\text{traj}}. \end{aligned} \tag{3.11}$$

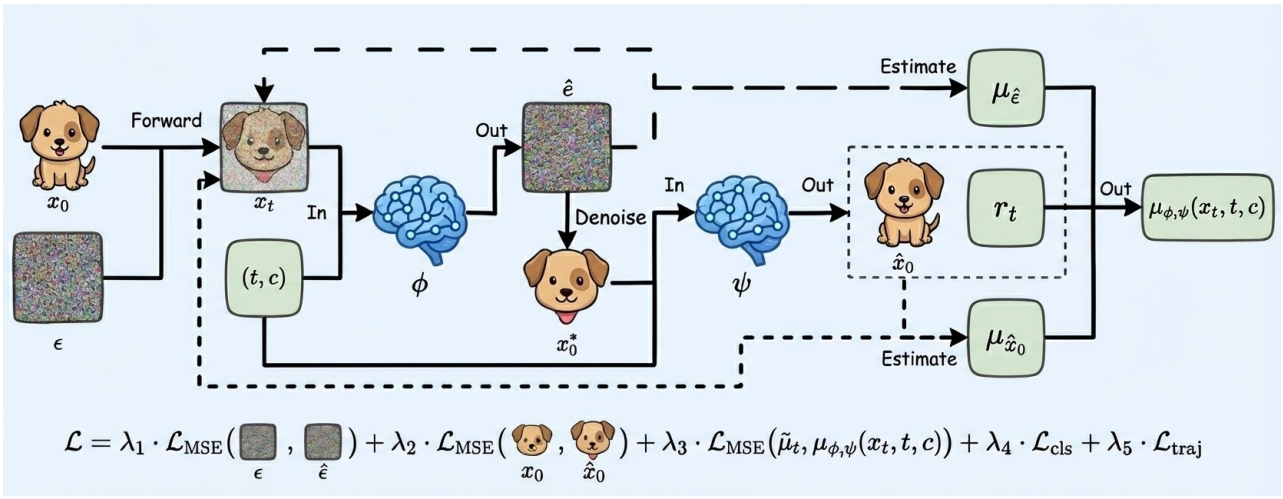


图4：模型反演阶段。该框架采用两个模型 ϕ 和 ψ。 ϕ 以去噪扩散概率模型的方式进行训练，以预测噪声 ϵ，而 ψ 则以去噪图像 x_0^* 作为输入进行训练，以输出干净图像 x_0 和系数 r_t。该双网络框架的最终输出是使用 r_t 对其输出进行加权后的组合。值得注意的是，除第一项外，损失函数中的其余项均用于正则化 ψ。

这里， λ_i是权重参数；第一项使 ϕ 更准确地预测ϵ；第二项促使 ψ 更精确地预测 x_0；第三项是一个约束，使 ψ输出更合理的 r_t，其中 $\tilde{\mu}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$； L_{MSE}表示均方误差损失：

$$L_{\text{MSE}}(X, Y) = \frac{1}{n}\sum_{i}^{n}(x_i - y_i)^2, \forall x_i \in X, y_i \in Y; \tag{3.12}$$

第四项 L_{cls} 是如下定义的分类

$$L_{\text{cls}} = L_{CE}(c, h(\hat{x}_0)). \tag{3.13}$$

这要求生成的$\hat{x}_0$ 使 h 产生预定义标签c，从而使扩散模型能够更好地学习 h 所训练的训练数据集的分布；第五项 L_{traj} 是轨迹损失，定义为

$$L_{\text{traj}}(\hat{x}_0, c, h, i) = \frac{\|\eta \nabla L_{\text{CE}}(c, h(\hat{x}_0; \tau_i^*))\|^2}{\|\tau_{i+1}^* - \tau_i^*\|^2} \tag{3.14}$$

此处， τ_i^*表示在 D_{syn}上的训练过程中于第 i 个训练轮次得到的 h的展平权重向量。简而言之，通过最小化L_{traj}， A 旨在使G生成的合成样本让 h 产生与 D_{syn}相同的损失轨迹。值得注意的是，除第一项外，其他项不用于更新 ϕ中的权重，而仅用于更新 ψ中的权重。

在采样时， A 按如下方式使用去噪扩散隐式模型（DDIM）[23]：

$$\mu_{\phi,\psi} = r_t \cdot \mu_{\hat{x}_0}^S + (1 - r_t) \cdot \mu_{\hat{\epsilon}}^S, \tag{3.15}$$

其中

$$\mu_{\hat{x}_0}^S = \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}} \cdot \frac{x_t - \sqrt{\bar{\alpha}_t}\hat{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \tag{3.16}$$

and

$$\mu_{\hat{\epsilon}}^S = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\alpha_t}} + \sqrt{1-\bar{\alpha}_{t-1}} \cdot \hat{\epsilon}. \tag{3.17}$$

此处省略了 σ_t，因为在去噪扩散隐式模型中， $\sigma_t = 0$， $\forall\, t$。攻击过程如算法3所示。

4 实验

4.1 设置

数据集。在本实验中，我们使用 CIFAR-10、CIFAR-100、CIFAR-100-Ext、CINIC-10、TinyImageNet-200 和 ImageNet。

- **CIFAR-10**[3]：50，000 个训练集样本；10，000 个测试集样本；每个样本包含 3×32×32 个像素；共 10 个类别。
- **CINIC-10**[4]：90，000 个训练集样本；90，000 个测试集样本；每个样本包含 3×32×32 个像素；共 10 个类别；它用作 CIFAR-10 数据集的辅助数据集。
- **CIFAR-100**[3]：训练集包含 50，000 个样本；测试集包含 10，000 个样本；每个样本包含 3 × 32 × 32 个像素；共 100 个类别。
- **CIFAR-100-Ext**[5]：训练集包含 50，000 个样本；测试集包含 10，000 个样本；每个样本包含 3 × 32 × 32 个像素；共 10 个类别；它用作 CIFAR-100 数据集的辅助数据集。
- **TinyImageNet-200**[6]：训练集包含 100，000 个样本；验证集包含 10，000 个样本；测试集包含 10，000 个样本；每个样本包含 3×64×64 个像素；共 200 个类别。
- **ImageNet**[7]：训练集和验证集中共有 14，197，122 张标注图像；使用前会将样本裁剪为 3×64×64 像素；总计 1，000 个类别；该数据集用作 Tiny Imagenet 数据集的辅助数据集。

[3]https://www.cs.toronto.edu/ kriz/cifar.html [4] https://github.com/BayesWatch/cinic-[5] https://www.kaggle.com/datasets/dunky11/cifar100-100x100-images-extension [6] https://www.kaggle.com/c/tiny-imagenet [7]https://image-net.org/

数据集蒸馏算法。 以下方法用作生成合成数据集的基线。

• **MTT**[24]：MTT优化合成数据集，使在合成数据集上训练的模型被引导至与在真实数据集上训练的模型相似的状态。这是通过添加一个约束来实现的，该约束缩小在合成数据集与真实数据集上训练的模型的损失轨迹之间的距离。

• **FTD**[7]：FTD优化合成数据集，使其为模型权重的下降提供平坦轨迹。通过向教师模型引入随机噪声，FTD有意对教师模型的权重进行扰动，从而形成平坦轨迹。其目标是使合成数据集对权重扰动具有鲁棒性。

• **DATM**[8]：DATM强调，被选择用于匹配的轨迹所处的训练阶段对合成数据集具有显著效果。早期轨迹对低基数数据集更为重要，而后期轨迹对大型数据集更为主导。因此，DATM使生成模式的难度与合成数据集的规模相一致。

• **SelMatch**[9]：SelMatch 在轨迹匹配数据集蒸馏过程中采用基于选择的初始化与部分更新。该方法根据合成数据集的规模管理其难度水平。

• **SeqMatch**[15]：SeqMatch 按顺序生成合成样本，以避免蒸馏数据集无法提取神经网络在后续训练轮次中学到的高级特征。当大量合成样本同时被优化时，就会出现这一问题。通过自适应地优化合成数据集，SeqMatch 有效解决了该问题。

评估指标。 对于在 D_{syn} 上训练的模型，我们使用 Top-1 准确率（Top-1 Acc.）来评估其性能。平衡准确率（BA）、低误报率下的真阳性率（T@LF）、曲线下面积（AUC）以及完整对数尺度 ROC [25–27] 用于评估成员推断攻击。对于属性推断攻击模型，我们使用 Top-1 Acc. 来衡量其有效性。最后，攻击准确率（Atk. Acc.）和 K近邻距离（KNN Dist.）[13, 14, 28, 29] 用于评估 MIV攻击。下文将详细说明各项指标。

• **Top-n 准确率** 是预测类别落在置信度得分最高的 n 个类别中的预测数量与预测总数之比。

• **平衡准确率** 是模型在由成员和非成员组成的平衡数据集上做出的正确预测数量与预测总数之比。

• **低误报率下的真阳性率** 是单个较低假正率对应的真正率，可概述攻击在少量样本上的攻击性能。在本次评估中，低误报率下的真阳性率设为假正率为 0。1% 时的真正率。

• **曲线下面积**是受试者工作特征（ROC）曲线下的面积，该曲线由成员身份推断攻击在所有可能阈值下的真正率和假正率绘制而成。

• **完整对数尺度 ROC** 以对数尺度绘制受试者工作特征曲线，从而强调低假阳性率区域中的真正率区域。这能更全面地展现攻击性能。

• **攻击准确率** 定义为：评估模型将合成样本分类为目标类别的数量与恢复数据集中合成样本总数之比。评估模型在真实数据集上训练。

• **KNN 距离** 是恢复样本与特征空间中真实样本之间的最短距离。该指标用于衡量特征层面的相似性。恢复样本和真实样本的特征是上述评估模型倒数第二层的输出。对于在同一数据集上进行的实验，我们始终使用同一个经过充分训练的评估模型。

实现细节。 受害者模型采用 ConvNet、AlexNet、ResNet18 和 VGG11BN。成员身份推断攻击模型和架构推断攻击模型采用相似的三层全连接网络架构，但输入和输出维度不同。 ϕ和ψ均采用 UNet 架构。本地模型使用 SGD优化器训练 100 轮，学习率初始为 0.01，并每 25 轮降低 0.2；用于生成轨迹的卷积神经网络使用 SGD优化器训练 100 轮，学习率为 0.01；攻击模型统一使用 SGD优化器训练 100 轮，学习率初始为 0.01。学习率按余弦退火方式衰减，最终降至 0.0001；ϕ和 ψ使用 Adam 优化器训练 $10,000$ 轮，学习率恒定为 0.0001。 λ默认设为 1。生成具有不同每类图像数（IPC）的合成数据集，以评估合成样本数量对隐私泄露的影响。

