基于元学习的通用黑盒对抗攻击

Fei Yin*, Yong Zhang*, Baoyuan Wu*†, *IEEE* 会员，Yan Feng、Jingyi Zhang、YanboFan、Yujiu Yang†，*IEEE* 会员

摘要——在黑盒对抗攻击场景中，目标模型的参数未知，攻击者旨在查询预算内根据查询反馈找到有效的对抗扰动。由于反馈信息有限，现有基于查询的黑盒攻击方法通常需要对每个良性样本进行大量查询。为降低查询成本，我们提出利用历史攻击中的反馈信息，这种现象称为样本级对抗迁移性。具体而言，我们将对每个良性样本的攻击视为一个任务，并通过训练元生成器来生成以良性样本为条件的扰动，从而构建元学习框架。在攻击新的良性样本时，元生成器可以根据新任务的反馈信息以及少量历史攻击快速微调，以生成有效扰动。此外，由于元训练过程需要消耗大量查询来学习具有泛化能力的生成器，我们利用模型级对抗可迁移性，在白盒替代模型上训练元生成器，再将其迁移过来，辅助攻击目标模型。所提出的结合两类对抗迁移性的框架可以自然地与任何现成的基于查询的攻击方法相结合，以提升其性能；大量实验验证了这一点。源代码可在 https://github.com/SCLBD/MCG-Blackbox 获取。

索引词—黑盒对抗攻击、元学习、样本级与模型级对抗迁移性、扰动的条件分布

arXiv:2301.00364v1 [cs.LG] 1 Jan 2023

1 引言

DEEP 深度神经网络（DNN）已被证明容易受到对抗样本的攻击 [1],：在良性样本上添加隐蔽且恶意的扰动，以欺骗 DNN 模型。根据可获取的受攻击 DNN 模型（*i.e.*，即目标模型）的信息，现有对抗攻击通常可分为两种场景。第一种是**白盒攻击**，假设攻击者了解目标模型的参数，因此可以根据目标模型的梯度轻松生成对抗扰动。第二种是**黑盒攻击**，攻击者不了解目标模型的参数，只能获取查询反馈。与白盒场景相比，黑盒场景因此，这项工作聚焦于黑盒场景，尤其是基于分数的黑盒攻击，其中反馈是一个连续分数（*i.e.*，分类问题中的后验概率）。

在基于分数的黑盒攻击场景中，对一个良性样本发起攻击的一般过程可描述如下：给定查询预算（*i.e.*，允许的查询次数）以及被攻击良性样本附近允许的扰动区域，并以该良性样本作为初始解，攻击者不断搜索满足黑盒目标模型攻击目标的成功扰动；如果在查询预算内于允许的扰动区域中找到这样的成功扰动，则攻击成功并停止。否则，攻击失败。

由于良性图像的图像统计特性各不相同，攻击一张良性图像可以视为一项独立任务，其中目标模型提供的反馈信息可作为监督信号，引导扰动生成。这一视角启发我们利用不同任务中的信息来学习通用先验知识，并将其迁移到各个独立任务中以提升性能，这与元学习的做法相同。直观来说，攻击更多良性样本时，攻击者应当更熟悉如何根据给定的良性图像生成扰动，也会更加了解目标模型。因此，与初次发起攻击的攻击者相比，经验丰富的攻击者在攻击新的良性样本时，预计能以更少的查询找到成功扰动。其内在依据是，不同良性样本周围的对抗扰动可能具有某些相似属性，我们将其称为**样本级对抗迁移性**。一个典型例子是通用对抗扰动 [2],，其中一个扰动可能同时欺骗多个良性样本。然而，据我们所知，样本级对抗迁移性尚未

• *Fei Yin*、*Yan Feng* 和 *Yujiu Yang* 就职于中国北京 *100190* 清华大学深圳国际研究生院。电子邮箱：{*yinf20*、*y-feng18*}*@mails.tsinghua.edu.cn*、*yang.yujiu@sz.tsinghua.edu.cn*。
• *Baoyuan Wu* 任职于香港中文大学（深圳）数据科学学院、深圳大数据研究院，中国深圳，邮编 *518172*。电子邮箱：*wubaoyuan@cuhk.edu.cn*。
• *Yong Zhang* 和 *Yanbo Fan* 任职于腾讯人工智能实验室，中国广东省深圳市，邮编 *518057*。电子邮箱：{*zhangyong201303, fanyanbo0124*}*@gmail.com*。
• *Jingyi Zhang* 任职于未来媒体中心和计算机科学与工程学院，电子科技大学，中国成都，邮编 *610056*。电子邮箱：*jingyi.zhang1995@gmail.com*。
• *Fei Yin*、*Yong Zhang* 和 *Baoyuan Wu* 为共同第一作者。*Baoyuan Wu* 和 *Yujiu Yang* 为通讯作者。本文已被 *T-PAMI 2022* 接收。本研究部分得到国家自然科学基金委员会项目（编号：*U1903213*）以及深圳市海洋智能感知与计算重点实验室项目（编号：*ZDSYS20200811142605016*）的资助。*Baoyuan Wu* 的研究工作得到国家自然科学基金项目（编号：*62076213*）、深圳市科技计划项目（编号：*RCYX20210609103057050* 和 *ZDSYS20211021111415025*）、香港中文大学（深圳）大学发展基金项目（编号：*01001810*）以及 *CCF*-腾讯开放基金的资助。

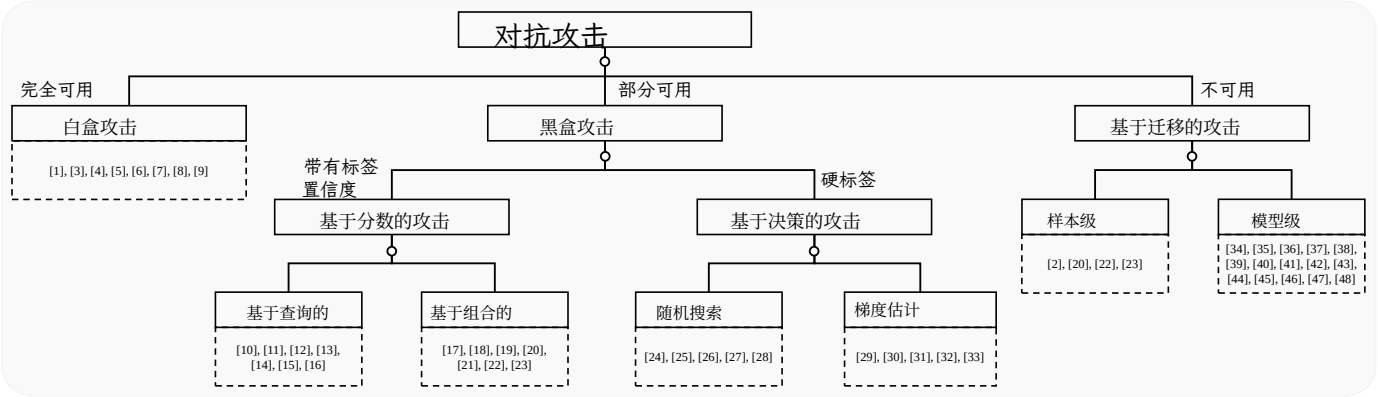


图 1. 现有黑盒对抗攻击方法的分类结构。

被明确提出用于提升攻击性能，尤其是在黑盒攻击场景中。为捕捉上述直觉，我们在此提出利用元学习，在不同攻击任务（$i.e.$，即攻击不同的良性样本）之间学习一个元生成器，称为元条件生成器（*MCG*）。它将先验知识编码到网络参数中，并能据此根据良性样本生成对抗扰动。攻击新的良性样本时，元生成器可以根据该良性样本的信息以及少量历史攻击的反馈信息快速微调，从而生成适用于该新良性样本的有效扰动。

然而，在元训练过程中，直接使用成功攻击的扰动来训练元生成器，仍可能需要对目标模型进行大量查询。为进一步减少查询次数，我们采用广泛使用的**模型级对抗可迁移性**，该方法假设某些共享特征可以从白盒替代模型迁移到目标模型，从而辅助黑盒攻击。具体而言，我们提出基于白盒替代模型进行元训练，再将训练得到的生成器迁移过来，以辅助攻击目标模型，这一过程即为元测试 流程。此外，为了减轻替代模型与目标模型之间的差异，我们还在元测试过程中，通过最小化替代模型与目标模型对同一查询的反馈差异，对替代模型进行微调，从而促使替代模型模仿目标模型的行为，使生成的扰动更适用于目标模型。随后，我们利用更新后的替代模型微调元生成器，以实现适应。简而言之，受元学习视角的启发，我们提出了一种用于黑盒对抗攻击的通用元学习框架，并利用了两种层次的对抗可迁移性：样本级和模型级。

所提出框架的一个突出优势是，它可以自然地与任何现成的黑盒攻击方法相结合，从而提升这些方法原有的性能。例如，对于基于采样的方法，元生成器可以充当采样分布；对于逐步调整扰动的基于随机搜索的方法，元生成器可以提供经过适当初始化的扰动。在基准数据集上开展的大量实验，以及针对多种最先进攻击方法进行的实验，都验证了所提出攻击框架的优越性。

本文的主要贡献有三点。**1)** 我们提出将针对每个良性样本的黑盒攻击视为一个独立任务，这启发我们利用不同任务之间的信息来提升每个任务的攻击性能。**2)**我们利用对抗迁移性的样本级和模型级特征，为黑盒攻击场景开发了一个通用的元学习框架。**3)** 大量实验表明，所提出的框架可以自然地与任何现有的基于查询的黑盒攻击方法相结合，并显著提升其原有性能。

2 相关工作

如图 1 所示，我们首先将现有对抗攻击方法划分为三类：**白盒方法** 利用受攻击模型的完整信息（$i.e.$、参数）生成扰动；**黑盒方法**利用受攻击模型返回的查询反馈；**基于迁移的方法**则不利用受攻击模型的任何信息（$i.e.$、目标模型）。下文各小节将对它们进行详细回顾。

2.1 白盒对抗攻击

白盒攻击问题通常表述如下:

$$\max_{\boldsymbol{\delta}\in\mathbb{B}_\epsilon(\boldsymbol{x})} \mathcal{L}(f(\boldsymbol{x}+\boldsymbol{\delta}),t), \tag{1}$$

其中， $\boldsymbol{\delta}$ 表示对抗扰动， $\mathbb{B}_\epsilon(\boldsymbol{x}) = \{\boldsymbol{x}' - \boldsymbol{x}| \ \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \epsilon\}$ 表示围绕 $\boldsymbol{x}$的邻域，攻击者指定的标量 $\epsilon > 0$ 表示允许扰动的上限。损失函数 $\mathcal{L}(f(\boldsymbol{x}+\boldsymbol{\delta}) \ t)$ 可以设为无目标攻击中的交叉熵损失，其中 t表示 $\boldsymbol{x}$的真实标签；也可以设为有目标攻击中的负交叉熵，其中 t 表示与 $\boldsymbol{x}$ 的真实标签不同的目标标签。许多基于梯度的优化方法已被用于解决上述问题，包括快速梯度符号法（FSGM） [1], 迭代快速梯度符号法（I-FGSM） [3], C&W方法 [4],功能性对抗攻击 [5],对抗伪装 [6], etc。此外，一些研究 [7], [8], [9] 关注物理世界中的对抗攻击，在这种情况下，多种环境失真可能会削弱生成的对抗扰动的有效性。

2.2 黑盒对抗攻击

问题（1）的表述也适用于黑盒攻击。然而，目标模型的参数 $f(\cdot)$ 是

未知，而对于每次查询，仅提供目标值 $f(\boldsymbol{x}+\boldsymbol{\delta})\,\boldsymbol{x}+\boldsymbol{\delta}$。因此，无法直接使用基于梯度的优化方法。根据查询反馈的类型 $f(\boldsymbol{x}+\boldsymbol{\delta})$，黑盒攻击还可进一步分为两个子类别：基于分数的攻击和基于决策的攻击。

2.2.1 基于分数的黑盒对抗攻击

在基于分数的攻击场景中，反馈 f（$\boldsymbol{x}+\boldsymbol{\delta}$）是连续的，例如图像分类任务中的后验概率。解决这一问题的现有方法通常可分为三类，包括基于查询的方法和基于组合的方法。**1）基于查询的方法**仅根据对目标模型的查询迭代调整扰动。研究中采用了多种黑盒优化方法，主要包括随机搜索（$e.g.$、[10]、Square、SignHunter[11]）、进化策略（$e.g.$、NES [12],、Bandit [13]）以及梯度估计方法（$e.g.$、ZOO [14],、AutoZOOM [15],、ZO-signSGD [16]）。与基于迁移的方法相比，基于查询的方法通常能取得更高的攻击成功率，但代价是需要对目标模型进行大量查询。**2）基于组合的方法**旨在同时实现较高的攻击成功率和查询效率，充分利用对目标模型的查询以及从替代模型迁移而来的信息。现有方法提出了不同类型的迁移策略，例如迁移扰动幅度（$e.g.$、Square [17]）扰动梯度（$e.g.$、 [18]、 [19] 和 Meta 攻击 [20]）、扰动分布（$e.g.$、 $\mathcal{N}$ATTACK [21],AdvFlow [22]）以及向低维空间的投影（$e.g.$、TREMBA [23]）。基于组合的方法在攻击性能方面优于另外两类方法，我们的方法也属于这一类。然而，现有方法大多（Meta 攻击 [20]除外）只利用了模型级对抗可迁移性，而我们的方法同时捕捉样本级和模型级迁移性，从而进一步提升查询效率。