4.2 评估

首先，将训练数据集随机划分为两个子集。第一个子集是真实数据集，包含 80% 的训练样本，用于数据集蒸馏。另一个子集由剩余的 20% 组成，作为成员样本纳入辅助数据集。这两个子集都按类别均衡。对于选出的五种数据集蒸馏方法，我们使用四种模型架构生成合成数据集。随后，利用这些合成数据集训练本地模型并生成损失轨迹。对于给定算法与架构构成的全部 20 种组合中的每个类别，我们生成 500 条损失轨迹。总计生成 $10，000$ 条轨迹，并按照 $4:1$ 的比例划分为类别均衡的训练集和测试集。对于本地模型，我们始终使用 ResNet18 作为模型架构。随后，使用本地模型、损失轨迹以及辅助数据集启动 IRA。

架构推断阶段。如表2、表5和表6所示，在包含20个类别的分类任务中，AIA 的 top-1测试准确率在大多数情况下都超过了 75%。这表明，由同一模型架构在不同方法生成的合成数据集上训练所产生的损失轨迹是可分离的。此外，不同

表2：在CIFAR10数据集上对最先进的数据集蒸馏算法进行的IRA评估。

方法	模型架构	测试集Top-1准确率 % IPC 100	250	500	1000	成员推断攻击 平衡准确率†/曲线下面积†/低误报率下的真阳性率 %† IPC 100	250	500	1000	属性推断攻击 Top-1 准确率 %† IPC 100	250	500	1000	MIV 攻击准确率↑/K近邻距离↓ IPC 100	250	500	1000
MTT	卷积神经网络	58.4	60.4	62.8	63.9	0.77/0.85/12.5	0.81/0.89/21.5	0.81/0.89/20.1	0.82/0.88/19.5					0.81/682.4	0.83/634.2	0.87/591.2	0.89/574.7
	AlexNet	56.7	58.2	60.5	61.3	0.71/0.78/4.6	0.72/0.80/5.9	0.80/0.88/20.2	0.80/0.88/20.2					0.79/653.7	0.82/613.2	0.86/588.2	0.86/572.8
	ResNet18	54.6	59.3	63.1	64.3	0.70/0.77/3.2	0.80/0.88/17.2	0.80/0.87/18.2	0.82/0.89/19.9					0.77/758.7	0.81/603.6	0.87/582.3	0.89/552.6
	VGG11-BN	57.2	60.7	63.5	65.9	0.77/0.85/11.2	0.79/0.87/14.3	0.81/0.88/16.7	0.89/0.95/53.6					0.81/633.7	0.84/617.8	0.86/599.2	0.89/542.8
FTD	卷积神经网络	68.8	72.4	73.7	75.4	0.80/0.88/20.7	0.82/0.89/22.0	0.89/0.95/51.0	0.90/0.95/46.4					0.90/551.3	0.91/542.5	0.91/537.9	0.92/512.4
	AlexNet	66.3	68.5	72.8	73.6	0.82/0.89/21.6	0.78/0.86/13.8	0.89/0.95/53.5	0.90/0.96/55.8					0.89/553.4	0.92/508.4	0.92/515.6	0.92/509.1
	ResNet18	67.5	72.1	74.9	77.8	0.80/0.88/16.2	0.82/0.89/25.0	0.90/0.96/57.9	0.89/0.95/54.1					0.89/546.8	0.90/565.8	0.93/502.7	0.93/494.5
	VGG11-BN	69.1	73.9	75.2	77.6	0.81/0.89/16.0	0.82/0.90/20.1	0.90/0.96/56.4	0.89/0.95/51.4					0.90/541.9	0.93/515.9	0.93/507.2	0.93/491.2
DATM	卷积神经网络	74.7	77.9	78.1	81.7	0.89/0.95/50.9	0.93/0.97/75.1	0.94/0.97/75.7	0.94/0.98/77.6					0.94/489.9	0.94/486.4	0.94/467.2	0.94/488.9
	AlexNet	71.4	74.6	77.2	79.2	0.90/0.96/51.0	0.89/0.95/49.5	0.90/0.96/53.3	0.93/0.97/71.0					0.90/569.2	0.93/506.2	0.94/468.3	0.94/476.1
	ResNet18	76.4	78.5	81.6	82.5	0.93/0.97/76.0	0.94/0.97/74.7	0.94/0.98/74.9	0.94/0.97/72.7					0.94/492.1	0.94/487.4	0.94/465.8	0.94/472.4
	VGG11-BN	75.2	78.1	80.9	82.2	0.90/0.95/53.7	0.93/0.97/76.3	0.93/0.98/77.1	0.93/0.97/73.5					0.93/511.0	0.94/489.2	0.94/487.5	0.94/492.4
SelMatch	卷积神经网络	78.1	80.2	81.0	82.6	0.88/0.94/53.1	0.91/0.96/66.7	0.92/0.96/67.1	0.93/0.97/73.7					0.94/489.1	0.94/467.3	0.94/477.6	0.94/481.5
	AlexNet	75.4	76.1	78.5	79.8	0.86/0.93/37.1	0.91/0.96/63.3	0.91/0.96/64.9	0.93/0.97/77.6					0.92/521.4	0.94/488.9	0.94/471.1	0.94/478.3
	ResNet18	78.2	82.5	85.3	83.2	0.88/0.95/45.7	0.92/0.97/68.3	0.92/0.97/69.1	0.94/0.98/74.8					0.94/468.2	0.94/477.0	0.94/467.9	0.95/457.2
	VGG11-BN	79.2	81.4	82.3	82.2	0.89/0.95/53.3	0.92/0.97/74.4	0.92/0.97/70.6	0.93/0.97/74.2					0.94/471.3	0.94/482.9	0.94/469.2	0.94/477.5
SeqMatch	卷积神经网络	68.4	69.2	72.3	74.2	0.79/0.87/17.1	0.81/0.89/25.6	0.82/0.89/18.0	0.85/0.92/29.6					0.90/552.6	0.90/562.7	0.91/547.9	0.91/533.2
	AlexNet	67.9	69.9	72.7	73.1	0.77/0.85/20.8	0.79/0.86/24.5	0.83/0.90/27.7	0.85/0.92/42.7					0.90/556.2	0.90/562.8	0.91/545.0	0.90/547.1
	ResNet18	70.5	73.3	75.6	77.9	0.83/0.90/32.5	0.83/0.88/22.0	0.84/0.91/30.7	0.84/0.92/42.9					0.90/571.2	0.91/567.4	0.91/542.8	0.92/52 6.7
	VGG11-BN	71.6	73.4	76.9	78.4	0.81/0.89/20.5	0.82/0.89/26.4	0.84/0.92/40.9	0.86/0.93/41.1					0.91/572.3	0.90/569.4	0.91/554.4	0.91/563.1

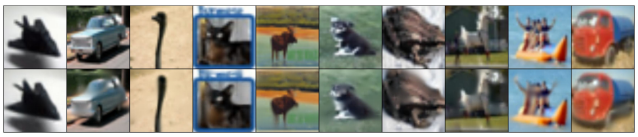


图5：CIFAR-10上的MIV定性结果。上行是真实数据集中每个类别的样本。下行是由扩散模型生成的合成样本。

在同一蒸馏过程中使用的模型架构也会产生可分离的损失轨迹。我们还发现，IPC 的变化对分类结果没有显著影响。

值得注意的是，通过不同方法生成的合成数据集会在优化相应目标函数的过程中自然包含梯度轨迹信息。例如，平坦轨迹是FTD [7],的目标，而一系列权重轨迹被选为DATM [8]中的优化目标。因此，这些不同方法所生成的合成数据集的损失轨迹是可分离的，因为这些方法具有内在不同的优化目标。对于同一方法在不同架构下的损失轨迹，由于蒸馏过程中使用的模型架构不同，权重轨迹信息也有所不同。类似现象也可在 [12],中发现，不同模型架构会呈现出攻击者可进一步利用的独特模式。

成员推断阶段。 成员推断攻击的评估结果列于表2、表5和表6中。完整对数尺度 ROC曲线如图6、图9和图10所示。首先，我们观察到，IPC 对本地模型的测试准确率有显著影响，而蒸馏过程中的模型架构选择相对而言影响较小。在这三个数据集的每项独立实验中，较高的 IPC 都能带来更高的测试准确率。

此外，我们还注意到，对于性能更好的本地模型，成员推断攻击的平衡准确率、曲线下面积和低误报率下的真阳性率往往更高。这表明高质量的合成数据集更有可能导致隐私泄露。例如，IPC为=1,000的SelMatch数据集算法与ResNet18模型架构能够生成高质量的合成数据集，使本地模型在CIFAR-10的测试数据集上达到83.2%的测试准确率。然而，该模型上的成员推断攻击性能也出人意料地高，其中平衡准确率= 0.94、曲线下面积=0.98和低误报率下的真阳性率= 74.8。

与其他最先进的成员推理攻击在不同场景下的表现相比，[25, 30, 31], 在这些场景中，攻击者对本地模型仅具有黑盒访问权限，而 IRA 的成员推断攻击评估结果却出奇地高。我们认为，以下事实导致了这些结果：受害者使用高保真度的数据集蒸馏算法合成合成数据集。由于蒸馏过程中固有的损失函数，这种蒸馏得到的数据集能够引导本地训练的模型形成与受害者直接使用真实数据集训练的模型相似的梯度下降轨迹。受害者将该合成数据集公开后，便向攻击者开放了对合成数据集的白盒访问权限。因此，攻击者能够训练出一个不仅性能相当，而且权重也与受害者模型相似的本地模型。换言之，**正是受害者间接将受害者模型交给了攻击者**。获得对本地模型的白盒访问权限后，攻击者便可利用其中的所有信息。本地模型每一层的输出现在都可用于训练攻击模型。这些因素共同促成了较高的攻击性能。