2.2.2 基于决策的黑盒对抗攻击

在基于决策的攻击场景中，反馈 f（$\boldsymbol{x}+\boldsymbol{\delta}$）是离散的，例如图像分类任务中的类别标签。现有解决这一问题的方法大体可分为基于随机搜索的方法和基于梯度估计的方法。**基于随机搜索的方法**旨在不可见的决策边界附近寻找最佳扰动，例如在 Boundary方法中从正态分布采样 [24]，在 Evolutionary方法中从可学习的高斯分布采样 [25], ，在 GeoDA 中沿估计出的决策边界法向搜索 [26], ，或在 SFA [27] 和 Rays [28] 中，在良性样本附近的允许扰动区域表面上搜索。**基于梯度估计的方法**提出了不同的梯度估计方法，例如在 NES [12] 和 qFool [29]，中利用当前解附近的点，在 HopSkipJumpAttack [30], 中采用蒙特卡洛估计，或在 QEBA [31] 中于低维子空间内估计梯度以加速计算。OPT [32] 和 Sign-OPT [33] 基于连续形式开发，通过交替优化扰动的幅度和方向，使得任何梯度估计方法都可以应用。

2.3 基于迁移的对抗攻击

我们将基于迁移的方法列为另一类别，因为每种基于迁移的方法都可用于白盒攻击或黑盒攻击。例如，(MI-FGSM) [34] 和 Nesterov 迭代快速梯度符号法 (NI-FGSM) [35] 可用作白盒攻击，因为它们是经典白盒攻击 FSGM [1] 和 I-FGSM [3]的扩展。然而，正如文稿 [34] 和 [35], 中所述，它们的目标是在黑盒设置下生成更具可迁移性的扰动，以提高攻击成功率，并且其实验涵盖了白盒和黑盒两种设置。因此，如果一种方法主要旨在提高对抗迁移性，我们就将其划分到基于迁移的方法类别，而非白盒或黑盒类别。根据迁移的目标，我们进一步分别介绍样本级和模型级可迁移性。

2.3.1 样本级对抗迁移性

尽管没有明确说明，一些研究实际上已经探讨了样本级迁移性。例如，通用对抗扰动 [2] 表明，针对同一个被攻击模型，可以找到一种扰动，同时欺骗多个良性样本。这说明不同的良性样本可能存在一些共同的脆弱方向，沿着这些方向可以轻易找到对抗扰动。另一个例子是基于生成的对抗攻击 [23], [22], ，这类方法训练生成模型，直接为每个良性样本生成对抗扰动或对抗样本。其默认假设是，不同良性样本周围的对抗扰动可能服从相同的分布或映射。然而，样本级对抗迁移性很少被用于提升攻击性能，尤其是在黑盒攻击场景中。我们发现唯一的尝试称为 Meta 攻击 [20],，它训练元攻击者生成梯度，并将其用作基于梯度估计的黑盒攻击方法中的更新方向。相比之下，我们提出的元学习框架旨在捕获对抗扰动的分布，因此可以自然地与各种基于查询的攻击方法相结合。

2.3.2 模型级对抗可迁移性

模型级对抗可迁移性已在许多现有工作中得到观察和研究 [36], [37], [23], [38], [39],[40], [41]。关于模型级迁移性，研究主要探讨了两个重要问题：其内在原因，以及如何增强不同模型之间的迁移性，尤其是在黑盒场景中。**1）模型级迁移性的内在原因**。Tram`er 等人 [40] 发现，不同模型共享了很大一部分对抗子空间，该子空间由正交基向量构成，且这些向量与损失函数梯度高度对齐。因此，位于这一共享子空间内的扰动很可能在不同模型之间具有可迁移性。 [37] 从几何角度指出，可迁移性部分源于不同模型的决策边界彼此高度对齐。Demontis 等人 [42]证明，模型级迁移性与目标模型的内在脆弱性以及替代模型的复杂度密切相关。 [41]中提供的理论分析基于数据分布相似性和模型梯度相似性推导出了可迁移性的两个下界，并基于梯度正交性和平滑性推导出了可迁移性的上界。Wang等人 [43] 表明，模型级迁移性与对抗扰动内部的相互作用呈负相关。**2）增强模型级迁移性**。与 FGSM [1], 相比，其扩展方法，包括动量迭代快速

梯度符号法（MI-FGSM） [34] 和 Nesterov 迭代快速梯度符号法（NI-FGSM） [35], 表现出更好的模型级迁移性。 [44], 进一步扩展了 MI-FGSM 和 NI-FGSM，通过降低迭代更新方向的方差，使更新方向更加稳定，从而逃离较差的局部最优解；即使目标模型采用了防御措施，也能提升可迁移性。集成攻击方法 [37]基于多个模型生成对抗扰动，以增强无目标攻击和有目标攻击的可迁移性。中间层攻击（ILA） [45] 提出基于替代模型的中间层生成对抗扰动，以避免过拟合，从而增强对目标模型的可迁移性。特征分布攻击（FDA） [46] 着重通过最大化良性样本与扰动样本之间某个中间层的激活值，来提升有目标攻击的可迁移性。[38] 将其从单个中间层扩展到多个中间层，从而进一步增强了有目标攻击的可迁移性。一些研究也采用了元学习思想来增强可迁移性。MSM [47] 通过优化可微攻击器获得元替代模型。该元替代模型从一个或一组替代模型中获取先验知识，能够生成具有出色可迁移性的对抗样本。元梯度 [48] 从模型库中随机抽取多个模型以构成不同任务，并在每个任务中迭代模拟白盒攻击和黑盒攻击，从而缩小白盒攻击与黑盒攻击之间的梯度方向差距。

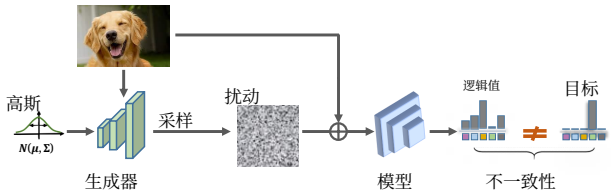


图 2. 元学习中的任务定义。给定一张图像和一个目标模型，目标是学习一个生成器，使其能够生成有效的对抗样本来攻击目标模型。

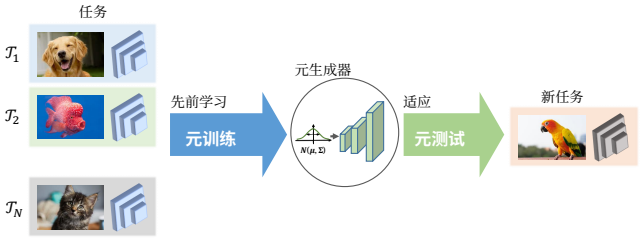


图 3. 元学习概览。在元训练阶段，通过在大量任务上进行训练，可以得到一个元生成器，其中包含如何为不同图像生成有效扰动以攻击目标模型的通用先验知识。在元测试阶段，只需经过少量微调步骤，元生成器便可快速适应新任务。

3 所提出的方法

3.1 问题表述

我们将分类模型记为 $f_{\boldsymbol{\theta}}$： $\mathcal{X} \to \mathcal{Y}$，其中在黑盒攻击设置下，模型参数 $\boldsymbol{\theta}$ 未知， $\mathcal{X}$ 和 $\mathcal{Y}$ 分别表示输入空间和输出空间。给定一个输入样本 $\boldsymbol{x}$，其真实标签的索引记为y； $f_{\boldsymbol{\theta}}(\boldsymbol{x}, i) \in [0,1]$ 表示第 i类的后验概率， $f_{\boldsymbol{\theta}}(\boldsymbol{x}, i)$ 表示对应的逻辑值。我们的攻击目标是为良性样本$\boldsymbol{x}$ 生成对抗扰动 $\boldsymbol{\delta}$，以欺骗模型 $f_{\boldsymbol{\theta}}(\cdot)$。由于模型参数 $\boldsymbol{\theta}$ 未知，这种攻击称为基于分数的黑盒对抗攻击

。它通常可以表述为以下优化问题：

$$\min_{\boldsymbol{\delta} \in \mathbb{B}_\epsilon(\boldsymbol{x})} \mathcal{L}^{\text{adv}}(\boldsymbol{\delta}, \boldsymbol{x}, y) = \begin{cases} \max(0, \triangle_{ut}), & \text{if } \textit{untargeted attack} \\ \max(0, \triangle_t), & \text{if } \textit{targeted attack} \end{cases} \tag{2}$$

其中 $\triangle_{ut} = f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}, y) - \text{最大}_{j \neq y} f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}, j)$，且 $\triangle_t = \text{最大}_{j \neq y} f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}, j) - f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta}, t)$，其中 $t \in \mathcal{Y}$为目标标签。注意， $\mathcal{L}^{\text{adv}}$ 为非负，如果达到0，则相应的 $\boldsymbol{\delta}$ 就是成功的对抗扰动。

3.2 元　学习条件生成器：黑盒　攻击

条件扰动生成器。与大多数先前基于组合的方法不同，它们使用确定性网络为良性图像预测初始扰动；我们的生成器会捕获以良性图像为条件的扰动条件分布，即 $p(\boldsymbol{\delta}|\boldsymbol{x};\boldsymbol{\varphi})$，其中 $\boldsymbol{\delta} = G_{\boldsymbol{\varphi}}(\boldsymbol{z};\boldsymbol{x})$。 G 是生成器， $\boldsymbol{\varphi}$ 是其参数。 $\boldsymbol{\delta}$ 是扰动， $\boldsymbol{z}$ 是服从简单分布（例如高斯分布）的随机向量。在该条件分布中，有效扰动被期望具有较高的采样概率。攻击目标模型时，我们可以采样一个扰动 $\boldsymbol{\delta} \sim p(\boldsymbol{\delta}|\boldsymbol{x};\boldsymbol{\varphi})$，并将其添加到良性图像中，以构造对抗样本来欺骗目标模型。

条件分布建模。生成器捕获扰动的条件分布，这可以通过使用一个简单分布和一个复杂的非线性函数来实现，该函数以图像为条件，将简单分布映射到复杂分布。在这项工作中，我们使用条件生成流（c-Glow） [49] 作为生成器，因为它具有出色的性质：随机向量与输出扰动之间的映射是可逆的。使用 c-Glow 后，我们有 $\boldsymbol{\delta} = g_{\boldsymbol{\varphi}}(\boldsymbol{z};\ \boldsymbol{x})$ 及其逆映射 $\boldsymbol{z} = g_{\boldsymbol{\varphi}}^{-1}(\boldsymbol{\delta};\ \boldsymbol{x})$。随机向量 $\boldsymbol{z}$ 服从高斯分布，即 即 $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu},\ \boldsymbol{\Sigma})$。由于 c-Glow 由一组可逆函数/层构成，参数 $\boldsymbol{\varphi}$ 可以分解为若干独立部分，即 即$g = g_{\boldsymbol{x},\ \varphi_1} \circ \cdots \circ g_{\boldsymbol{x},\ \varphi_M}$。 M 表示层数， φ_i 表示第 i 层的参数。利用变量替换 [50], ，我们可以将条件似然写为

$$\log p(\boldsymbol{\delta}|\boldsymbol{x};\boldsymbol{\varphi}) = \log p(\boldsymbol{z}) + \sum_{i=1}^{M} \log \left| \det\left(\frac{\partial g_{\varphi_i}^{-1}(\boldsymbol{r}_{i-1};\boldsymbol{x})}{\partial \boldsymbol{r}_{i-1}}\right) \right|, \tag{3}$$

其中 $\boldsymbol{r}_i = g_{\varphi_i}(\boldsymbol{r}_{i-1};\ \boldsymbol{x})$ 、 $\boldsymbol{r}_0 = \boldsymbol{x}$ 和 $\boldsymbol{r}_M = \boldsymbol{z}$。

元学习视角。 对一张良性图像发起攻击可视为一个独立过程：生成器从条件分布中采样若干扰动，并从目标模型获取相应的反馈分数作为监督信号，以此进行优化。然而，攻击黑盒模型时，查询预算总是有限的，即 我们只能采样少量扰动。因此，每个攻击过程中的生成器学习可以表述为一个，即 少样本学习问题：给定 $\{\boldsymbol{\delta}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\ y_i\}_{i=1}^K$，目标是学习生成器 $G_{\boldsymbol{\varphi}}(\boldsymbol{z};\ \boldsymbol{x})$。 K 是样本数。