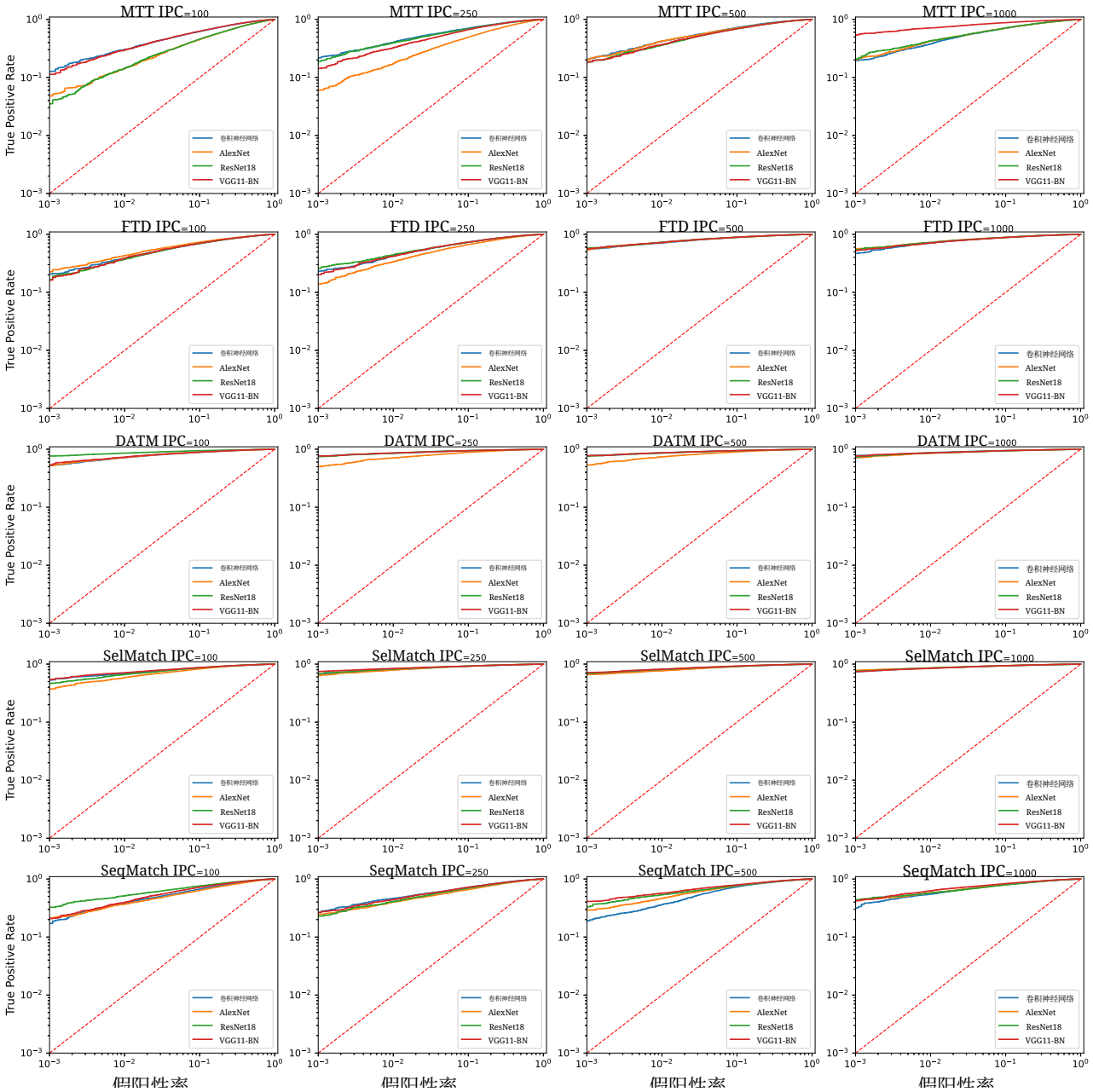


图6：合成CIFAR-10数据集上成员推断攻击的对数尺度ROC曲线。

MIV攻击。在表2、表5和表6中，我们列出了 MIV攻击的定量结果。我们观察到，对于同一个真实数据集，随着本地模型测试准确率的提高，攻击准确率往往也会随之提高，而 KNN距离则会降低。这表明，高质量的合成数据集更有可能导致隐私泄露。同一方法中使用不同的模型架构对 MIV攻击结果影响甚微。这是因为 MIV攻击是在具有一致模型架构的本地模型。当分类任务变得更加复杂时，例如在CIFAR-10中类别数从10增加到100，或在Tiny ImageNet中增加到200，攻击性能会下降。在 [13, 14], 中也观察到了攻击性能的下降，其中数据集不断增加的复杂性和维度可能导致了这一现象。

定性实验结果如图 5、图 7 和图 8 所示。这些结果表明，MIV模型能够

表3：在成员推断攻击中将特征用作输入的影响。

数据集	不将特征用作输入		
	BA	AUC	低误报率下的真阳性率 %
CIFAR-10	0.88/0.67	0.94/0.71	53.1/3.3
CIFAR-100	0.74/0.61	0.82/0.63	9.7/0.4
TinyImageNet-200	0.75/0.59	0.83/0.61	9.1/0.1

表4：模型反演攻击中使用的损失函数对 CIFAR-10 合成数据集的影响。

损失函数		攻击准确率	K近邻距离
L_{cls}	L_{traj}		
×	×	0.47	1048.3
✓	×	0.91	593.2
×	✓	0.68	832.4
✓	✓	0.94	489.1

按类别生成高质量样本。合成样本十分逼真，能够捕捉细微的细节和特征。

4.3 消融研究

在本节中，我们研究 IRA 攻击的几个关键因素如何影响攻击性能。

输入中使用的隐藏输出。为考察 MIA 中作为输入的本地模型隐藏输出如何影响攻击性能，我们使用 SelMatch数据集蒸馏算法和 ResNet18 模型架构，在三个数据集上以 IPC= 100 发起 MIA。实验结果列于表3。可以观察到，当攻击模型的输入中不包含隐藏输出时，攻击性能会显著下降。除了置信度向量（即最终层的 logits）所提供的信息外，隐藏输出还为推断成员身份提供了额外线索。这表明，特征对较高的攻击性能有很大贡献，因为这充分利用了对本地模型的白盒访问。

模型反演攻击中的损失函数。接下来，我们研究模型反演攻击所使用的损失函数如何影响攻击性能。我们考虑四种情形，分别测试分类损失和轨迹损失。随后，我们使用卷积神经网络模型架构，对由 SelMatch算法生成的合成CIFAR-10数据集训练出的本地模型发起攻击。实验结果见表4。结果显示，若不使用这两种损失函数，攻击性能会大幅下降，因为 MIV模型只学习了辅助数据集的分布。与轨迹损失相比，分类损失对攻击性能的影响更大，但轨迹损失同样有助于提升攻击效果。

5 讨论

传统的数据集蒸馏 [1]或数据集凝缩 [2]可能有助于实现隐私保护 [32], , 因为合成数据集包含的敏感信息较少，或者能够使在其上训练的模型达到相近的性能。然而，这一结论已不再适用于最先进的数据集蒸馏技术。合成数据集的质量会直接影响 IRA 的攻击性能。隐私泄露的直接原因可能是，最先进的数据集蒸馏算法试图将权重轨迹保存到合成数据集中。在数据集蒸馏过程中，受害者模型经过训练，会对真实数据集过拟合。攻击者在本地使用合成数据集训练本地模型，使其对合成数据集过拟合。因此，本地模型很可能具有与受害者模型相似的权重，而受害者模型也对真实数据集过拟合。这最终导致 IRA 攻击成功。

正如先前研究所讨论的 [33, 34],，数据集蒸馏生成的合成数据集并非解决隐私问题的完美方案。若合成数据保留了实现效用所需的特征，也就同时为隐私攻击创造了条件。反之，若合成数据能够提供有意义的隐私保护，则往往要付出巨大的效用代价，通常是压制那些使异常值分析具有价值的信号。此外，合成数据的隐私风险并非在所有记录中都一样。异常值本身更为独特，也更容易被区分，是数据集的“阿喀琉斯之踵” [35]。因此，仍需进一步研究如何生成保护隐私的合成数据集。

在数据集蒸馏过程中，可以将 DP-SGD [36, 37] 等潜在防御措施应用于受害者模型的训练过程。通过在训练过程中向梯度添加噪声，受害者模型的权重轨迹会偏离原始轨迹，而偏离程度由隐私预算控制。同样，也可以利用差分隐私 [38]直接对合成样本施加扰动。此外，当前的数据集蒸馏方法倾向于使用软标签，因为软标签有助于控制梯度流。因此，可以对软标签进行扰动，以实现隐私保护。虽然这能保护数据隐私，但不可避免地会在合成数据集的质量与隐私保护效果之间有所取舍。这表明，要生成高质量的合成数据集，同时又不牺牲隐私，很可能是不可能的。

6 结论

我们针对当前最先进的数据集蒸馏方法提出了信息泄露攻击。通过充分利用合成数据集中的信息，攻击者能够对真实数据集和数据集蒸馏算法实施成员推断、架构推断和模型反演攻击。实验结果表明，IRA 的威力足以使攻击者揭露有关真实数据集和数据集蒸馏算法的敏感信息，并对数据集所有者的利益造成不可忽视的损害。这提醒当前与数据集蒸馏相关的研究：这项技术本身存在隐私泄露的重大风险。未来该领域的工作应着重研究如何制定数据集蒸馏的隐私保护方案。

伦理考量

开展这项研究时，我们严格遵循“无害”原则。我们设计的方法旨在纯理论且相互隔离的情境中揭示当前数据集蒸馏方法的脆弱性，确保不会对任何个人、组织或运行系统造成负面影响。

与运行系统隔离 本文所述的所有实验均在完全离线的本地环境中进行。我们采用了数据集蒸馏协议的开源实现，并将其部署在自有硬件上。我们的攻击方法在任何时候都未与在线商业API交互、查询或对其进行压力测试，也未接触外部网络服务。因此，本研究未给任何模型提供商或平台造成服务中断、延迟或经济损失。

使用公开的非敏感数据 本次评估完全使用标准的、公开可用的学术数据集进行。我们没有从私人用户处采集数据，也没有针对特定的内容创作者。仅依靠既有基准数据集，我们确保在本研究过程中未处理任何个人数据，也未侵犯任何个人的隐私或知识产权。

安全评估 我们的工作将合成数据集的敏感信息视为有待分析的数学产物，而不是攻击托管这些信息的基础设施。通过将我们的“红队测试”工作限制在沙箱环境中，我们证明可以在不对更广泛的数字生态系统及其利益相关方构成风险的情况下识别并记录安全漏洞。

开放科学

为促进透明度和可复现性，我们将公开相关资料，包括本文所用所有实证研究的全部代码。代码将在论文录用后发布。我们使用的所有数据集均包含在相关资料中，因为它们均可公开获取。