如第 1 节所述，不同良性图像周围的对抗扰动可能具有某些共同属性，即，

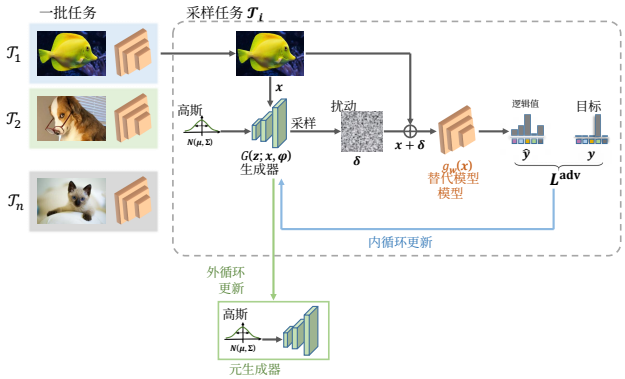


图 4. 元训练流程。元训练采用批量版 REPTILE。我们对一批任务进行采样，并对每个任务执行任务特定参数的内循环更新。然后，对批次中所有任务的任务特定参数进行聚合，以执行外循环更新，即 更新元参数。

样本级对抗迁移性。受元学习中“学会学习”这一概念的启发，我们可以从元学习的角度解决少样本学习问题，并通过执行大量攻击任务来学习一个元生成器，以捕捉扰动的共同属性。通过各种各样的任务，学习如何围绕良性图像生成扰动条件分布的先验知识。由于不同良性图像的扰动分布由同一个生成器生成，因此该先验知识被隐式编码在生成器的参数中。

任务定义。 任务是元学习中的基本单位。在对抗攻击场景中，“任务”定义为：“给定一张良性图像和一个目标模型，目标是学习一个条件生成模型，使得从中采样出的扰动能够成功欺骗目标模型。”如图 2 所示，给定一张良性图像，我们可以从生成模型中采样出一个扰动。将良性图像与扰动相加，得到一个对抗样本，然后将其输入目标网络进行攻击。如果对抗样本的预测标签与真实标签不一致（即，无目标攻击），或者对抗样本的预测标签与指定标签相同（即 有目标攻击），那么该对抗样本就是有效的。因此，通过最大化预测结果与真实标签之间的差异，或最小化预测结果与指定标签之间的差异，来更新生成器的参数。

由于目标模型的参数和架构未知，受对抗样本可迁移性的启发， [36],我们在元训练阶段使用替代模型代替目标模型。如图 3 所示，通过执行大量任务，我们可以获得一个元生成器，它能够捕捉生成对抗扰动的先验知识，并在元测试阶段适应不同的任务。

3.3 使用替代模型进行元训练

如第 3.2 节所述，在任务 $\mathcal{T}$ 中，给定一张良性图像 $\boldsymbol{x}$ 和一个目标模型 $f_{\boldsymbol{\theta}}(\boldsymbol{x},\cdot)$，目标是学习一个条件扰动生成器 $G_{\boldsymbol{\varphi}}(\boldsymbol{z};\boldsymbol{x})$，以生成有效扰动 $\boldsymbol{\delta}$ 来欺骗目标模型，例如，无目标攻击采用 $f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta},y) < \max_{j\neq y} f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta},j)$，有目标攻击采用 $f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta},t) > \max_{j\neq t} f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta},j)$，其中 $\boldsymbol{\delta}=G_{\boldsymbol{\varphi}}(\boldsymbol{z};\boldsymbol{x})$。元训练过程如图 4 所示。在这项工作中，不同任务使用相同的目标黑盒模型。

在黑盒攻击场景中，我们无法访问目标模型的参数和架构。因此，我们必须针对每张良性图像进行大量查询，以生成成功扰动，再用这些扰动学习元生成器；这会耗费大量查询成本，而且难以在现实场景中实现。基于模型级对抗可迁移性，在元训练阶段，我们用替代模型$g_{\boldsymbol{w}}$（$\boldsymbol{x}$）取代目标模型，该替代模型的架构和参数均可获取。因此，梯度可以通过替代模型反向传播，以更新生成器。由于元训练阶段不涉及任何目标模型，元训练完成后，元生成器便可在元测试阶段应用于任意目标模型。

为了评估生成的扰动，其对抗损失函数与公式（2）中的类似。区别~ ~在于， $\triangle_{ut}$ 和 $\triangle_t$ 是使用替代模型而非目标模型计算的，即

$$\tilde{\triangle}_{ut} = g_{\boldsymbol{w}}(\boldsymbol{x}+\boldsymbol{\delta},y) - \max_{j\neq y} g_{\boldsymbol{w}}(\boldsymbol{x}+\boldsymbol{\delta},j),$$
$$\tilde{\triangle}_{t} = \max_{j\neq y} g_{\boldsymbol{w}}(\boldsymbol{x}+\boldsymbol{\delta},j) - g_{\boldsymbol{w}}(\boldsymbol{x}+\boldsymbol{\delta},t), \tag{4}$$

其中， t 是有目标攻击中指定的目标类别。

给定一组任务 $\{\mathcal{T}_i\}_{i=1}^{N}$，我们采用 REPTILE [51] 的批量版本进行元学习。我们采样 n 个任务组成一个批次，并使用 Adam [52] 为每个任务更新任务特定参数 k 次。内循环优化的目标是 $\boldsymbol{\phi}_{\mathcal{T}_i} = \text{argmin}_{\boldsymbol{\phi}_{\mathcal{T}_i}} \mathcal{L}^{\text{adv}}_{\mathcal{T}_i}$。优化过程可表示为

$$\boldsymbol{\phi}_{\mathcal{T}_i} = \text{Adam}(\mathcal{L}^{\text{adv}}_{\mathcal{T}_i}, \boldsymbol{\varphi}, k, \alpha), \tag{5}$$

其中， $\boldsymbol{\phi}_{\mathcal{T}_i}$ 是生成器从 $\boldsymbol{\varphi}$开始执行 Adam k 步后的最终任务特定参数。在每一步（共 k 步）中，都会从当前条件分布中采样一个扰动。 $\mathcal{L}^{\text{adv}}_{\mathcal{T}_i}$是第 i个任务的对抗损失。 α 是内循环的学习率。

然后，对于外循环优化，我们可以利用一个小批次中得到的任务特定参数来更新生成器的元参数，即

$$\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} + \beta \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{\phi}_{\mathcal{T}_i} - \boldsymbol{\varphi}), \tag{6}$$

其中， β 是外循环的学习率。

3.4 使用历史攻击经验进行元测试

标准的元测试流程不适用于黑盒攻击，因为目标模型未知，且无法通过该模型反向传播梯度以微调元生成器。为了解决元测试中的这一问题，我们提出利用先前攻击的反馈来微调替代模型，从而将黑盒目标模型的信息迁移到替代模型中。利用历史攻击经验可以使替代模型模仿目标网络的行为，与直接使用替代模型相比，这能为微调元生成器提供更准确、更有效的信息。元测试流程如图 5 所示。给定一张新的良性图像，对目标模型进行攻击包括三个步骤：即，微调替代模型以引入目标模型的信息，微调元生成器以适应新的良性图像，以及增强现成的黑盒攻击方法。整个框架的攻击能力在循环过程中逐步提升。

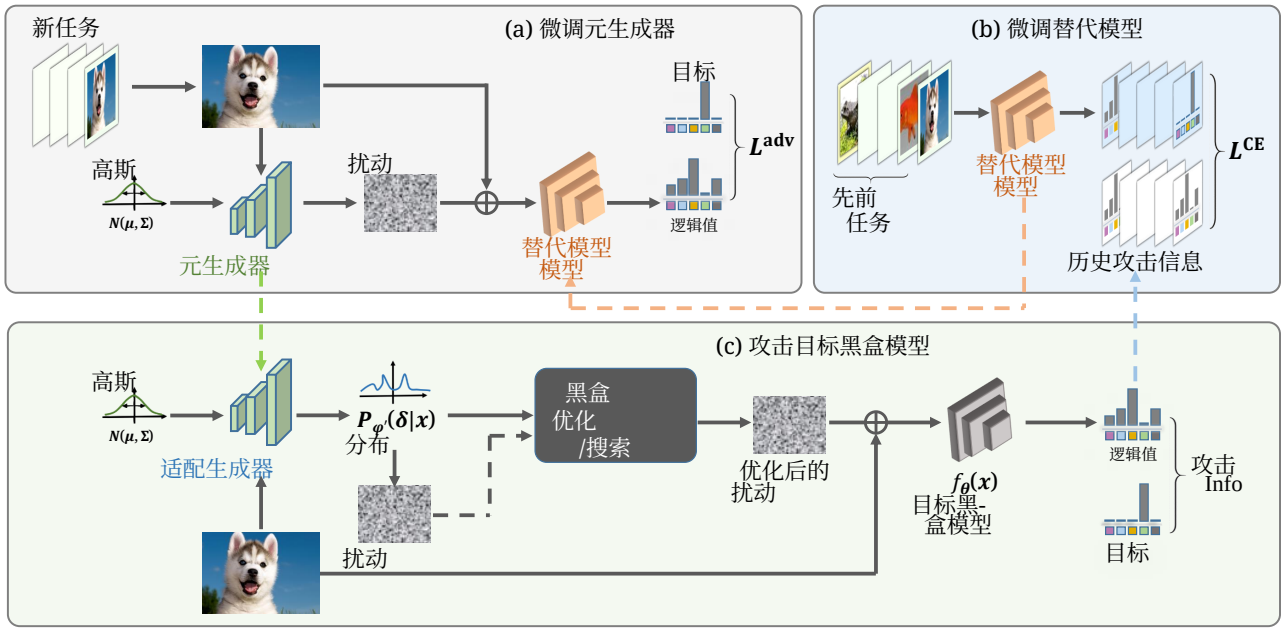


图 5. 元测试流程。(a) 微调元生成器。给定新任务的图像和更新后的替代模型，通过使用 $\mathcal{L}^{adv}$ 执行内层优化来微调元生成器。(b) 微调替代模型。为了将目标黑盒模型的信息迁移到替代模型，使用先前任务以及当前任务的历史信息（即，对抗样本的逻辑值和目标标签），使替代模型模仿目标模型的行为。(c) 攻击目标黑盒模型。适配生成器与现成的黑盒攻击方法相结合。生成器可根据给定图像提供扰动的初始分布或采样扰动。攻击方法利用该初始化作为初始状态，以获得精炼扰动。随后，使用生成的对抗样本攻击目标模型。记录攻击的逻辑值，以便在阶段 (b) 微调替代模型。

微调替代模型。 如图 5(b) 所示，为了利用历史攻击经验，我们收集目标模型对先前良性图像、其对抗样本以及当前良性图像的预测逻辑值，为替代模型的微调提供监督信号。损失函数定义为

$$\begin{aligned}\mathcal{L}^{\mathrm{CE}} = \; & CE(g_{\boldsymbol{w}}(\boldsymbol{x}+\boldsymbol{\delta}), f_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\delta})) \\ & + CE(g_{\boldsymbol{w}}(\boldsymbol{x}), f_{\boldsymbol{\theta}}(\boldsymbol{x}))\end{aligned} \tag{7}$$

其中 $CE(\cdot)$ 是交叉熵损失函数。这两项分别表示对抗样本和良性图像的损失。对于当前任务中的良性图像，仅使用第二项。替代模型参数的优化可表示为

$$\boldsymbol{w}' = \mathrm{Adam}(\sum_{i}^{m} \mathcal{L}_{\mathcal{T}_i}^{\mathrm{CE}}, \boldsymbol{w}, s, \lambda), \tag{8}$$

其中， $\boldsymbol{w}'$ 表示更新后的替代模型的参数。λ 是学习率， m 是一个小批次中的任务数。$\mathcal{L}_{\mathcal{T}_i}^{\mathrm{CE}}$ 是第 i 个任务的损失。 s 是更新步数。

微调元生成器。图 5(a) 展示了在新任务的良性图像上，使用更新后的替代模型 $g_{\boldsymbol{w}'}(\boldsymbol{x})$ 对元生成器进行微调。元测试阶段的微调过程与元训练阶段的内层优化类似。我们使用式（4）和式（5）中的损失来更新元生成器的参数，即，

$$\boldsymbol{\phi}_{\mathcal{T}} = \mathrm{Adam}(\mathcal{L}_{\mathcal{T}}^{\mathrm{adv}}, \boldsymbol{\varphi}, k, \alpha), \tag{9}$$

其中， $\boldsymbol{\phi}_{\mathcal{T}}$ 表示适配新任务后的参数。与式（5）不同，式（9）使用更新后的替代模型生成对抗样本的逻辑值，即 $g_{\boldsymbol{w}'}(\boldsymbol{x}+\boldsymbol{\delta})$ 。由于更新后的替代模型包含从目标模型迁移而来的信息，因此它能提供更准确、有效的监督信号，以微调元生成器。相比原始替代模型，生成的扰动因此更针对目标模型。

增强现成的黑盒攻击方法。 我们的适配生成器可以提供扰动的初始分布，或提供以给定良性图像为条件的扰动，使其能够与其他现成的黑盒攻击方法结合，从而提升这些方法原有的性能。图 5 (c) 展示了攻击目标模型的过程。黑盒攻击方法将初始分布或扰动用作初始状态。随后，将进一步优化后的扰动添加到良性图像中，以生成对抗样本。目标模型输出的逻辑值会被记录为历史攻击信息，并用于微调替代模型。与基于采样的方法 [12], [53], 结合时，适配生成器用作分布。与基于随机搜索的方法 [10], [11], [17], 结合时，可以从生成器中采样得到扰动，并将其用作初始状态。