生成式人工智能的使用

在准备本稿件的过程中，我们严格仅将生成式人工智能工具用于语言润色。具体而言，我们使用这些工具检查语法错误，并改善文本的行文流畅度和可读性。我们声明：

- **不进行内容生成：** 核心概念、实验设计、数据分析和科学结论完全由作者完成。
- **不生成合成文本：** 文本中没有任何大段内容由人工智能生成。这些工具仅充当文字编辑助手。
- **人工监督：** 人工智能提出的所有建议均由作者人工审查和核实，作者对最终出版物的准确性和完整性承担全部责任。

参考文献

[1] T. Wang, J.-Y. Zhu, A. Torralba 和 A. A. Efros，《数据集蒸馏》，*arXiv* 预印本 *arXiv:1811.10959*，2018 年。[2] B. Zhao、K. R. Mopuri 和 H. Bilen，《基于梯度匹配的数据集凝缩》，载于第九届国际学习表征会议 *2021*，2021 年。[3] Y. Hu、F. Wu、Q. Li、Y. Long、G. Garrido、C. Ge、B. Ding、D. Forsyth、B. Li 和 D. Song，“Sok：隐私保护数据合成”，载于 2024 年 IEEE安全与隐私研讨会（SP）。IEEE计算机学会，2023 年，第 2–2 页。[4] M. Giuffrè 和 D. L. Shung，“利用医疗保健中合成数据的力量：创新、应用与隐私”，NPJ数字医学，第 6 卷，186，2023 年 1 月，第 .[5] S. Lei 和 D. Tao，“数据集蒸馏综合综述”，IEEE模式分析与机器智能汇刊，第 46 卷，32，2023 年 1 月，第 17no. pp. – 。[6] R. Yu、S. Liu 和 X. Wang，“数据集蒸馏：全面综述”，IEEE模式分析与机器智能汇刊，第 46 卷，170，2023 年 1 月，第 150no. pp. – 。[7] J. Du、Y. Jiang、V. Y. Tan、J. T. Zhou 和 H. Li，“通过最小化累积轨迹误差改进数据集蒸馏”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2023 年，第 3749–3758 页。[8] Z. Guo、K. Wang、G. Cazenavette、H. LI、K. Zhang 和 Y. You，“通过难度对齐轨迹匹配实现无损数据集蒸馏”，载于第十二届国际学习表征会议，2023 年。[9] Y. Lee 和 H. W. Chung，“SelMatch：通过基于选择的初始化和轨迹匹配部分更新有效扩展数据集蒸馏”，载于第四十一届国际机器学习会议，2024 年。[10] R. Shokri、M. Stronati、C. Song 和 V. Shmatikov，“针对机器学习模型的成员推断攻击”，载于 2017 年 IEEE安全与隐私研讨会（SP）。IEEE，2017 年，第 3–18 页。[11] M. Fredrikson、S. Jha 和 T. Ristenpart，“利用置信度信息的模型反演攻击及基本对策”，载于第 22 届 ACM SIGSAC计算机与通信安全会议论文集，2015 年，第 1322–1333 页。[12] B. Zhang、X. He、Y. Shen、T. Wang 和 Y. Zhang，“一图胜千言：利用科学图表窃取模型信息的攻击”，载于第 32 届 USENIX安全研讨会（USENIX Security 23），2023 年，第 5289–5306 页。[13] R. Liu、D. Wang、Y. Ren、Z. Wang、K. Guo、Q. Qin 和 X. Liu，“无法阻挡的攻击：基于条件扩散模型的仅标签模型反演”，IEEE信息取证与安全汇刊，2024 年。[14] M. Kahla、S. Chen、H. A. Just 和 R. Jia，“基于边界排斥的仅标签模型反演攻击”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2022 年，第 15 045–15 053 页。[15] J. Du、Q. Shi 和 J. T. Zhou，“用于数据集蒸馏的序列子集匹配”，神经信息处理系统进展，第 36 卷，2024 年。[16] Y. Gao、H. Qiu、Z. Zhang、B. Wang、H. Ma、A. Abuadbba、M. Xue、A. Fu 和 S. Nepal，“Deeptheft：通过功耗侧信道窃取深度神经网络模型架构”，载于 2024 年 IEEE安全与隐私研讨会（SP）。IEEE，2024 年，第 3311–3326 页。[17] G. Gao、H. Peng、H. Ma、Z. Zhang、S. Wang、R. Holland、A. Fu、M. Xue 和 D. Abbott，“想污染我的深度学习数据？你的轨迹频谱无处藏身！”，载于网络与分布式系统安全研讨会（NDSS）。互联网协会，2025 年。[18] H. Hu、Z. Salcic、L. Sun、G. Dobbie、P. S. Yu 和 X. Zhang，“机器学习中的成员推断攻击：综述”，ACM计算调查（CSUR），第 54 卷，第 11s 期，第 1–37 页，2022 年。[19] A. Salem、Y. Zhang、M. Humbert、P. Berrang、M. Fritz 和 M. Backes，“ML-Leaks：针对机器学习模型、独立于模型和数据的成员推断攻击与防御”，载于 2019 年网络与分布式系统安全研讨会论文集。互联网协会，2019 年。[20] J. Ho、A. Jain 和 P. Abbeel，“去噪扩散概率模型”，神经信息处理系统进展，第 33 卷，第 6840–6851 页，2020 年。[21] N. Haim、G. Vardi、G. Yehudai、O. Shamir 和 M. Irani，“从训练后的神经网络中重建训练数据”，神经信息处理系统进展，第 35 卷，第 22 911–22 924 页，2022 年。[22] Y. Benny 和 L. Wolf，“动态双输出扩散模型”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2022 年，第 11 482–11 491 页。[23] J. Song、C. Meng 和 S. Ermon，“去噪扩散隐式模型”，载于国际学习表征会议，2021 年。 [在线]。链接：https://openreview.net/forum?id=St1giarCHLP [24] G. Cazenavette、T. Wang、A. Torralba、A. A. Efros 和 J.-Y. Zhu，“通过匹配训练轨迹进行数据集蒸馏”，载于 IEEE/CVF计算机视觉与模式识别会议论文集，2022 年，第 4750–4759 页。[25] H. Li、Z. Li、S. Wu、C. Hu、Y. Ye、M. Zhang、D. Feng 和 Y. Zhang，“SeqMIA：基于序列度量的成员推理攻击”，arXiv 预印本 arXiv:2407.15098，2024 年。[26] N. Carlini、S. Chien、M. Nasr、S. Song、A. Terzis 和 F. Tramer，“基于第一性原理的成员推理攻击”，载于 2022 年 IEEE安全与隐私研讨会（SP）。IEEE，2022 年，第 1897–1914 页。[27] H. Liu、J. Jia、W. Qu 和 N. Z. Gong，“EncoderMI：对比学习中针对预训练编码器的成员推断”，载于 2021 年 ACM SIGSAC计算机与通信安全会议论文集，2021 年，第 2081–2095 页。

[28] F. Boenisch、A. Dziedzic、R. Schuster、A. S. Shamsabadi、I. Shumailov 和 N. Papernot，“当好奇者不再诚实：联邦学习并非隐私保护”，载于 *2023 年 IEEE 第八届欧洲安全与隐私研讨会（EuroS&P）*。IEEE，2023 年，第 175–199 页。[29] W. Issa、N. Moustafa、B. Turnbull 和 K.-K. R. Choo，“Rve-pfl：基于稳健变分编码器的个性化联邦学习，可抵御模型反演攻击”，*IEEE信息取证与安全汇刊*，第 19 卷，3787，2024 年，第 3772– 页。[30] D. Das、J. Zhang 和 F. Trantèr，“盲基线胜过针对基础模型的成员推断攻击”，载于 *2025 年 IEEE安全与隐私研讨会（SPW）*。IEEE，2025 年，第 118–125 页。[31] J. Shang、J. Wang、K. Wang、J. Liu、N. Jiang、M. Armanuzzaman 和 Z. Zhao，“针对迭代剪枝深度神经网络的成员推断攻击防御。”载于 *NDSS*，2025 年。[32] T. Dong、B. Zhao 和 L. Lyu，“免费隐私：数据集凝缩如何助力隐私保护？ ”，载于 国际机器学习会议。PMLR，2022 年，第 5378–5396 页。[33] T. Stadler、B. Oprisanu 和 C. Troncoso，“合成数据匿名化：似曾相识的一天”，载于 第 *31 届 USENIX安全研讨会（USENIX Security 22）*，2022 年，第 1451–1468 页。[34] M. Giomi、F. Boenisch、C. Wehmeyer 和 B. Tasnádi，“用于量化合成数据隐私风险的统一框架”，隐私增强技术论文集，第 2 卷，第 312–328 页，2023 年。[35] M. Meeus、F. Guepin、A.-M. Creţu 和 Y.-A. de Montjoye，“阿喀琉斯之踵：合成数据发布中的脆弱记录识别”，载于 欧洲计算机安全研究研讨会。Springer，2023 年，第 380–399 页。[36] M. Abadi、A. Chu、I. Goodfellow、H. B. McMahan、I. Mironov、K. Talwar 和 L. Zhang，“差分隐私下的深度学习”，载于 *2016 年 ACM SIGSAC计算机与通信安全会议论文集*，2016 年，第 308–318 页。[37] J. Jordon、J. Yoon 和 M. Van Der Schaar，“Pate-gan：生成具有差分隐私保证的合成数据”，载于 国际学习表征会议，2018 年。[38] K. Zhang、Y. Zhang、R. Sun、P.-W. Tsai、M. U. Hassan、X. Yuan、M. Xue 和 J. Chen，“有界且无偏的复合差分隐私”，载于 *2024 年 IEEE安全与隐私研讨会（SP）*。IEEE，2024 年，第 972–990 页。[39] Z. Ji 和 M. Telgarsky，“梯度下降使深度线性网络的各层对齐”，载于 国际学习表征会议，2019 年。 [在线]。链接：https://openreview.net/forum?id=HJflg30qKX [40] K. Lyu 和 J. Li，“梯度下降最大化齐次神经网络的间隔”，载于 国际学习表征会议，2020 年。[在线]。链接：https://openreview.net/forum?id=SJeLIgBKPS [41] H. Amann 和 J. Escher，《分析学*II*》。*Silvio Levy* 和 *Matthew Cargo* 从德文译出。巴塞尔：Birkhäuser，2008 年。