4 实验

4.1 实验设置

数据集。为展示所提出方法的有效性，我们在两个常用的基准数据库上开展了全面的，即 CIFAR-10 [54] 和 ImageNet [55]实验。遵循 [53]，中针对 CIFAR-10 数据集的设置，我们从测试集中随机选取1,000 张图像进行评估，并确保涵盖所有类别且各类别数量均衡。图像被调整为 32×32。对于 ImageNet 数据集，我们首先从 10 个类别中随机选取

来自1,000 个类别，然后从验证集中选取每个类别的500 张图像进行评估。图像被调整为224×224。目标模型和替代模型均在相应数据集的训练集上进行训练。在 CIFAR-10 上，我们使用完整的训练集进行元学习，以学习元生成器。在 ImageNet 上，所选 10 个类别的训练集用于元学习。

评估。我们选用基于 l_∞ 的攻击，并将 CIFAR-10 和 ImageNet 的最大扰动分别设为 $\epsilon = 0.031$ 和 $\epsilon = 0.05$，图像像素值重新缩放至 $[0, 1]$。所有实验的最大查询预算均设为 10,000 次。如果攻击者无法在查询上限内成功欺骗目标模型，则将其视为失败案例。遵循先前工作 [53],，我们采用攻击成功率（ASR）、成功攻击的平均查询次数（Mean）和中位查询次数（Median）来评估攻击性能。

目标模型和替代模型。在 CIFAR-10 上，我们考虑四种目标模型: ResNet-Preact-110 [56], 、DenseNet-121 [57], 、VGG-19 [58], 和 PyramidNet-110 [59]。我们遵循图像分类的标准训练流程，获取这些目标模型的检查点。这四种目标模型在标准测试集上的 Top-1 错误率分别为 6.29% 、 6.17% 、 7.28% 和 7.51%。在 ImageNet 上，我们还在四种目标模型上评估了我们的方法: ResNet-18 [56], 、VGG-16-BN [58], 、WRN-50 [60], 和 InceptionV3 [61]。我们使用这些方法的官方实现，并从 torchvision 下载其预训练检查点。这些目标模型在 I mageNet 验证集上的 Top-1 错误率分别为 28.41%、30.24%、21.53% 和 22.71%。在所有实验中，ResNet-18 [56] 和 ResNet-50 [56] 分别用作 CIFAR-10 和 ImageNet 上的替代模型。相应替代模型的 Top-1 错误率分别为 ResNet-18 的 6.37% 和 ResNet-50 的 23.97%。

为进一步验证我们框架的性能，我们还在 ImageNet 上对黑盒对抗防御模型进行了攻击实验，包括 JPEG压缩-WideResNet-50 [62],小噪声防御-WideResNet-50 [63], FreeAdv-ResNet-50 [64], 和 FastAdv-ResNet-50 [65]。

竞争方法。 由于我们的框架能够为扰动提供良好的初始化，因此可将其视为即插即用组件，与其他黑盒攻击方法结合以提升其性能。为验证模型的通用性，我们将框架与 6 种基于查询的黑盒攻击方法结合，包括 NES [12], CG-Attack [53], SimBA [10], Signhunter [11], Square [17], 和 MetaAttack [20]。对于 Signhunter、Square、SimBA 和 MetaAttack 等基于搜索的方法，生成器生成的采样扰动用作初始化。对于 NES 和 CG-Attack 等基于采样的方法，生成器表示的扰动分布用作初始化。

我们还将所提出的方法与几种基于迁移的攻击方法进行比较，以验证其可迁移性，包括 PGD [66],MI [34],TIMI [67],和 DI [68],，这些方法采用迁移攻击设置，仅能获取替代模型的信息。此外，我们最后还在查询攻击设置下与几种组合式方法进行比较，包括 Ad- vFlow [22] 和 TREMBA [23]。这些方法同时利用对目标模型的查询以及从替代模型迁移来的项目。为确保比较公平，我们使用与本文相同的替代模型重新训练了这些组合式方法。所有实验均在相同设置下使用作者提供的源代码实现。

实现细节。在所有实验中，我们都采用相同的 c-Glow 生成器架构，其中包含 3 个模块，每个模块由 8 个流步骤组成。每个模块以挤压操作开始，随后执行 8 个流步骤，并以分割操作结束。为提高效率，我们采用离散余弦变换（DCT）和逆离散余弦变换进行降维，在将图像输入生成器之前，对 ImageNet 图像进行下采样，将其尺寸缩小至 $\frac{1}{8} \times \frac{1}{8}$ 较低频率子空间。对于 CIFAR-10，由于图像尺寸较小，我们使用其原始形状。

在元训练之前，我们对 c-Glow 进行预训练，使其初步建模扰动分布。我们使用替代模型，通过 PGD攻击生成大量扰动，扰动强度为 $\ell_{\inf} = 0.05$，步长为 0.01，迭代次数为 50。通过最大化对数似然来优化参数，即最大化$\boldsymbol{\varphi}$ 对数 $p(\boldsymbol{\delta}|\boldsymbol{x}; \boldsymbol{\varphi})$。

生成器预训练的学习率设为0.001。批量大小为 $m = 16$。生成器训练 10 个周期。元训练时，每个批次采样 16 个任务。内层优化的更新步长设为 $k = 4$。内循环和外循环的学习率分别设为 $\alpha = 0.0003$和 $\beta = 0.0006$。元测试时，微调替代模型时，我们冻结除最后三层外所有层的参数。替代模型使用良性图像及其对抗样本进行微调。微调的学习率设为 $\lambda = 0.0003$。每个批次中的良性图像数量设为 4。微调元生成器时，该过程与任务特定参数的内循环优化类似。

4.2 闭集攻击场景中的实验

在闭集攻击场景中，替代模型和目标模型使用同一训练集进行训练，即，训练图像和类别均相同。实验包括提升现成的黑盒攻击方法、攻击防御模型、与基于迁移的方法进行比较，以及与基于组合的方法进行比较。

CIFAR-10 上的性能。第 4.1 节介绍了替代模型和目标模型的具体设置。表 1 展示了几种现成的黑盒攻击方法及其与我们提出的 MCG 结合后的结果。

无论是无目标攻击还是有目标攻击，MCG 都能提升所有黑盒攻击方法的攻击效率，同时不会降低攻击成功率。具体而言，在无目标攻击设置下，利用 MCG 提供的初始化，对所有目标模型，这些方法的中位查询次数均降至1次；这意味着，对于超过50% 的图像，我们可以利用初始生成的扰动欺骗目标模型。同时，几乎所有情况下的攻击成功率都提升至接近100% 。此外，查询次数均值也显著改善。例如，对于最先进的Square 攻击，针对全部四个目标模型，我们的 MCG 都能将其平均查询次数降低至原来的五分之一，从而进一步提升查询效率。我们还在图 6 中绘制了攻击成功率相对于查询次数的趋势曲线。可以观察到，对于所有参与比较的攻击方法，在所有查询次数下，MCG 都能提升攻击性能，尤其是在查询次数较少时。

在有目标攻击设定下，我们提出的 MCG 也能提升攻击性能。有目标攻击通常

表 1 CIFAR-10 数据集上的闭集评估。

目标模型 → / 攻击方法 ↓	ResNet-PreAct-110			DenseNet-121			VGG-19			PyramidNet-110		
	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数
无目标攻击												
NES [12]	**100.0%**	285.2	211.0	99.4%	430.8	274.0	99.1%	822.8	421.0	**100.0%**	287.9	190.0
MCG +NES	**100.0%**	**124.1**	**1.0**	**100.0%**	**133.2**	**1.0**	**99.9%**	**371.7**	**1.0**	**100.0%**	**90.4**	**1.0**
CG-Attack [53]	**100.0%**	230.0	81.0	**100.0%**	222.7	21.0	**100.0%**	386.4	101.0	**100.0%**	93.7	**1.0**
MCG +CG-Attack	**100.0%**	**112.3**	**1.0**	**100.0%**	**113.4**	**1.0**	**100.0%**	**262.7**	**1.0**	**100.0%**	**74.1**	**1.0**
SimBA-DCT [10]	**100.0%**	428.0	359.5	**100.0%**	449.9	352.0	**99.7%**	557.6	426.0	**100.0%**	341.1	273.5
MCG +SimBA-DCT	**100.0%**	**199.3**	**1.0**	**100.0%**	**127.0**	**1.0**	**99.7%**	**234.8**	**1.0**	**100.0%**	**114.6**	**1.0**
Signhunter [11]	**100.0%**	167.0	83.0	**100.0%**	196.6	87.0	**100.0%**	238.3	99.0	**100.0%**	128.4	63.5
MCG +Signhunter	**100.0%**	**83.3**	**1.0**	**100.0%**	**61.3**	**1.0**	**100.0%**	**157.9**	**1.0**	**100.0%**	**26.9**	**1.0**
Square [17]	**100.0%**	227.3	144.5	**100.0%**	260.3	159.0	**100.0%**	342.0	175.5	**100.0%**	165.5	100.5
MCG +Square	**100.0%**	**39.7**	**1.0**	**100.0%**	**47.6**	**1.0**	**100.0%**	**57.9**	**1.0**	**100.0%**	**29.6**	**1.0**
MetaAttack [20]	**100.0%**	642.6	632.0	99.8%	759.4	635.0	**100.0%**	686.4	633.0	**100.0%**	601.8	430.0
MCG +MetaAttack	**100.0%**	**204.6**	**1.0**	**100.0%**	**81.4**	**1.0**	**100.0%**	**188.6**	**1.0**	**100.0%**	**99.3**	**1.0**
有目标攻击												
NES [12]	**100.0%**	774.3	610.0	99.8%	1205.2	946.0	92.4%	2738.8	1954.0	**100.0%**	889.7	694.0
MCG +NES	**100.0%**	**591.8**	**443.0**	**100.0%**	**1009.2**	**800.0**	**98.5%**	**2688.6**	**1472.0**	**100.0%**	**657.8**	**506.0**
CG-Attack [53]	**100.0%**	884.8	741.0	**99.8%**	859.7	681.0	96.8%	1476.9	901.0	**100.0%**	567.3	501.0
MCG +CG-Attack	**100.0%**	**430.8**	**181.0**	**99.8%**	**608.5**	**241.0**	**97.4%**	**944.1**	**381.0**	**100.0%**	**290.6**	**141.0**
SimBA-DCT [10]	99.6%	836.3	729.0	99.7%	944.8	842.0	98.6%	1170.3	986.0	99.8%	735.7	661.0
MCG +SimBA-DCT	**99.8%**	**664.2**	**623.5**	**99.8%**	**845.5**	**768.00**	**98.9%**	**983.3**	**861.0**	**99.9%**	**601.9**	**543.5**
Signhunter [11]	**100.0%**	386.1	272.0	**100.0%**	465.1	323.0	**100.0%**	556.3	385.5	**100.0%**	320.9	232.0
MCG +Signhunter	**100.0%**	**267.3**	**124.5**	**100.0%**	**321.0**	**181.0**	**100.0%**	**399.1**	**210.0**	**100.0%**	**167.9**	**81.0**
Square [17]	99.6%	504.7	369.0	**100.0%**	624.8	471.0	**100.0%**	827.2	593.5	**100.0%**	400.1	301.0
MCG +Square	**100.0%**	**92.3**	**20.0**	**100.0%**	**133.1**	**33.0**	**100.0%**	**141.7**	**22.0**	**100.0%**	**55.0**	**17.0**
MetaAttack [20]	**100.0%**	1174.7	899.0	**100.0%**	1294.5	1106.0	99.0%	1721.4	1106.0	**100.0%**	1065.9	890.0
MCG +MetaAttack	**100.0%**	**757.3**	**669.0**	**100.0%**	**992.9**	**887.0**	**100.0%**	**1180.7**	**882.0**	**100.0%**	**718.3**	**669.0**

表2 在 ImageNet 数据集上的闭集评估。

目标模型 → / 攻击方法 ↓	ResNet-18			VGG-16			WRN-50			Inception-V3		
	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数
无目标攻击												
NES [12]	99.0%	2085.5	1597.0	98.2%	1554.0	1072.0	96.5%	2366.0	1681.0	95.8%	1111.6	526.0
MCG +NES	**99.7%**	**1057.5**	**1.0**	**99.1%**	**596.5**	**1.0**	**98.4%**	**1385.1**	**1.0**	**96.7%**	**889.9**	**390.5**
CG-Attack [53]	96.3%	272.0	**21.0**	96.5%	178.3	21.0	89.6%	323.0	**21.0**	94.1%	**251.1**	**21.0**
MCG +CG-Attack	**100.0%**	**95.4**	**21.0**	**99.7%**	**69.3**	**1.0**	**99.2%**	**247.3**	**21.0**	**97.0%**	271.1	**21.0**
SimBA-DCT [10]	**100.0%**	520.6	381	98.3%	486.1	350.0	**94.9%**	810.6	593.0	81.9%	496.9	290.5
MCG +SimBA-DCT	98.1%	**68.7**	**1.0**	**98.8%**	**73.6**	**1.0**	94.7%	**159.6**	**1.0**	**95.0%**	**129.2**	**9.5**
Signhunter [11]	**100.0%**	60.2	23.0	**100.0%**	69.8	34.0	**100.0%**	116.0	38.5	99.4%	178.3	49.0
MCG +Signhunter	**100.0%**	**29.6**	**1.0**	**100.0%**	**19.0**	**1.0**	**100.0%**	**62.8**	**1.0**	**100.0%**	**91.8**	**8.0**
Square [17]	**100.0%**	72.8	26.0	**100.0%**	82.6	28.0	**100.0%**	136.1	68.0	99.4%	210.8	64.0
MCG +Square	**100.0%**	**31.7**	**1.0**	**100.0%**	**24.8**	**1.0**	**100.0%**	**59.9**	**1.0**	**100.0%**	**123.8**	**24.0**
MetaAttack [20]	92.2%	3302.6	2641.0	95.4%	3075.2	2213.0	87.4%	3824.2	3309.0	94.7%	2634.8	1772.0
MCG +MetaAttack	**94.1%**	**1692.3**	**1.0**	**96.8%**	**1025.9**	**1.0**	**92.5%**	**2076.8**	**1.0**	**95.3%**	**2107.1**	**1324.0**
有目标攻击												
NES [12]	67.9%	5734.8	5860.0	79.4%	4944.1	4621.0	33.0%	6137.9	6259.0	63.3%	4921.1	4726.0
MCG +NES	**75.3%**	**5499.6**	**5294.0**	**81.0%**	**4720.9**	**4559.0**	**37.5%**	**5839.1**	**5882.0**	**66.2%**	**4687.7**	**4370.0**
CG-Attack [53]	96.9%	2553.3	1801.0	**92.9%**	2447.7	1731.0	77.8%	2976.5	2191.0	91.0%	**2260.3**	1481.0
MCG +CG-Attack	**98.0%**	**2501.8**	**1781.0**	91.9%	**2374.4**	**1721.0**	**79.8%**	**2960.2**	**2101.0**	**94.9%**	2272.1	**1441.0**
SimBA-DCT [10]	56.0%	6927.2	7196.0	71.7%	6569.9	6507.0	41.0%	6795.4	7028.0	56.8%	6094.9	6107.0
MCG +SimBA-DCT	**66.8%**	**6460.8**	**6607.0**	**79.3%**	**6023.3**	**6038.0**	**60.6%**	**5744.8**	**5626.0**	**72.8%**	**5576.9**	**5564.0**
Signhunter [11]	**100.0%**	1332.7	922.0	**100.0%**	1115.8	734.0	99.4%	1786.8	1064.0	**100.0%**	1836.2	1063.0
MCG +Signhunter	**100.0%**	**1043.8**	**596.5**	**100.0%**	**788.8**	**446.0**	**99.7%**	**1428.4**	**782.0**	99.7%	**1486.7**	**865.0**
Square [17]	**100.0%**	1097.7	819.0	**100.0%**	1112.7	819.0	99.7%	1678.7	1242.0	99.0%	1789.0	1120.0
MCG +Square	**100.0%**	**797.6**	**518.5**	**100.0%**	**784.4**	**529.0**	**100.0%**	**1148.9**	**661.0**	**99.7%**	**1455.2**	**868.0**
MetaAttack [20]	33.8%	8202.6	8041.5	17.5%	8341.4	8806.0	10.3%	8408.4	8585.0	**19.5%**	7176.8	8141.5
MCG +MetaAttack	**45.6%**	**7315.1**	**8033.0**	**38.1%**	**6425.6**	**7721.0**	**20.6%**	**7029.9**	**7596.5**	**19.5%**	**6946.5**	**7706.0**

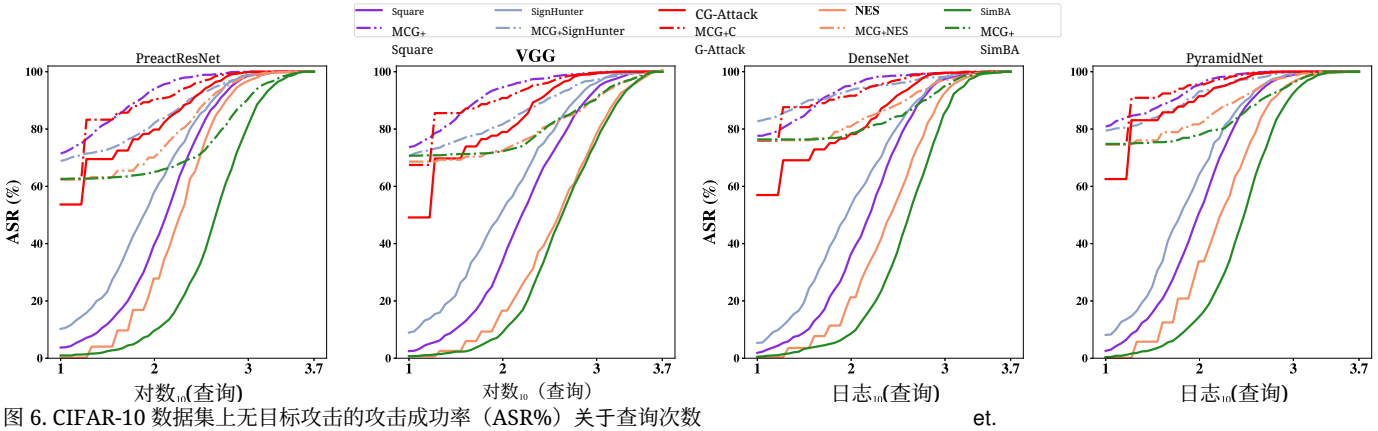


图 6. CIFAR-10 数据集上无目标攻击的攻击成功率（ASR%）关于查询次数 et.

表3 ImageNet 数据集上针对对抗防御模型的无目标攻击。

目标模型 → / 攻击方法 ↓	JPEG-Compress-WRN-50			SND-WRN-50			FastAdv-ResNet-50			FreeAdv-ResNet-50		
	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数
NES [12]	13.2%	5682.1	3243.0	76.6%	3879.8	3308.5	**23.5%**	7501.5	7986.5	15.7%	7188.5	6322.0
MCG +NES	**98.9%**	**1412.1**	**1.0**	**81.7%**	**1795.7**	**1.0**	**23.5%**	**2415.4**	**715.0**	**22.4%**	**2104.1**	**685.0**
CG-Attack [53]	39.0%	2227.5	1161.0	**79.8%**	540.5	81.0	58.5%	1789.7	871.0	61.3%	2374.0	1121.0
MCG +CG-Attack	**91.7%**	**180.9**	**1.0**	**79.8%**	**331.5**	**1.0**	**64.3%**	**1634.4**	**801.0**	**64.7%**	**2305.1**	**1081.0**
SimBA-DCT [10]	14.3%	4688.3	4545.0	10.5%	608.3	22.0	**45.5%**	702.5	72.5	**52.7%**	**792.5**	245.0
MCG +SimBA-DCT	**67.1%**	**543.9**	**1.0**	**60.4%**	**498.3**	**1.0**	**45.5%**	**634.4**	**34.0**	**52.7%**	821.9	**77.0**
Signhunter [11]	**100.0%**	120.7	39.0	**89.4%**	133.4	31.0	**89.4%**	1101.5	37.5	**88.2%**	**931.3**	42.0
MCG +Signhunter	**100.0%**	**69.8**	**1.0**	89.1%	**65.6**	**1.0**	**89.4%**	**1083.5**	**36.5**	86.5%	1058.3	**40.0**
Square [17]	**100.0%**	140.9	71.0	88.9%	1487.8	154.0	**92.4%**	940.5	182.0	**91.4%**	871.3	170.0
MCG +Square	**100.0%**	**57.5**	**1.0**	**90.7%**	**307.4**	**1.0**	**92.4%**	**864.9**	**137.0**	**91.4%**	**724.9**	**123.5**
MetaAttack [20]	8.0%	3515.9	1557.0	11.7%	2449.1	1777.0	53.8%	4832.2	5063.5	55.3%	4614.9	4631.0
MCG +MetaAttack	**50.9%**	**117.2**	**1.0**	**53.1%**	**535.7**	**1.0**	**55.7%**	**4119.9**	**3969.5**	**56.3%**	**3580.2**	**3094.0**

表 4 基于组合的方法在 ImageNet 数据集上的比较

目标模型 → / 攻击方法 ↓	ResNet-18			VGG-16			WRN-50			Inception-V3		
	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数
无目标攻击												
TREMBA [23]	**100.0%**	664.5	169.0	99.1%	160.2	**1.0**	99.0%	697.6	148.0	96.9%	1232.7	211.0
AdvFlow [22]	**100.0%**	578.8	400.0	**100.0%**	693.3	420.0	**100.0%**	937.9	400.0	**97.2%**	716.3	200.0
MCG +CG-Attack	**100.0%**	**95.4**	**21.0**	99.7%	**69.3**	**1.0**	99.2%	**247.3**	**21.0**	97.0%	**271.1**	**21.0**
有目标攻击												
TREMBA [23]	81.4%	3197.2	1997.5	91.1%	2493.5	1759.0	60.2%	5140.5	3529.0	13.3%	9462.1	9433.0
AdvFlow [22]	83.8%	4650.0	4400.0	81.6%	4407.8	4200.0	61.4%	5210.6	4980.0	81.8%	3734.1	3200.0
MCG +CG-Attack	**98.0%**	**2501.8**	**1781.0**	**91.9%**	**2374.4**	**1721.0**	**79.8%**	**2960.2**	**2101.0**	**94.9%**	**2272.1**	**1441.0**

比无目标攻击更难实现，但我们的 MCG 在攻击全部四个目标模型时仍取得了令人满意的攻击成功率。MCG+Square 达到了最佳攻击性能。与原始 Square 攻击相比，MCG+Square 以显著更少的查询次数实现了 100% 的攻击成功率。例如，MCG 的平均查询次数和中位查询次数分别比原始 Square 攻击少 4.6 倍以上和 14.3 倍以上。

ImageNet 上的性能。ImageNet 是一个比 CIFAR-10 大得多的数据集。表2报告了无目标攻击和有目标攻击在 ImageNet 上的性能。

我们观察到类似的现象：将所提出的 MCG 与不同攻击方法结合时，均能取得稳定的提升。在无目标攻击设置下，有些情况下攻击成功率的提升十分明显。例如，使用 CG-Attack 攻击 WRN-50 和 VGG-16 时，攻击成功率分别为 89.6% 和 96.5%。我们的 MCG 进一步提升了 CG-Attack 攻击 WRN-50 时约提升 10% ，攻击 VGG-16 时约提升3% 。此外，效率方面的提升也十分显著，尤其是对于 NES 和 SimBA。在有目标攻击设置下，MCG 几乎在所有情况下都带来了不同程度的提升。相比无目标攻击设置，攻击成功率的提升更为明显。尽管将 MCG 与 SignHunter 结合以攻击 InceptionV3 时，攻击成功率略有下降（低了0.3% ），但攻击成本减少了近20%。

所有这些结果都表明，我们的 MCG 能够为各种现成的黑盒攻击方法提供有效的初始扰动或扰动分布，从而提升其性能。请注意，元生成器的元训练过程不涉及任何目标模型或攻击方法。因此，元生成器只需训练一次，便可与不同的黑盒攻击方法结合，对不同的黑盒模型发起攻击，而无需重新训练，这证实了

表5 与基于迁移的方法比较：ImageNet 数据集上的结果。

目标模型 → 攻击方法 ↓	ResNet-18 ASR	VGG-16 ASR	WRN-50 ASR	Inception-V3 ASR
PGD [66]	35.5%	29.6%	36.6%	15.7%
PGD +不使用代理模型的MCG	56.8%	70.0%	49.5%	**26.4%**
PGD +使用固定代理模型的MCG	**57.9%**	**70.4%**	**50.1%**	**26.4%**
MI [34]	56.1%	62.0%	66.8%	26.4%
MI + 不使用代理模型的MCG	62.6%	71.6%	67.6%	45.1%
MI + 使用固定代理模型的MCG	**63.2%**	**73.6%**	**69.3%**	**45.4%**
TIMI [67]	56.7%	**63.4%**	**62.3%**	42.5%
TIMI + 不使用代理模型的MCG	66.4%	62.1%	58.3%	42.7%
TIMI + 使用固定代理模型的MCG	**66.7%**	**63.4%**	59.4%	**45.4%**
DI [68]	44.2%	42.0%	43.3%	20.1%
DI + 不使用代理模型的MCG	68.5%	**80.0%**	67.1%	49.6%
DI + 使用固定代理模型的MCG	**69.2%**	**80.0%**	**68.7%**	**50.7%**

所提出框架的灵活性和泛化能力。

攻击防御模型。 为验证所提出方法针对对抗防御模型的有效性，我们对采用不同防御策略训练的多种防御模型进行了攻击实验，包括 JPEG压缩 [62],随机噪声扰动 [63],和对抗训练 [64]。

JPEG压缩防御（$i.e.$，JPEG-Compress-WRN-50）试图通过压缩过程消除对抗样本的影响。小噪声防御（$i.e.$，SND-WRN- 50）专门针对基于查询的攻击，通过引入额外的随机噪声来阻碍攻击者正确估计梯度。如表3所示，在攻击 JPEG-Compress- WRN-50 和 SND-WRN-50 时，NES、CG-Attack、SimBA-DCT 和 MetaAttack 的性能均大幅下降。相比之下，与我们的框架结合后，这些方法在三个指标上的性能都得到显著提升。例如，在使用 CG-Attack 攻击 JPEG-Compress- WRN-50 时，攻击成功率仅为 39% ，中位查询次数为 1161，在大多数攻击中均无法成功。我们的框架将其攻击成功率提升至91.7% ，并将其中位查询次数降至1。此外，我们的方法还降低了 Square 和 Signhunter 的均值与中位查询次数。