架构推断攻击的理论分析

我们将讨论数据集与训练损失之间的关系。我们希望以下结果有助于理解所提出的架构推断攻击的有效性，该攻击利用损失轨迹识别数据集蒸馏方法。有关训练损失理论分析的参考资料，请参阅 [39, 40] 。

首先，我们列出假设和符号。设 d为输入向量的维度，C为类别数。数据集是若干元素的集合。数据集中的一个元素是由输入样本向量 $\boldsymbol{x}$及其对应的标签 $y \in \{1, 2, \ldots\ldots, C\}$组成的有序对。对于有限维向量 $\boldsymbol{a}$，$\|\boldsymbol{a}\|_2$表示向量 $\boldsymbol{a}$的 l_2-范数，而 $B(\boldsymbol{a}, \delta)$表示以 $\boldsymbol{a}$为中心、半径为 δ的开球。

神经网络。 我们将神经网络视为一个向量值函数

$$\boldsymbol{f}: \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^C, (\boldsymbol{x}, \boldsymbol{\theta}) \mapsto (f_1(\boldsymbol{x}, \boldsymbol{\theta}), \ldots, f_C(\boldsymbol{x}, \boldsymbol{\theta})),$$

其中 $\boldsymbol{\theta}$ 表示神经网络中的所有参数。

当我们训练神经网络时，样本是固定的，我们将 $\boldsymbol{f}$视为 $\boldsymbol{\theta}$的函数。完成训练过程后，我们将 $\boldsymbol{f}$视为 $\boldsymbol{x}$的函数，并用它来对测试输入进行分类。然而，在后续讨论中，对于每个 $i \in \{1, 2, \ldots\ldots, C\}$，我们将 f_i视为 $\mathbb{R}^d \times \mathbb{R}^m$上的函数。

损失函数。 我们通过交叉熵损失来定义 $\boldsymbol{f}$在单个样本点（$\boldsymbol{x}$，y）处的损失函数

$$L(\boldsymbol{x}, \boldsymbol{\theta}) = -\log \frac{e^{f_y(\boldsymbol{x}, \boldsymbol{\theta})}}{\sum_{j=1}^{C} e^{f_j(\boldsymbol{x}, \boldsymbol{\theta})}} \tag{A.1}$$

并在数据集 D 上通过全局交叉熵损失来定义

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) := -\frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x}, y) \in Z} L(\boldsymbol{x}, \boldsymbol{\theta}). \tag{A.2}$$

梯度。 损失函数 $\mathrm{L}_{\mathcal{D}}$ 的梯度为

$$\nabla \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{D}} \left[\frac{\sum_{i=1}^{C} \nabla f_i(\boldsymbol{x}, \boldsymbol{\theta}) e^{f_i(\boldsymbol{x}, \boldsymbol{\theta})}}{\sum_{i=1}^{C} e^{f_i(\boldsymbol{x}, \boldsymbol{\theta})}} - \nabla f_y(\boldsymbol{x}, \boldsymbol{\theta}) \right].$$

梯度下降。我们假设训练过程中损失函数 L_Z是一个 C^1-函数，并将梯度下降过程描述为 $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta(t) \nabla \mathrm{L}_Z(\boldsymbol{\theta}(t))$，其中 $\eta(t)$ 是时刻 t的学习率。

我们假设 $\mathrm{D}_1 = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ 是一个包含 N 个样本的数据集。那么，

$$\mathcal{L}_{\mathcal{D}_1}(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^{N} \log \frac{e^{f_{y_n}(\boldsymbol{x}_n, \boldsymbol{\theta})}}{\sum_{i=1}^{C} e^{f_i(\boldsymbol{x}_n, \boldsymbol{\theta})}}.$$

显然，

$$\nabla \mathcal{L}_{\mathcal{D}_1}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \left[\frac{\sum_{i=1}^{C} \nabla f_i(\boldsymbol{x}_n, \boldsymbol{\theta}) e^{f_i(\boldsymbol{x}_n, \boldsymbol{\theta})}}{\sum_{i=1}^{C} e^{f_i(\boldsymbol{x}_n, \boldsymbol{\theta})}} - \nabla f_{y_n}(\boldsymbol{x}_n, \boldsymbol{\theta}) \right].$$

我们用 $\boldsymbol{\theta}_0$ 表示神经网络的初始参数，并将其在 D_1 上的训练过程描述为

$$\begin{aligned} &\boldsymbol{\theta}_1(0) = \boldsymbol{\theta}_0; \\ &\boldsymbol{\theta}_1(t+1) = \boldsymbol{\theta}_1(t) - \eta(t) \nabla \mathcal{L}_{\mathcal{D}_1}(\boldsymbol{\theta}_1(t)). \end{aligned} \tag{A.3}$$

在接下来的讨论中，我们对神经网络作如下假设：

(A1) 对于每个 $i \in \{1, 2, \ldots\ldots, C\}$，$f_i$是定义在 $\mathbb{R}^d \times \mathbb{R}^m$上的 C^1-函数；

(A2) 对每个 $i \in \{1, 2, \ldots\ldots, C\}$，$\frac{\partial f_i}{\partial \theta_l}$在 $\mathbb{R}^d \times \mathbb{R}^m$ 上局部利普希茨；

我们对数据集作如下假设：

(A3) 存在一个常数 $M_1 > 0$，使得对于所有 $n \in \{1, 2, \ldots\ldots, N\}$，$\|\boldsymbol{x}_n\|_2 \le M_1$。

我们对 D_1 上的训练过程作如下假设：

(A4) 存在合适的训练轮次数量 $T_0 \in \mathbb{N}_+$ 和适当的常数学习率 $\eta(t) = \eta_0 > 0$，使得神经网络经过 T_0 个训练轮次后，准确率足够高。

(A5) 存在一个常数 $M_2 > 0$，使得对于所有 $t \in \{1, 2, \ldots\ldots, T_0\}$，$\|\boldsymbol{\theta}_1(t)\|_2 \le M_2$。

在大多数具体任务中，输入向量集合和参数集合都是有界集。因此，设定 (A3) 和 (A5) 似乎是合理的。

令 $D = \{(\boldsymbol{x}, \boldsymbol{\theta}) : \|\boldsymbol{x}\|_2 \le M_1 + 1; \|\boldsymbol{\theta}\|_2 \le M_2 + 1\}$。则 D是 $\mathbb{R}^d \times \mathbb{R}^m$的闭有界子集。因此，$D$在 $\mathbb{R}^d \times \mathbb{R}^m$上是紧的。

引理 A.1。 对于任意 $(\boldsymbol{x}', \boldsymbol{\theta}') \neq (\boldsymbol{x}'', \boldsymbol{\theta}'') \in D$，存在 $L_1 > 0$，使得

$$|f_i(\boldsymbol{x}', \boldsymbol{\theta}') - f_i(\boldsymbol{x}'', \boldsymbol{\theta}'')| \leq L_1(\|\boldsymbol{x}' - \boldsymbol{x}''\|_2 + \|\boldsymbol{\theta}' - \boldsymbol{\theta}''\|_2),$$

对于每个 $i \in \{1、2、\ldots、C\}$。

证明。对于任意固定的 $i \in \{1、2、\ldots、C\}$，f_i根据 (A1) 是 $\mathbb{R}^d \times \mathbb{R}^m$ 上的 C^1-函数。
容易验证，对于任意 $j \in \{1、2、\ldots、d\}$ $k \in \{1、2、\ldots、m\}$，$\frac{\partial f_i}{\partial x_j}$和 $\frac{\partial f_i}{\partial \theta_k}$在 D 上有界。于是，我们有

$$|f_i(\boldsymbol{x}', \boldsymbol{\theta}') - f_i(\boldsymbol{x}'', \boldsymbol{\theta}'')| \leq L_1(\|\boldsymbol{x}' - \boldsymbol{x}''\|_2 + \|\boldsymbol{\theta}' - \boldsymbol{\theta}''\|_2),$$

根据 [41, 定理 VII.3，9], 其中

$$L_1 = \max_{j,k} \left\{ \max_D \left| \frac{\partial f_i}{\partial x_j}(\boldsymbol{x}, \boldsymbol{\theta}) \right|, \max_D \left| \frac{\partial f_i}{\partial \theta_k}(\boldsymbol{x}, \boldsymbol{\theta}) \right| \right\}.$$

□

引理 A.2. 对于任意 $(\boldsymbol{x}', \boldsymbol{\theta}') \neq (\boldsymbol{x}'', \boldsymbol{\theta}'') \in D$，存在 $L_2 > 0$，使得

$$\left| \frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}', \boldsymbol{\theta}') - \frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}'', \boldsymbol{\theta}'') \right| \leq L_2(\|\boldsymbol{x}' - \boldsymbol{x}''\|_2 + \|\boldsymbol{\theta}' - \boldsymbol{\theta}''\|_2),$$

对于每个 $i \in \{1, 2, \ldots, C\}$ $l \in \{1, 2, \ldots, m\}$。

证明。对于任意固定的 $i \in \{1, 2, \ldots, C\}$ $l \in \{1, 2, \ldots, m\}$，根据 (A2)，$\frac{\partial f_i}{\partial \theta_l}$ 在 $\mathbb{R}^d \times \mathbb{R}^m$ 上局部利普希茨。

对于任意 $(\boldsymbol{x}, \boldsymbol{\theta}) \in D$，存在 $\delta_{(\boldsymbol{x},\boldsymbol{\theta})}$，使得 $\frac{\partial f_i}{\partial \theta_l}$ 在 $B((\boldsymbol{x}, \boldsymbol{\theta}); \delta_{(\boldsymbol{x},\boldsymbol{\theta})})$ 上满足利普希茨条件，且利普希茨常数为 $Lip_{(\boldsymbol{x},\boldsymbol{\theta})}$。显然，$\{B((\boldsymbol{x}, \boldsymbol{\theta}); \delta_{(\boldsymbol{x},\boldsymbol{\theta})}) : (\boldsymbol{x}, \boldsymbol{\theta}) \in D\}$ 是 D 的一个开覆盖。因此，存在一个有限子覆盖 $\{B(\boldsymbol{z}_n; \delta_{\boldsymbol{z}_n}) : n = 1, 2, \ldots, n_0\}$。因此，我们有

$$\left| \frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}', \boldsymbol{\theta}') - \frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}'', \boldsymbol{\theta}'') \right| \leq L_2(\|\boldsymbol{x}' - \boldsymbol{x}''\|_2 + \|\boldsymbol{\theta}' - \boldsymbol{\theta}''\|_2),$$

其中 $L_2 = \max_{n=1,2,\ldots,n_0} Lip_{\boldsymbol{z}_n}$。 □

我们希望验证：如果 D_2 与 D_1足够接近，那么 D_2 上的训练损失将与 D_1 上的非常接近。为了应用数学工具，我们将 D_2 视为 D_1的扰动。在考察训练损失之前，我们先分析决定训练损失的参数变化。

定理 A.1。 假设$\mathcal{D}_2$ 是$\mathcal{D}_1$, 的扰动，即

$$\mathcal{D}_2 = \{(\boldsymbol{x}_n + \boldsymbol{h}_n, y_n) : \|\boldsymbol{h}_n\|_2 < \delta, n = 1, 2, \ldots, N\},$$

其中 $\delta > 0$ 是一个常数。将$\mathcal{D}_2$ 上的训练过程描述为

$$\begin{aligned} &\boldsymbol{\theta}_2(0) = \boldsymbol{\theta}_0, \\ &\boldsymbol{\theta}_2(t+1) = \boldsymbol{\theta}_2(t) - \eta(t)\nabla \mathcal{L}_{\mathcal{D}_2}(\boldsymbol{\theta}_2(t)). \end{aligned} \tag{A.4}$$

对于任意 $\varepsilon > 0$，我们有

$$\|\boldsymbol{\theta}_1(t) - \boldsymbol{\theta}_2(t)\| < \varepsilon, \ t \in \{1, 2, \ldots, T_0\}, \tag{A.5}$$

当 δ 足够小时。

然后，利用损失函数在每个单独样本点处的局部连续性，我们立即得出关于训练损失的以下结果。

推论 A.1. 对于任意 $\varepsilon > 0$，我们有

$$|\mathcal{L}_{\mathcal{D}_1}(\boldsymbol{\theta}_1(t)) - \mathcal{L}_{\mathcal{D}_2}(\boldsymbol{\theta}_2(t))| < \varepsilon, t \in \{0, 1, \ldots, T_0\}, \tag{A.6}$$

当 δ 足够小时。

定理 A.1 的证明。由 (A.3) 和 (A.4)，我们得到以下估计

$$\begin{aligned} \|\boldsymbol{\theta}_1(t+1) - \boldsymbol{\theta}_2(t+1)\|_2 &\leq \|\boldsymbol{\theta}_1(t) - \boldsymbol{\theta}_2(t)\|_2 \\ &+ \eta(t)\|\nabla \mathcal{L}_{\mathcal{D}_1}(\boldsymbol{\theta}_1(t)) - \nabla \mathcal{L}_{\mathcal{D}_2}(\boldsymbol{\theta}_2(t))\|_2, \end{aligned} \tag{A.7}$$

其中 $t \in \{0, 1, \ldots, T_0 - 1\}$。

根据引理 A.1，我们有

$$\begin{aligned}
&\sum_{i=1}^{C}\left|\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))\right|\left|\frac{e^{f_i(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))}}-\frac{e^{f_i(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}\right| \\
&\le \sum_{i=1}^{C}\left|\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))\right|\frac{e^{f_i(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))}}\sum_{k=1}^{C}\frac{\left|e^{f_k(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))-f_k(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}-1\right|e^{f_k(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}} \\
&\quad+\sum_{i=1}^{C}\left|\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))\right|\frac{\left|e^{f_i(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))-f_i(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}-1\right|e^{f_i(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}\le 4L_1^2(\|\boldsymbol{h}_n\|_2+\|\boldsymbol{\theta}_1(t)-\boldsymbol{\theta}_2(t)\|_2),
\end{aligned}\tag{A.8}$$

对于每个 $l=\{1,2,,m\}$ 和 $n\in\{1,2,,N\}$.

根据引理 A.2，我们有

$$\left|\frac{\partial f_{y_n}}{\partial \theta_l}(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))-\frac{\partial f_{y_n}}{\partial \theta_l}(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))\right|\le L_2(\|\boldsymbol{h}_n\|_2+\|\boldsymbol{\theta}_1(t)-\boldsymbol{\theta}_2(t)\|_2),\tag{A.9}$$

and

$$\sum_{i=1}^{C}\frac{\left|\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))-\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))\right|e^{f_i(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}\le L_2(\|\boldsymbol{h}_n\|_2+\|\boldsymbol{\theta}_1(t)-\boldsymbol{\theta}_2(t)\|_2),\tag{A.10}$$

对于每个 $l=\{1,2,\ldots,m\}$ 和 $n\in\{1,2,\ldots,N\}$。

由 (A.8) 和 (A.10) 可知

$$\begin{aligned}
&\left|\sum_{i=1}^{C}\frac{\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))e^{f_i(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))}}-\sum_{i=1}^{C}\frac{\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))e^{f_i(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}\right| \\
&\le\sum_{i=1}^{C}\left|\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))-\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))\right|\frac{e^{f_i(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}} \\
&\quad+\sum_{i=1}^{C}\left|\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))\right|\left|\frac{e^{f_i(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))}}-\frac{e^{f_i(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}\right| \\
&\le L_2(\|\boldsymbol{h}_n\|_2+\|\boldsymbol{\theta}_1(t)-\boldsymbol{\theta}_2(t)\|_2)+4L_1^2(\|\boldsymbol{h}_n\|_2+\|\boldsymbol{\theta}_1(t)-\boldsymbol{\theta}_2(t)\|_2),
\end{aligned}\tag{A.11}$$

对于每个 $l=\{1、2、\ldots、m\}$ 以及 $n\in\{1、2、\ldots、N\}$。

由（A.9）和（A.11）可知

$$\begin{aligned}
&\left|\frac{\partial \mathcal{L}_{\mathcal{D}_1}}{\partial \theta_l}(\boldsymbol{\theta}_1(t))-\frac{\partial \mathcal{L}_{\mathcal{D}_2}}{\partial \theta_l}(\boldsymbol{\theta}_2(t))\right|\le\frac{1}{N}\sum_{n=1}^{N}\left|\frac{\partial f_{y_n}}{\partial \theta_l}(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))-\frac{\partial f_{y_n}}{\partial \theta_l}(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))\right| \\
&+\frac{1}{N}\sum_{n=1}^{N}\left|\sum_{i=1}^{C}\frac{\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))e^{f_i(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))}}-\sum_{i=1}^{C}\frac{\frac{\partial f_i}{\partial \theta_l}(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))e^{f_i(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}{\sum_{j=1}^{C}e^{f_j(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))}}\right| \\
&\le(4L_1^2+2L_2)(\delta+\|\boldsymbol{\theta}_1(t)-\boldsymbol{\theta}_2(t)\|_2).
\end{aligned}\tag{A.12}$$

此外，我们有

$$\|\nabla\mathcal{L}_{\mathcal{D}_1}(\boldsymbol{\theta}_1(t))-\nabla\mathcal{L}_{\mathcal{D}_2}(\boldsymbol{\theta}_2(t))\|_2\le\sqrt{m}(4L_1^2+2L_2)(\delta+\|\boldsymbol{\theta}_1(t)-\boldsymbol{\theta}_2(t)\|_2).\tag{A.13}$$

因此，我们得出

$$\|\boldsymbol{\theta}_1(t+1)-\boldsymbol{\theta}_2(t+1)\|_2+\delta\le[1+\eta_0\sqrt{m}(4L_1^2+2L_2)](\|\boldsymbol{\theta}_1(t)-\boldsymbol{\theta}_2(t)\|_2+\delta),$$

根据（A.7）和（A.13）。于是，我们得到以下估计

$$\|\boldsymbol{\theta}_1(t)-\boldsymbol{\theta}_2(t)\|_2\le[(1+\eta_0\sqrt{m}(4L_1^2+2L_2))^t-1]\delta,t\in\{0,1,\ldots,T_0\}.\tag{A.14}$$

因此，当 $\delta<$ $\frac{\varepsilon}{(1+\eta_0\sqrt{m}(4L_1^2+2L_2))^{T_0}-1}$，时，我们有（A.5） □

推论 A.1 的证明。对于 任意固定的训练轮次 $t\in\{0、1、\ldots\ldots、T_0\}$ 以及任意固定的样本 $(\boldsymbol{x}_n,\ y_n)\ \in\mathrm{D}_1,$，单个样本点处的损失函数本点 L 在 $(\boldsymbol{x}_n,\ \boldsymbol{\theta}_1(t)$ 处连续)。因此，存在 $\iota_{n,\ t_0}(\varepsilon)\ >0$ 使得

$$|L(\boldsymbol{x}',\boldsymbol{\theta}')-L(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))|<\varepsilon,$$

当 $(\boldsymbol{x}',\boldsymbol{\theta}')\ \in B((\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))\,\iota_{n,t}(\varepsilon))$。

利用我们在定理 A.1 中证明的估计式 (A.14)，我们推出当

$$\delta < \min\left\{\frac{\iota_{n,t}(\varepsilon)}{[1+\eta_0\sqrt{m}(4L_1^2+2L_2)]^t} : n=1,2,\ldots,N, t=0,1,\ldots,T_0\right\}, \tag{A.15}$$

我们有

$$\|(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))-(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))\| < \iota_{n,t}(\varepsilon), t\in\{0,1,\ldots,T_0\},\ n\in\{1,2,\ldots,N\},$$

因此我们有

$$|L(\boldsymbol{x}_n+\boldsymbol{h}_n,\boldsymbol{\theta}_2(t))-L(\boldsymbol{x}_n,\boldsymbol{\theta}_1(t))| < \varepsilon,$$
$$t\in\{0,1,\ldots,T_0\},\ n\in\{1,2,\ldots,N\}.$$