对抗训练扩大了鲁棒模型与普通模型之间分类边界的差异，并极大地限制了模型级对抗可迁移性。如表3所示，在攻击经过对抗训练的模型时，与攻击普通模型相比，所有方法的攻击性能都大幅下降。尽管提升幅度不如攻击普通模型时显著，但我们的方法在大多数情况下仍能提高攻击性能。

与基于迁移的方法比较。 为验证元学习增强样本级对抗迁移性的有效性，我们在 transferattack 设置下进行了无目标攻击实验，将我们的元生成器与几种基于迁移的攻击方法进行比较； $i.e.$，微调时未使用来自黑盒目标模型的任何信息。

我们的元生成器包含 c-Glow 的预训练阶段，预训练时使用一种攻击方法生成对抗样本进行训练。对抗样本的质量会极大地影响性能。在其他实验中，攻击方法采用 PGD。由于基于迁移的方法比 PGD 能生成可迁移性更好的对抗样本，为了与基于迁移的方法进行公平比较，我们分别使用这三种基于迁移的方法生成的对抗样本来预训练元生成器。

结果如表5所示。“X +不使用代理模型的MCG”表示我们使用“X”预训练 c-Glow，然后直接使用元生成器生成对抗样本，而不使用替代模型。“X +使用固定代理模型的MCG”表示预训练后，我们先使用替代模型对元生成器进行微调，然后使用元生成器生成对抗样本。如表5所示，在大多数情况下，直接使用我们的元生成器比基于迁移的方法性能更好。使用替代模型微调元生成器还能进一步提升性能。基于迁移的方法可能会对替代模型过拟合，因为对抗样本的生成完全依赖于替代模型。相比之下，我们的元生成器能够捕捉样本级迁移性，从而缓解过拟合问题。生成过程由元生成器学习到的先验知识和替代模型共同决定。

与基于组合的方法比较。由于我们的方法可视为基于迁移的方法与基于查询的方法的组合，因此我们将其与两种基于组合的黑盒攻击方法进行比较： $i.e.$、TREMBA [23]和 AdvFlow [22]。这些方法利用模型级对抗可迁移性来提升性能，并且通常结合进化方法来调整其分布映射。为保证公平，在元测试阶段，我们将分布调整方法 CG-Attack [53]整合到框架中进行评估。

ImageNet 上的结果如表 4 所示。在几乎所有攻击情形下，我们的方法在全部三项指标上均取得最佳性能。尽管其他方法试图利用不同模型之间的可迁移性，但在攻击不同目标模型时，其攻击成功率似乎并不稳定。例如，在有目标攻击中攻击 ResNet-18 和 VGG-16 时，TREMBA 的攻击成功率分别为 81.4% 和 91.1%。但在攻击 WRN-50 和 Inception-V3 时，攻击成功率则降至 60.2% 和 13.3%。这些结果表明，TREMBA 无法很好地泛化到对不同目标模型发起攻击。相比之下，我们的方法在攻击不同模型时具有更强的泛化能力。CIFAR-10 上的实验结果见附录B.1节。

4.3 开放集攻击场景中的实验

在闭集攻击场景中，替代模型和目标模型使用相同的训练数据集，即，目标模型的训练数据集对攻击者可见，而这在真实攻击场景中难以实现。在现实场景中，替代模型与目标模型共享的知识会更少。这种情况大大增加了攻击难度。因此，本节验证我们的方法在开放集攻击场景中的有效性：替代模型和目标模型使用互不相交的训练数据集进行训练，且两个模型的输出类别没有重叠。在这种开放集攻击设置下，攻击者要将有限的先验知识迁移到未知领域，颇具挑战性。具体而言，在实验中，我们使用两个数据集，分别在一个数据集上训练替代模型，在另一个数据集上训练目标模型。元生成器的训练数据与替代模型使用的数据相同。在 CIFAR-10 上训练并在 CIFAR-100 上测试的结果如表 6 所示。请注意，ImageNet 和 OpenImage [69] 上的其他实验结果见附录 B.2节。

由于替代模型和目标模型是在类别不同的训练集上训练的，模型层面的对抗迁移性效果受到限制，因此与表 1 中的结果相比，性能有所下降。然而，

表 6 在开放集场景下，于 CIFAR-10 数据集上训练并在 CIFAR-100 数据集上测试的无目标攻击。

目标模型 →	ResNet-18			VGG-16			WRN-50			Inception-V3		
攻击方法 ↓	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数
NES [12]	96.1%	2054.0	1555.0	84.3%	1691.0	904.0	89.5%	2229.6	1534.0	94.6%	2044.7	1471.0
MCG +NES	**97.0%**	**773.1**	**1.0**	**88.5%**	**910.5**	**1.0**	**94.0%**	**833.7**	**1.0**	**95.7%**	**1025.2**	**1.0**
CG-Attack [53]	99.4%	226.2	**1.0**	**98.1%**	294.7	**1.0**	98.4%	285.2	**1.0**	99.1%	330.8	**1.0**
MCG +CG-Attack	**99.5%**	**135.1**	**1.0**	97.8%	**244.8**	**1.0**	**98.7%**	**163.5**	**1.0**	**99.6%**	**196.5**	**1.0**
SimBA-DCT [10]	**99.2%**	222.8	84.0	96.8%	321.4	66.0	97.4%	268.1	86.5	98.5%	231.6	76.0
MCG +SimBA-DCT	**99.2%**	**119.5**	**1.0**	**97.0%**	**218.3**	**1.0**	**98.0%**	**155.2**	**1.0**	**98.9%**	**184.1**	**1.0**
Signhunter [11]	**100.0%**	88.2	28.0	**99.6%**	136.7	22.0	**99.8%**	134.5	28.0	**100.0%**	117.8	32.0
MCG +SignHunter	**100.0%**	**57.8**	**1.0**	**99.6%**	**110.2**	**1.0**	**99.8%**	**94.1**	**1.0**	**100.0%**	**92.0**	**1.0**
Square [17]	**99.9%**	103.8	17.0	**99.5%**	213.9	24.0	**99.8%**	178.1	16.0	99.8%	121.2	15.0
MCG +Square	**99.9%**	**47.9**	**1.0**	**99.5%**	**92.7**	**1.0**	**99.8%**	**83.1**	**1.0**	**99.9%**	**75.6**	**1.0**
MetaAttack [20]	**100.0%**	518.9	443.0	99.7%	591.4	443.0	99.9%	610.1	447.0	**100.0%**	523.6	445.0
MCG +MetaAttack	99.9%	**227.4**	**1.0**	**99.9%**	**396.4**	**1.0**	**100.0%**	**303.6**	**1.0**	**100.0%**	**291.4**	**1.0**

表7 针对*Imagga* 标签 API 的无目标攻击。

	基线			结合 MCG		
方法	ASR	Mean	中位数	ASR	Mean	中位数
NES [12]	44.0%	35.3	21.0	**67.0%**	**8.8**	**1.0**
CG-Attack [53]	74.0%	33.5	21.0	**81.0%**	**21.4**	**1.0**
SimBA-DCT [10]	45.0%	93.8	56.0	**57.0%**	**45.5**	**1.0**
SignHunter [11]	51.0%	45.3	20.5	**82.0%**	**19.5**	**1.0**
Square [17]	49.0%	50.8	15.5	**69.0%**	**13.7**	**1.0**
MetaAttack [20]	16.0%	230.1	95.0	**61.0%**	**101.4**	**1.0**

我们的框架在开放集设置下仍能提升现有的黑盒攻击方法。MCG 可以降低攻击的查询成本，并在大多数情况下提高攻击成功率，这表明元生成器能够快速适应，在不同数据集上攻击不同的目标模型。

4.4 攻击现实世界 API 的实验

为验证我们的框架在现实场景中的有效性，我们对 I magga 图像标注 API[1]进行了攻击实验。Imagga 的模型是在一个包含3,000 多种日常物体的未知数据集上训练的。给定一张查询图像，该 API 会返回一个可能的标签列表以及相应的置信度分数。我们从 ImageNet 验证集中随机选取100 张图像，并将查询上限设为500。我们将无目标攻击的目标定义为移除良性图像排名前三的标签。由于这些图像来自 ImageNet，因此可以使用预训练的替代模型对元生成器进行微调。$\mathcal{L}_{adv}$ 设为排名前三的标签中的最高分数。随后使用适配生成器为现有黑盒攻击方法生成初始扰动，以攻击该 API。如表7所示，与我们的框架结合后，所有方法的性能都显著提升。具体而言，所有方法的中位查询次数均降至1，平均查询次数也大幅改善。这些结果表明，我们的框架适用于现实场景。

4.5 消融研究

为验证元训练以及元测试中的微调阶段是否有效，我们在 ImageNet 上进行了无目标

1. https://imagga.com/solutions/auto-tagging

表 8 ImageNet 数据集上的消融研究。表中的所有方法均与 Square 攻击结合，用于无目标攻击。

目标模型 →	ResNet-18			VGG-16		
攻击方法 ↓	FASR	Mean	中位数	FASR	Mean	中位数
Flow	35.2%	48.6	13.0	46.1%	39.1	4.0
使用固定代理模型的MCG	57.9%	35.3	**1.0**	70.4%	30.0	**1.0**
使用微调替代模型的MCG	**60.1%**	**31.7**	**1.0**	**71.3%**	**24.8**	**1.0**

攻击消融实验。结果如表 8 所示。‘Flow’是一种直接利用扰动学习条件 Glow（c-Glow）模型的方法，而不使用元学习中的对抗损失或参数更新策略。扰动通过对替代模型应用投影梯度下降（PGD） [66] 攻击生成。测试期间不进行微调。‘使用固定代理模型的MCG’表示测试期间使用固定替代模型微调元生成器。‘使用微调后的替代模型的MCG’表示测试期间先利用目标模型的查询反馈更新替代模型，再使用更新后的替代模型微调元生成器。所有这些方法都结合 Square 攻击来查询目标模型。为评估初始扰动的有效性，我们使用首次攻击成功率（FASR）而非攻击成功率作为指标。FASR 指的是直接使用生成器生成的扰动攻击目标模型时的成功率。

如表 8 所示，与 Flow 相比，使用固定替代模型的 MCG 在三项指标上均取得了显著更好的性能。FASR 和中位查询次数均有明显改善。结果证明了元训练形式的有效性：它通过捕获更有效的通用先验知识，了解如何攻击不同样本，从而提升了样本级迁移性。其提供的初始扰动优于 Flow 生成的扰动。此外，与使用固定替代模型的 MCG 相比，使用微调替代模型的 MCG 在 FASR 和攻击效率方面进一步提升，这证实了历史攻击信息的有效性，即，通过将目标模型的信息迁移至替代模型，我们可以获得适配性更强的生成器。

5 结论

我们提出了一种新颖的黑盒攻击框架，将其建模为元学习问题，以提升样本级

对抗迁移性和攻击效率。由于黑盒目标模型的架构和参数未知，我们利用模型级对抗可迁移性，提出使用替代模型进行元训练。由于标准的元测试流程无法应用于黑盒攻击，我们提出了一种三阶段攻击流程来微调元模型，包括利用目标模型的历史攻击信息微调替代模型、使用更新后的替代模型为良性图像微调元生成器，以及将其用作初始化，以加速现成的黑盒攻击方法。涵盖闭集和开集场景以及攻击在线 API 的全面实验，证明了所提模型的有效性。