因此，当 (A.15) 成立时，(A.6) 也成立。 □

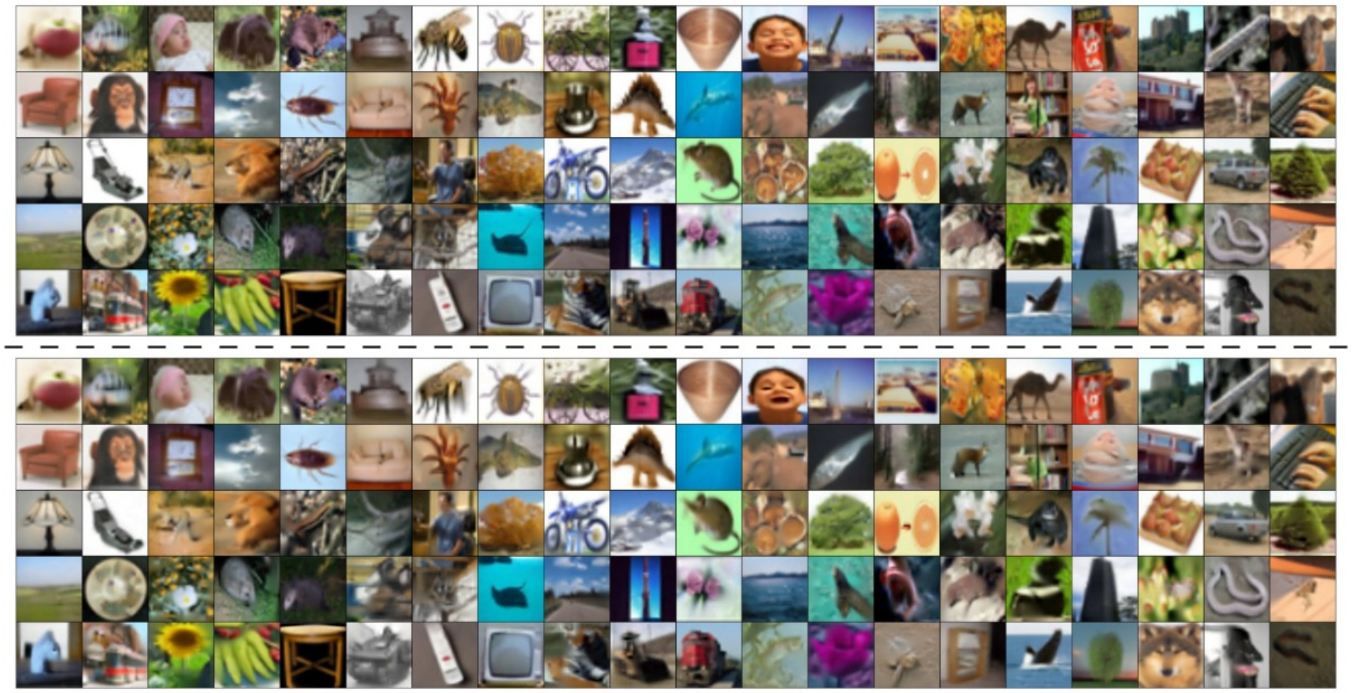


图 7：MIV 在 CIFAR-100 上的定性结果。真实样本位于上方，相应的恢复样本位于下方。



图 8：MIV攻击在 TinyImagenet-200 上的定性结果。

表5：在 CIFAR-100数据集上对最先进的数据集蒸馏算法进行 IRA 评估。

方法	模型架构	测试集 Top-1 准确率 % IPC 10	25	50	100	成员推断攻击 平衡准确率†/曲线下面积†/低误报率下的真阳性率 %↑ IPC 10	25	50	100	属性推断攻击 Top-1 准确率 %↑ IPC 10	25	50	100	MIV 攻击准确率↑/K近邻距离↓ IPC 10	25	50	100
MTT	卷积神经网络	21.5	28.2	34.8	40.5	0.60/0.64/0.6	0.65/0.71/1.8	0.73/0.81/9.5	0.79/0.86/18.9					0.54/1048.2	0.56/1032.7	0.55/1082.5	0.59/1023.4
	AlexNet	20.7	26.3	35.6	38.7	0.63/0.68/0.9	0.68/0.74/1.9	0.77/0.85/10.4	0.77/0.85/16.0					0.55/1103.7	0.55/1079.2	0.56/1043.9	0.58/1037.5
	ResNet18	22.4	30.1	33.5	41.9	0.62/0.67/1.1	0.65/0.71/1.3	0.73/0.80/7.4	0.81/0.88/24.8					0.53/1072.4	0.54/1034.0	0.56/1052.1	0.56/1089.2
	VGG11-BN	22.5	29.2	36.2	40.7	0.63/0.69/1.4	0.66/0.72/1.9	0.76/0.80/14.1	0.79/0.87/20.9					0.52/1132.1	0.54/1058.6	0.55/1075.2	0.58/1035.2
FTD	卷积神经网络	22.4	26.5	37.6	42.8	0.63/0.68/0.5	0.68/0.75/3.6	0.74/0.82/9.6	0.77/0.85/16.2					0.55/1088.9	0.56/1112.4	0.60/1073.3	0.61/1067.3
	AlexNet	22.1	25.7	31.8	38.5	0.62/0.67/0.7	0.63/0.68/2.7	0.73/0.81/6.3	0.75/0.83/15.1					0.56/1097.3	0.56/1047.3	0.59/1056.9	0.60/1018.3
	ResNet18	19.3	25.9	36.6	40.5	0.59/0.63/0.3	0.69/0.74/4.0	0.72/0.80/10.5	0.77/0.84/14.5					0.55/1107.2	0.56/1092.1	0.59/1067.4	0.60/1043.5
	VGG11-BN	21.4	24.3	35.7	41.6	0.61/0.65/0.4	0.67/0.73/2.8	0.77/0.86/15.1	0.76/0.85/15.2					0.56/1068.4	0.56/1062.9	0.58/1036.9	0.61/1028.9
DATM	卷积神经网络	26.4	32.6	44.6	51.3	0.66/0.72/2.6	0.74/0.82/10.2	0.76/0.84/12.2	0.82/0.90/30.1					0.53/1089.2	0.57/1056.8	0.60/992.5	0.61/987.3
	AlexNet	24.5	33.7	42.8	50.5	0.68/0.75/3.1	0.75/0.83/9.1	0.75/0.83/8.7	0.80/0.88/16.1					0.55/1058.2	0.56/1043.8	0.59/1014.7	0.62/954.1
	ResNet18	28.3	35.1	44.3	54.2	0.70/0.77/4.2	0.76/0.85/11.7	0.75/0.83/14.1	0.81/0.89/25.9	81.5	83.2	82.7	84.1	0.54/1077.5	0.57/1048.2	0.59/1021.1	0.63/930.4
	VGG11-BN	28.1	33.2	45.0	52.5	0.69/0.76/1.4	0.75/0.83/8.6	0.77/0.85/16.1	0.81/0.89/26.5					0.53/1089.0	0.55/1036.4	0.61/990.6	0.62/966.3
SelMatch	卷积神经网络	32.2	38.6	48.1	54.8	0.74/0.82/9.7	0.76/0.83/15.7	0.79/0.87/23.3	0.82/0.89/28.5					0.54/1067.4	0.55/1036.5	0.59/991.3	0.62/924.5
	AlexNet	31.3	39.9	46.7	52.3	0.73/0.80/8.9	0.77/0.85/12.2	0.78/0.86/17.8	0.81/0.89/25.4					0.54/1064.9	0.54/1084.6	0.58/1003.5	0.61/956.3
	ResNet18	30.6	37.5	49.3	55.4	0.72/0.79/4.3	0.78/0.86/17.4	0.80/0.88/21.7	0.83/0.91/27.5					0.53/1078.4	0.57/1028.5	0.60/988.6	0.62/927.3
	VGG11-BN	30.5	38.4	49.7	53.3	0.74/0.82/7.1	0.79/0.86/17.3	0.79/0.87/17.1	0.82/0.89/18.1					0.54/1089.4	0.55/1042.6	0.58/1014.2	0.60/936.0
SeqMatch	卷积神经网络	27.2	32.8	39.2	43.0	0.70/0.78/5.2	0.73/0.81/8.8	0.77/0.85/16.1	0.77/0.86/17.1					0.54/1089.2	0.54/1077.5	0.57/1037.6	0.58/1024.5
	AlexNet	27.4	31.6	37.2	42.1	0.71/0.77/2.9	0.72/0.80/8.9	0.75/0.83/12.9	0.77/0.85/16.1					0.52/1143.1	0.54/1086.3	0.55/1078.3	0.57/1032.5
	ResNet18	28.5	35.2	38.1	45.9	0.69/0.76/2.6	0.74/0.81/7.5	0.74/0.81/12.7	0.78/0.86/20.9					0.52/1092.3	0.55/1053.1	0.56/1044.8	0.58/1035.4
	VGG11-BN	29.1	33.9	40.4	44.5	0.71/0.77/3.7	0.74/0.82/7.2	0.77/0.85/13.6	0.78/0.86/17.4					0.52/1102.3	0.53/1088.4	0.56/1033.3	0.58/1051.2

表6：评估 IRA 在 TinyImageNet-200 数据集上对最先进的数据集蒸馏算法的 t.

方法	模型架构	测试集 Top-1 准确率 % IPC 10	25	50	成员推断攻击平衡准确率†/曲线下面积†/低误报率下的真阳性率 %↑ IPC 10	25	50	属性推断攻击 Top-1 准确率 %↑ IPC 10	25	50	MIV 攻击准确率↑/KNN 距离↓ IPC 10	25	50
MTT	卷积神经网络	15.3	19.7	21.5	0.65/0.70/0.7	0.68/0.75/1.8	0.71/0.78/3.3				0.32/1359.2	0.34/1344.2	0.36/1306.8
	AlexNet	15.1	17.8	19.9	0.63/0.68/1.3	0.66/0.71/1.6	0.70/0.77/2.8				0.33/1382.1	0.35/1363.7	0.35/1327.8
	ResNet18	14.7	19.4	22.1	0.62/0.69/0.9	0.68/0.74/2.2	0.71/0.78/3.5				0.32/1402.8	0.34/1352.8	0.35/1323.5
	VGG11-BN	15.4	18.6	21.7	0.61/0.66/1.0	0.67/0.74/2.8	0.69/0.75/3.7				0.32/1376.9	0.34/1368.9	0.36/1324.9
FTD	卷积神经网络	18.9	23.4	26.9	0.65/0.71/3.6	0.72/0.80/8.3	0.74/0.82/7.3				0.33/1374.6	0.34/1324.7	0.36/1309.4
	AlexNet	18.1	21.8	23.5	0.69/0.76/2.5	0.71/0.77/5.6	0.72/0.79/5.7				0.35/1342.8	0.35/1347.3	0.36/1318.4
	ResNet18	19.4	24.3	27.7	0.70/0.77/4.5	0.74/0.82/9.1	0.74/0.81/9.2				0.35/1328.4	0.36/1320.7	0.36/1302.8
	VGG11-BN	18.5	23.7	26.4	0.68/0.74/3.2	0.73/0.81/6.7	0.74/0.82/7.3				0.34/1342.7	0.35/1311.7	0.36/1321.4
DATM	卷积神经网络	23.5	32.6	38.3	0.72/0.79/4.1	0.77/0.85/11.4	0.79/0.87/15.9				0.35/1325.6	0.36/1287.4	0.37/1253.8
	AlexNet	24.8	31.0	36.7	0.73/0.80/7.0	0.77/0.85/11.1	0.78/0.86/13.2				0.34/1347.8	0.36/1317.2	0.36/1299.4
	ResNet18	23.0	33.6	38.5	0.72/0.79/5.2	0.78/0.85/13.2	0.78/0.86/14.8	75.2	73.6	77.0	0.35/1331.5	0.36/1307.8	0.36/1297.0
	VGG11-BN	24.2	31.8	37.2	0.71/0.78/3.8	0.76/0.84/9.3	0.79/0.87/12.8				0.35/1326.9	0.35/1343.9	0.37/1285.4
SelMatch	卷积神经网络	25.9	34.2	40.5	0.75/0.83/10.5	0.77/0.85/12.8	0.79/0.87/18.3				0.35/1337.1	0.35/1346.2	0.37/1264.8
	AlexNet	25.7	33.7	38.9	0.74/0.82/6.7	0.76/0.85/13.1	0.78/0.86/17.0				0.34/1351.2	0.35/1322.2	0.35/1298.4
	ResNet18	26.3	34.9	41.8	0.75/0.83/9.1	0.78/0.86/12.3	0.79/0.87/16.1				0.33/1368.5	0.35/1317.3	0.36/1289.5
	VGG11-BN	24.2	35.4	39.9	0.74/0.82/9.3	0.77/0.85/15.4	0.79/0.87/19.7				0.34/1363.2	0.36/1306.7	0.36/1294.8
SeqMatch	卷积神经网络	23.0	28.6	34.9	0.71/0.79/5.2	0.74/0.82/9.3	0.77/0.84/11.5				0.33/1365.9	0.34/1354.6	0.35/1327.1
	AlexNet	23.1	30.1	33.7	0.73/0.80/7.6	0.75/0.83/9.6	0.78/0.86/13.7				0.33/1358.3	0.34/1363.8	0.35/1348.3
	ResNet18	21.8	29.7	33.8	0.71/0.77/3.8	0.76/0.83/7.4	0.77/0.86/9.7				0.34/1382.0	0.35/1371.8	0.35/1362.1
	VGG11-BN	22.5	30.7	34.3	0.72/0.79/7.7	0.76/0.84/10.6	0.76/0.84/10.3				0.34/1361.7	0.34/1350.1	0.36/1305.8

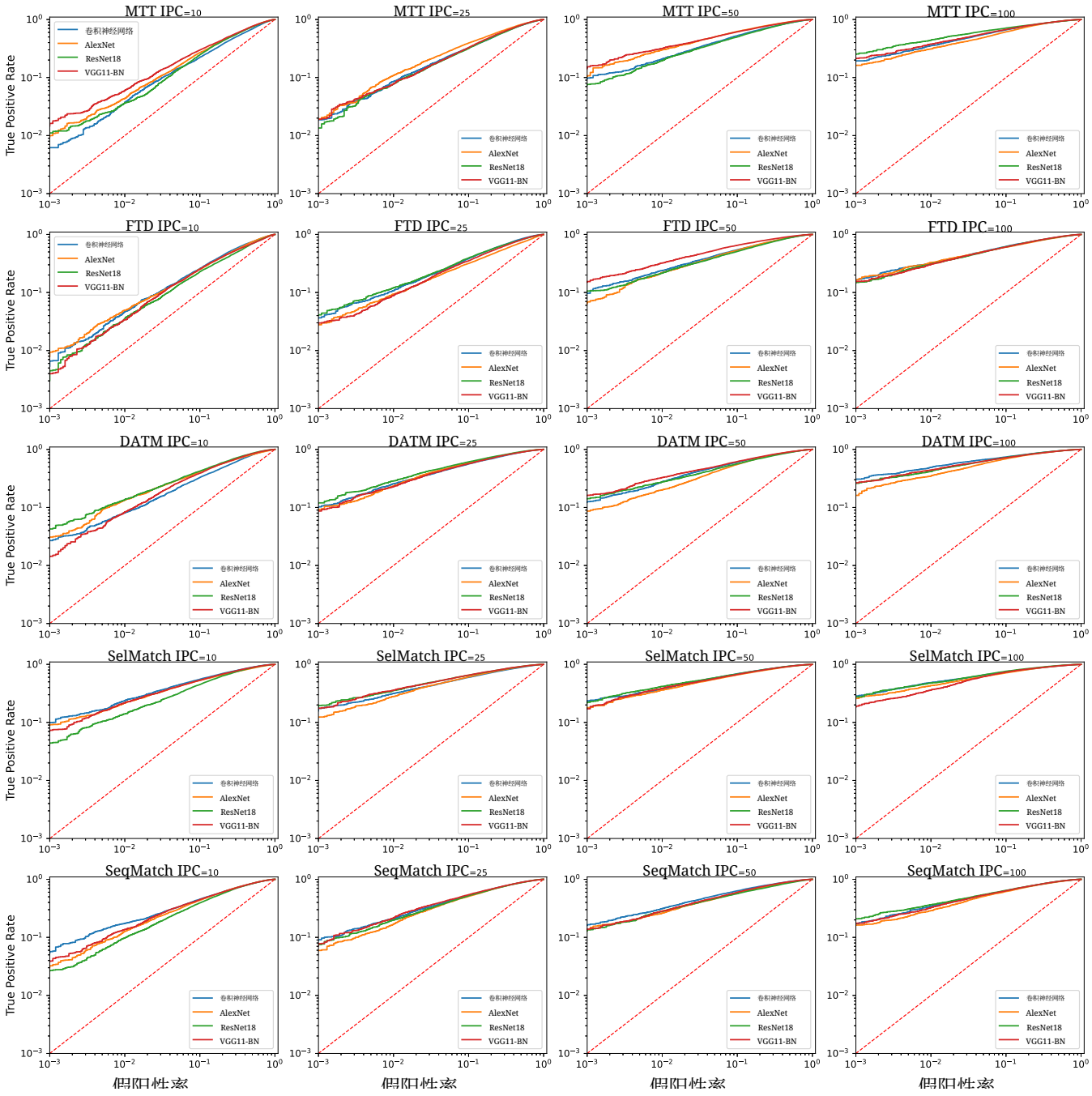


图9：合成CIFAR-100数据集上成员推断攻击的对数尺度ROC曲线。由于页面限制，请放大查看细节。

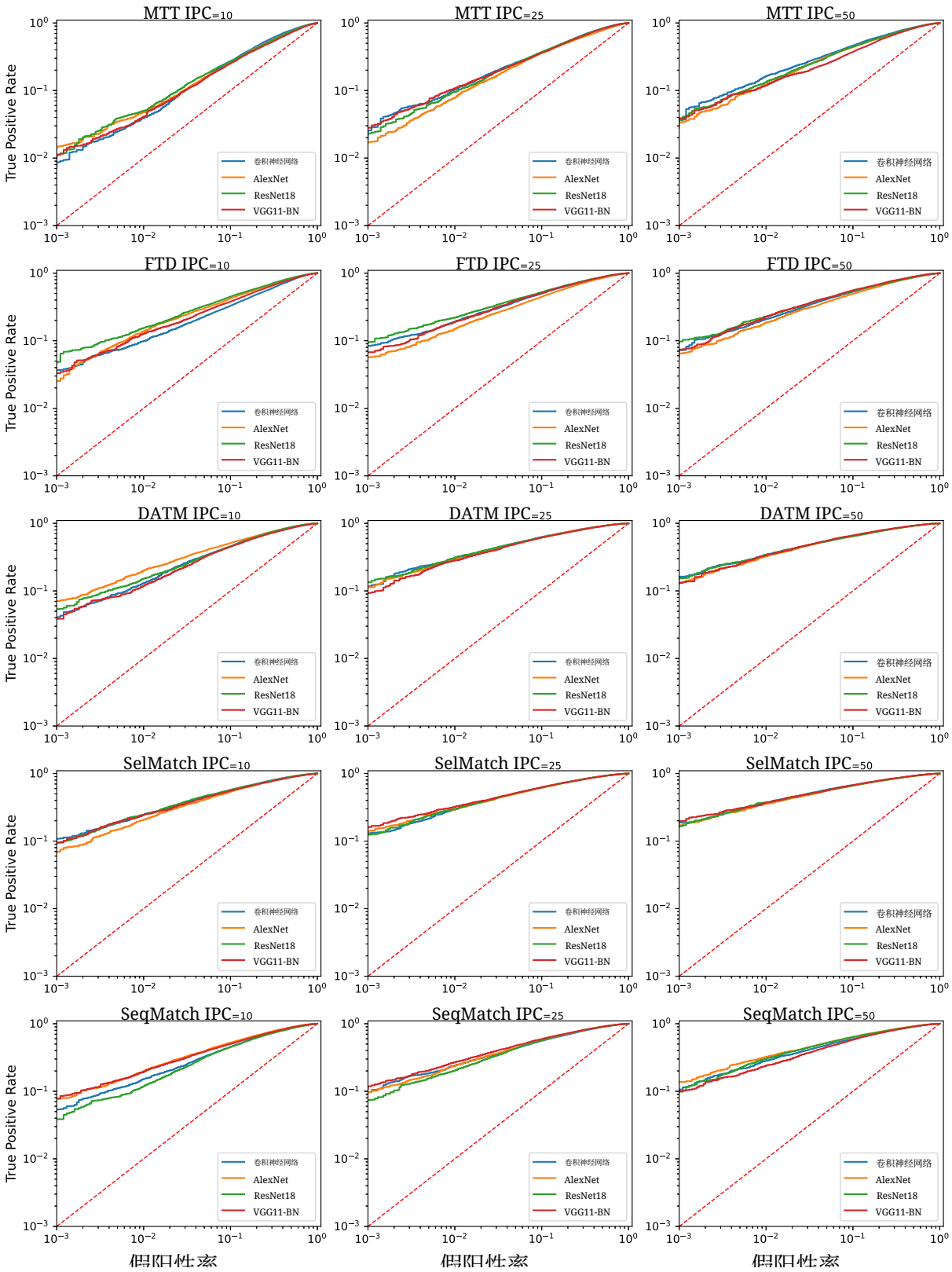


图10：合成TinyImagenet-200数据集上成员推断攻击的对数尺度ROC曲线。由于页面限制，请放大查看。