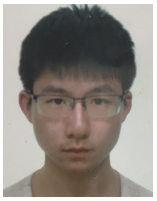
参考文献

[1] I. J. Goodfellow、J. Shlens 和 C. Szegedy，《解释并利用对抗样本》，载于国际学习表征会议论文集，2015年。[2] S.-M. Moosavi-Dezfooli、A. Fawzi、O. Fawzi 和 P. Frossard，《通用对抗扰动》，载于*IEEE*计算机视觉与模式识别会议论文集，2017年，第1765–1773页。[3] A. Kurakin、I. J. Goodfellow 和 S. Bengio，《物理世界中的对抗样本》，载于人工智能安全与防护。Chapman and Hall/CRC，2018年，第99–112页。[4] N. Carlini 和 D. Wagner，《走向神经网络鲁棒性的评估》，载于*2017*年 *IEEE*安全与隐私研讨会（*SP*）。IEEE，2017年，第39–57页。[5] C. Laidlaw 和 S. Feizi，《函数型对抗攻击》，载于神经信息处理系统进展会议论文集，第32卷，2019年。[6] R. Duan、X. Ma、Y. Wang、J. Bailey、A. K. Qin 和 Y. Yang，《对抗伪装：利用自然风格隐藏物理世界攻击》，载于*IEEE*计算机视觉与模式识别会议论文集，2020年，第1000–1008页。[7] W. Feng、B. Wu、T. Zhang、Y. Zhang 和 Y. Zhang，《Meta-attack：类别无关且模型无关的物理对抗攻击》，载于*IEEE*计算机视觉与模式识别会议论文集，2021年，第7787–7796页。[8] A. Athalye、L. Engstrom、A. Ilyas 和 K. Kwok，《合成鲁棒对抗样本》，载于国际机器学习会议论文集，2018年，第284–293页。[9] S. T. Jan、J. Messou、Y.-C. Lin、J.-B. Huang 和 G. Wang，《连接数字世界与物理世界：提升对抗攻击的鲁棒性》，载于*AAAI*人工智能会议论文集，第33卷，第01期，2019年，第962–969页。[10] C. Guo、J. R. Gardner、Y. You、A. G. Wilson 和 K. Q. Weinberger，《简单的黑盒对抗攻击》，载于国际机器学习会议论文集，2019年，第2484–2493页。[11] A. Al-Dujaili 和 U. O'Reilly，《黑盒攻击只需符号位》，载于国际学习表征会议论文集，2020年。[12] A. Ilyas、L. Engstrom、A. Athalye 和 J. Lin，《在查询和信息受限条件下的黑盒对抗攻击》，载于国际机器学习会议论文集，2018年，第2137–2146页。[13] A. Ilyas、L. Engstrom 和 A. Madry，《先验信念：结合多臂老虎机与先验的黑盒对抗攻击》，载于国际学习表征会议论文集，2019年。[14] P.-Y. Chen、H. Zhang、Y. Sharma、J. Yi 和 C.-J. Hsieh，《ZOO：无需训练替代模型、基于零阶优化的深度神经网络黑盒攻击》，载于第十届 *ACM*人工智能与安全研讨会论文集，2017年，第15–26页。[15] C.-C. Tu、P. Ting、P.-Y. Chen、S. Liu、H. Zhang、J. Yi、C.-J. Hsieh 和 S.-M. Cheng，《Autozoom：基于自编码器、用于攻击黑盒神经网络的零阶优化方法》，载于*AAAI*人工智能会议论文集，第33卷，第01期，2019年，第742–749页。[16] S. Liu、P.-Y. Chen、X. Chen 和 M. Hong，《通过零阶预言机实现 signSGD》，载于国际学习表征会议论文集，2018年。[17] M. Andriushchenko、F. Croce、N. Flammarion 和 M. Hein，《Square 攻击：通过随机搜索实现查询高效的黑盒对抗攻击》，载于欧洲计算机视觉会议论文集，2020年，第484–501页。[18] S. Cheng、Y. Dong、T. Pang、H. Su 和 J. Zhu，《利用基于迁移的先验改进黑盒对抗攻击》，载于神经信息处理系统进展会议论文集，第32卷，2019年。[19] Y. Guo、Z. Yan 和 C. Zhang，《子空间攻击：利用有潜力的子空间实现查询高效的黑盒攻击》，载于神经信息处理系统进展会议论文集，2019年，第3820–3829页。

[20] J. Du, H. Zhang, J. T. Zhou, Y. Yang, 和 J. Feng，《面向深度神经网络的查询高效元攻击》，载于《国际学习表征会议论文集》，2020年。[21] Y. Li, L. Li, L. Wang, T. Zhang, 和 B. Gong，《NATTACK：学习对抗样本的分布，以改进针对深度神经网络的黑盒攻击》，载于《国际机器学习会议论文集》，2019年，第3866–3876页。[22] H. Mohaghegh Dolatabadi, S. Erfani, 和 C. Leckie，《Advflow：使用归一化流实施隐蔽的黑盒对抗攻击》，载于《神经信息处理系统进展会议论文集》，2020年，第15 871–15 884页。[23] Z. Huang 和 T. Zhang，《利用可迁移的基于模型的嵌入进行黑盒对抗攻击》，载于《国际学习表征会议论文集》，2020年。[24] W. Brendel, J. Rauber, 和 M. Bethge，《基于决策的对抗攻击：针对黑盒机器学习模型的可靠攻击》，载于《国际学习表征会议论文集》，2018年。[25] Y. Dong, H. Su, B. Wu, Z. Li, W. Liu, T. Zhang, 和 J. Zhu，《针对人脸识别的高效基于决策的黑盒对抗攻击》，载于《IEEE计算机视觉与模式识别会议论文集》，2019年，第7714–7722页。[26] A. Rahmati, S.-M. Moosavi-Dezfooli, P. Frossard, 和 H. Dai，《GeoDA：黑盒对抗攻击的几何框架》，载于《IEEE计算机视觉与模式识别会议论文集》，2020年，第8446–8455页。[27] W. Chen, Z. Zhang, X. Hu, 和 B. Wu，《利用随机符号翻转增强基于决策的黑盒对抗攻击》，载于《欧洲计算机视觉会议论文集》，2020年，第276–293页。[28] J. Chen 和 Q. Gu，《Rays：一种用于硬标签对抗攻击的射线搜索方法》，载于《知识发现与数据挖掘会议论文集》，2020年，第1739–1747页。[29] Y. Liu, S. Moosavi-Dezfooli, 和 P. Frossard，《一种受几何启发的基于决策的攻击》，载于《国际计算机视觉会议论文集》，2019年，第4889–4897页。[30] J. Chen, M. I. Jordan, 和 M. J. Wainwright，《HopSkipJumpAttack：一种查询高效的基于决策的攻击》，载于《IEEE安全与隐私研讨会》。IEEE，2020年，第1277–1294页。[31] H. Li, X. Xu, X. Zhang, S. Yang, 和 B. Li，《QEBA：一种查询高效的基于边界的黑盒攻击》，载于《IEEE计算机视觉与模式识别会议论文集》，2020年，第1218–1227页。[32] M. Cheng, T. Le, P. Chen, H. Zhang, J. Yi, 和 C. Hsieh，《查询高效的硬标签黑盒攻击：一种基于优化的方法》，载于《国际学习表征会议论文集》，2019年。[33] M. Cheng, S. Singh, P. H. Chen, P. Chen, S. Liu, 和 C. Hsieh，《Sign-OPT：一种查询高效的硬标签对抗攻击》，载于《国际学习表征会议论文集》，2020年。[34] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, 和 J. Li，《利用动量增强对抗攻击》，载于《IEEE计算机视觉与模式识别会议论文集》，2018年，第9185–9193页。[35] J. Lin, C. Song, K. He, L. Wang, 和 J. E. Hopcroft，《用于对抗攻击的Nesterov加速梯度与尺度不变性》，载于《国际学习表征会议论文集》，2020年。[36] N. Papernot, P. D. McDaniel, 和 I. J. Goodfellow，《机器学习中的可迁移性：从现象到利用对抗样本实施黑盒攻击》，arXiv预印本 arXiv:1605.07277，2016年。[37] Y. Liu, X. Chen, C. Liu, 和 D. Song，《深入探究可迁移对抗样本与黑盒攻击》，载于《国际学习表征会议论文集》，2017年。[38] N. Inkawhich, K. J. Liang, B. Wang, M. Inkawhich, L. Carin, 和 Y. Chen，《跨越特征层级进行扰动，以提升标准和严格黑盒攻击的可迁移性》，载于《神经信息处理系统进展会议论文集》，2020年，第20 791–20 801页。[39] D. Wu, Y. Wang, S. Xia, J. Bailey, 和 X. Ma，《跳跃连接至关重要：使用残差网络生成的对抗样本的可迁移性》，载于《国际学习表征会议论文集》，2020年。[40] F. Tram`er, N. Papernot, I. Goodfellow, D. Boneh, 和 P. McDaniel，《可迁移对抗样本的空间》，arXiv预印本 arXiv:1704.03453，2017年。[41] Z. Yang, L. Li, X. Xu, S. Zuo, Q. Chen, B. Rubinstein, C. Zhang, 和 B. Li，《通过梯度正交性和平滑性刻画对抗迁移性》，载于《国际机器学习会议研讨会论文集》，2020年。[42] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, 和 F. Roli，《对抗攻击为何具有可迁移性？解释规避攻击和投毒攻击的可迁移性》，载于第28届USENIX安全研讨会，2019年，第321–338页。[43] X. Wang, J. Ren, S. Lin, X. Zhu, Y. Wang, 和 Q. Zhang，《解释并增强对抗迁移性的统一方法》，载于《国际学习表征会议论文集》，2021年。[44] X. Wang 和 K. He，《通过方差调节增强对抗攻击的可迁移性》，载于《IEEE计算机视觉与模式识别会议论文集》，2021年，第1924–1933页。

[45] Q. Huang, I. Katsman, H. He, Z. Gu, S. Belongie, 和 S.-N. Lim, 《通过中间层攻击增强对抗样本的可迁移性》, 载于《国际计算机视觉会议论文集》, 2019年, 第4732–4741页。[46] N. Inkawhich, K. Liang, L. Carin, 和 Y. Chen, 《深层特征分布的可迁移扰动》, 载于《国际学习表征会议论文集》, 2019年。[47] Y. Qin, Y. Xiong, J. Yi, 和 C.-J. Hsieh, 《训练用于可迁移对抗攻击的元替代模型》, arXiv预印本 arXiv:2109 01983, 2021年。。[48] Z. Yuan, J. Zhang, Y. Jia, C. Tan, T. Xue, 和 S. Shan, 《元梯度对抗攻击》, 载于《国际计算机视觉会议论文集》, 2021年, 第7728–7737页。[49] Y. Lu 和 B. Huang, 《利用条件生成流进行结构化输出学习》, 载于《AAAI人工智能会议论文集》, 2020年, 第5005–5012页。[50] E. Tabak 和 E. Vanden-Eijnden, 《通过对数似然的对偶上升进行密度估计》, 《Communications in Mathematical Sciences》, 第8卷, 第1期, 第217–233页, 2010年。[51] A. Nichol, J. Achiam, 和 J. Schulman, 《关于一阶元学习算法》, arXiv预印本 arXiv:1803.02999, 2018年。[52] D. P. Kingma 和 J. Ba, 《Adam: 一种随机优化方法》, 载于《国际学习表征会议论文集》, 2015年。[53] Y. Feng, B. Wu, Y. Fan, L. Liu, Z. Li, 和 S. Xia, 《利用部分迁移的条件对抗分布增强黑盒攻击》, 载于《IEEE计算机视觉与模式识别会议论文集》, 2022年, 第15 095–15 104页。[54] A. Krizhevsky, 《从微型图像中学习多层特征》, 2009年。[55] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, 和 F. Li, 《ImageNet大规模视觉识别挑战赛》, 《国际计算机视觉杂志》, 2015年。[56] K. He, X. Zhang, S. Ren, 和 J. Sun, 《深度残差网络中的恒等映射》, 载于《欧洲计算机视觉会议论文集》, 2016年, 第630–645页。[57] G. Huang, Z. Liu, L. van der Maaten, 和 K. Q. Weinberger, 《密集连接卷积网络》, 载于《IEEE计算机视觉与模式识别会议论文集》, 2017年, 第2261–2269页。[58] K. Simonyan 和 A. Zisserman, 《用于大规模图像识别的超深卷积网络》, 载于《国际学习表征会议论文集》, 2015年。[59] D. Han, J. Kim, 和 J. Kim, 《深度金字塔残差网络》, 载于《IEEE计算机视觉与模式识别会议论文集》, 2017年, 第6307–6315页。[60] S. Zagoruyko 和 N. Komodakis, 《宽残差网络》, 载于《英国机器视觉会议》, 2016年。[61] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, 和 Z. Wojna, 《重新思考面向计算机视觉的Inception架构》, 载于《IEEE计算机视觉与模式识别会议论文集》, 2016年, 第2818–2826页。[62] C. Guo, M. Rana, M. Cisse, 和 L. van der Maaten, 《利用输入变换抵御对抗图像》, 载于《国际学习表征会议论文集》, 2018年。[63] J. Byun, H. Go, 和 C. Kim, 《少量输入噪声就足以防御基于查询的黑盒攻击》, 载于《国际学习表征会议论文集》, 2021年。[64] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, 和 T. Goldstein, 《免费进行对抗训练!》, 载于《神经信息处理系统进展会议论文集》, 2019年, 第3353–3364页。[65] E. Wong, L. Rice, 和 J. Z. Kolter, 《快胜于免费: 重新审视对抗训练》, 载于《国际学习表征会议论文集》, 2019年。[66] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, 和 A. Vladu, 《迈向能够抵御对抗攻击的深度学习模型》, 载于《国际学习表征会议论文集》, 2018年。[67] Y. Dong, T. Pang, H. Su, 和 J. Zhu, 《通过平移不变攻击规避针对可迁移对抗样本的防御》, 载于《IEEE计算机视觉与模式识别会议论文集》, 2019年, 第4312–4321页。[68] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, 和 A. L. Yuille, 《利用输入多样性提高对抗样本的可迁移性》, 载于《IEEE计算机视觉与模式识别会议论文集》, 2019年, 第2730–2739页。[69] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov 等, 《Open Images数据集V4》, 《国际计算机视觉杂志》, 第1956–1981页, 2020年。[70] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, 和 D. Song, 《利用对抗网络生成对抗样本》, 载于《国际人工智能联合会议论文集》, 2018年, 第3905–3911页。[71] S. Jandial, P. Mangla, S. Varshney, 和 V. Balasubramanian, 《Advgan++: 利用潜层进行对手生成》, 载于《国际计算机视觉会议研讨会论文集》, 2019年, 第2045–2048页。[72] Y. Xiong 和 C.-J. Hsieh, 《通过学习型优化器改进对抗训练》, 载于《欧洲计算机视觉会议论文集》, 2020年, 第85–100页。

[73] D. Wu、Y. Wang、S.-T. Xia、J. Bailey 和 X. Ma, "跳跃连接很重要: 基于残差网络生成的对抗样本的可迁移性", 发表于国际学习表征会议论文集, 2020年。



FeiYin 目前是清华大学清华大学深圳国际研究生院的硕士生。他目前的研究兴趣包括多媒体和计算机视觉。



Yong Zhang于2018年获得中国科学院自动化研究所模式识别与智能系统专业博士学位。2015年至2017年，他曾在伦斯勒理工学院担任访问学者。目前，他就职于腾讯人工智能实验室。其研究兴趣包括计算机视觉和机器学习。



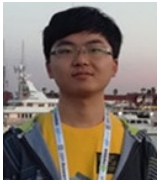
Baoyuan Wu 是香港中文大学（深圳）（香港中文大学（深圳））数据科学学院副教授。他同时担任深圳市大数据研究院（深圳市大数据研究院）大数据安全计算实验室主任。2014年6月，他获得中国科学院自动化研究所模式识别国家重点实验室博士学位。2016年11月至2020年8月，他在腾讯人工智能实验室担任高级研究员及首席研究员。他的研究兴趣包括人工智能安全与隐私、机器学习、计算机视觉和优化。



YanFeng 目前是清华大学深圳国际研究生院的硕士研究生。他目前的研究兴趣包括计算机视觉和人工智能安全。



JingyiZhang 目前是中国电子科技大学计算机科学与工程学院的硕士研究生。他目前的研究兴趣包括多媒体和计算机视觉。



Yanbo Fan 目前是腾讯人工智能实验室的高级研究员。他于2018年在中国北京的中国科学院自动化研究所获得博士学位，并于2013年获得湖南大学计算机科学与技术专业学士学位。他的研究兴趣包括计算机视觉和机器学习。



Yujiu Yang Yujiu Yang（IEEE会员）获得了中国科学院自动化研究所博士学位。他是清华大学深圳国际研究生院副教授。他的研究兴趣包括自然语言处理和计算机视觉。

附录 A 方法分析

A.1 与直接使用替代模型的方法进行比较

我们的框架可以在元测试阶段，将学习到的先验知识与不同类型的现成基于查询的黑盒攻击方法相结合，并显著提升这些方法在攻击效率和攻击成功率方面的性能。

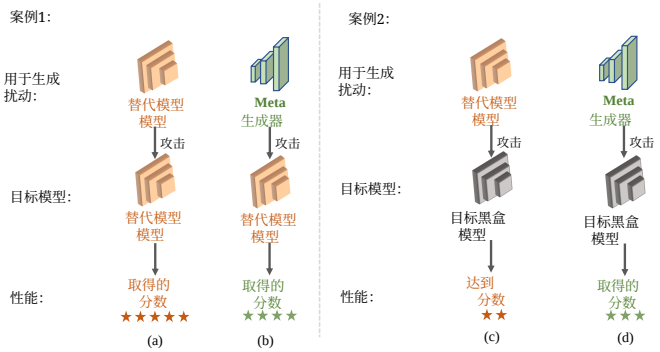


图 7. 迁移攻击的两种案例。

为了与直接使用替代模型的基于迁移的方法进行公平比较，我们仅使用训练好的生成器进行评估。图 7 展示了在迁移攻击场景中，替代模型与我们的元生成器在两种案例下的比较，即不查询未知的目标模型。在案例1中，将替代模型视为目标模型。分别使用替代模型和我们的元生成器生成扰动。在这种情况下，使用替代模型生成扰动的性能远优于我们的元生成器（例如，**攻击成功率：PGD-100 为 100%，而我们的方法为 74%。替代模型：ResNet-50，目标模型：ResNet-50**）。这是因为未知目标模型与替代模型相同。尽管我们的元模型是利用替代模型的梯度训练得到的，但它并不试图精确学习从样本到梯度的映射，而是通过攻击大量样本来学习样本相关的扰动分布，即捕捉样本之间的一些共同属性。给定一个样本，我们的元生成器可以提供一种扰动分布，指示采样扰动有效的概率。它无法像替代模型那样预测精确的扰动。因此，在案例1中，替代模型胜出。

在案例2中，未知的目标模型与替代模型不同。给定一个样本，其生成的扰动完全由该样本和模型本身决定。由于梯度完全来自替代模型，该扰动似乎对替代模型产生了“过拟合”。与此不同，我们的元模型根据样本和学习到的先验知识（即样本之间的共性）生成扰动。因此，生成扰动时考虑的不只是单个样本，而是一组样本。在这种情况下，它的泛化能力始终优于替代模型（例如，**攻击成功率：PGD-100 为 35.5%，而我们的方法为 56.8%。替代模型：ResNet-50；目标模型：ResNet-18**）。

A.2 MCG 的工作原理

我们在图8中展示了本方法的工作示意图，以便更好地理解。图8 (a) 展示了通过**“PGD”**使用替代模型进行攻击的过程。扰动是根据替代模型和干净样本本身生成的。图8 (b) 展示了我们的元生成器如何通过执行涉及多个样本的大量攻击任务，捕获依赖于样本的条件分布。扰动分布以黑色虚线圆圈表示。图8(c) 展示了元生成器如何通过采样一个概率较高的扰动，直接用于攻击未知的目标模型。这是一种迁移攻击。图8(d) 展示了元生成器的元更新。元生成器与基于查询的攻击方法相结合。查询反馈用于更新元生成器。因此，扰动分布从黑色虚线圆圈移至黑色虚线圆圈。后者更优，因为更新利用了有关目标模型的反馈信息。图8(e) 展示了更新后的元生成器的用法。与需要以扰动作为初始化的基于查询的方法结合时，我们可以采样一个概率较高的扰动作为初始化。与需要以分布作为初始化的基于查询的方法结合时，则可以使用生成器表示的分布作为初始化。

如图8所示，我们的方法在元训练阶段学习先验知识，并在元测试阶段更新生成器。它可以与现成的基于查询的方法结合使用。元更新利用查询反馈来改进元生成器。我们的框架能够提高基于查询的方法的攻击效率和攻击成功率。

A.3 使用 c-Glow 作为元生成器的优势

c-Glow 能够对底层分布的精确对数似然进行建模，因此可以直接最小化近似条件对抗分布与真实条件对抗分布（CAD）之间的 KL 散度，而不是像 VAE模型 那样只优化下界，或像学会学习方法那样寻找 CAD 中的近似最大值点。

在学会学习方法中，卷积神经网络和循环神经网络生成器根据分类器的梯度进行优化，这意味着攻击者只能在替代分类器上预训练这些生成器，然后将其参数完全迁移以用于黑盒攻击。正如 [53], 所指出的，这种 完全迁移机制会引入所谓的 替代偏差（这是由于替代模型与目标模型在架构和训练数据集方面存在差异所致），不可避免地损害黑盒攻击的性能。相比之下，由于 c-Glow 的参数分为两部分，$i.e.$、高斯参数和映射参数，我们可以利用部分迁移机制来减轻替代偏差。

A.4 对抗性扰动的可视化

我们在图 9 中展示了五种攻击方法的部分可视化示例。实验设置如下：使用 ResNet-50 作为替代模型、ResNet-18 作为目标模型，执行无目标黑盒攻击。评估了五种攻击方法，包括 MCG（纯迁移）、CG-Attack、MCG +CG-Attack、Square 和 MCG + Square。扰动上限设为$\ell_\infty \leq 0.05$。值得注意的是，我们提出的 MCG 可能会生成近乎对称且呈菱形的图案（见图 1 顶行）。鉴于这些扰动具有极高的可迁移性，它们或许能为分析高可迁移性扰动的特征提供良好的实例。不过，我们也意识到，这些扰动会因干净图像和模型的不同而有所变化。要揭示其某些一般特征，还需要进行更全面的

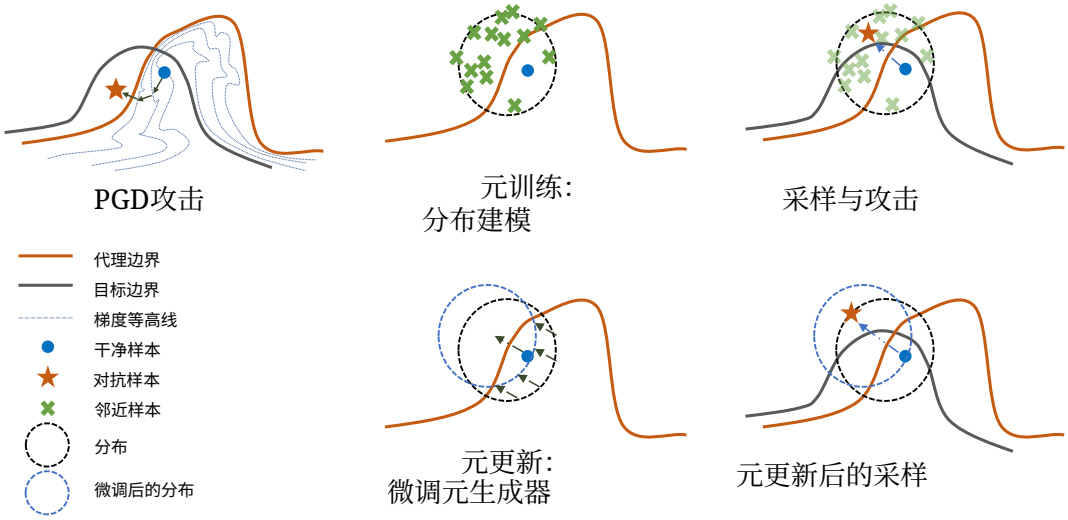


图 8. 所提出方法的工作原理示意图。

评估，并采用巧妙的分析工具/方法。这将作为我们未来工作的研究方向。

附录 B 附加实验结果

B.1 与 CIFAR-10 数据集上的基于组合的方法进行比较

在正文的表 4 中，我们展示了在 ImageNet 数据集上与基于组合的方法进行比较的结果。这里，我们提供在 CIFAR-10 数据集上的比较结果。参与比较的方法包括 TREMBA [23] 和 AdvFlow [22]。结果如表 9 所示。在 CIFAR-10 上，我们的方法在所有攻击场景下的三项指标均取得最佳性能，这与 ImageNet 上的结果一致。

B.2 开集攻击场景下的补充实验

在真实场景中，替代模型与目标模型共享的知识较少，这会大幅增加攻击难度。在正文第 4.3 节中，我们通过在 CIFAR-10 上训练并在 CIFAR-100 上测试，验证了所提方法在不同数据集上的适应性。这里，我们补充了在 ImageNet [55] 和 OpenImage [69]上的实验。

ImageNet 上的实验。为模拟开放集场景，我们首先从 ImageNet 的 1,000 个类别中随机选取 10 个类别，并将这 10 个类别平均分为两组。我们使用其中一组类别的训练集训练替代模型，使用另一组类别的训练集训练目标模型。元生成器的训练数据与替代模型使用的数据相同。测试数据为与目标模型训练类别对应的验证集。ImageNet 上无目标攻击的结果如表 10 所示。与 CIFAR-10 上的开放集攻击相比，ImageNet 上的开放集攻击更具挑战性，因为若干攻击方法的中位查询次数相对较高，例如 NES、SimBA-DCT 和 MetaAttack。在这种设置下，我们的方法仍能提升它们的性能，尤其是 NES 和 SimBA-DCT 的攻击成功率。

在 ImageNet 上训练并在 OpenImage 上测试的实验。为进一步验证在不同数据集之间的适应能力，我们进行了另一项无目标攻击实验：在 ImageNet 数据集上训练元生成器，并在 OpenImage 数据集上测试。目标模型（即 ResNet-18、VGG-16、WRN-50 和 Inception-V3）使用从 OpenImage 中随机选取的 10 个类别的训练数据进行训练。元生成器则在替代模型的指导下，使用 ImageNet 的训练数据进行训练。结果如表 11所示。在大多数情况下，我们的 MCG 都能降低攻击的查询成本并提高攻击成功率，这表明元生成器能够快速适应不同数据集上的不同目标模型。

B.3 与 基于 CNN 和 RNN 的生成器的比较。

基于 CNN 和 RNN 的生成器也可以通过微调来减轻我们框架中的偏差。在我们的框架中，生成器用于捕获对抗样本的先验分布，它是一个可替换组件。其他类型的生成器也可以灵活地集成到框架中，以替换 c-Glow。为比较不同生成器的影响，我们通过将基于 CNN 或 RNN 的生成器集成到框架中来替换 c-Glow，并开展实验。

基于 CNN 的生成器的实现细节。我们遵循[70] 和 [71] 重新实现了基于 CNN 的生成器。骨干网络包括一个特征提取器 f、一个生成器网络G和一个判别器网络 D。我们将图像 x 的特征$f(x)$ 与从可学习均值参数 z中采样的噪声向量拼接起来。然后将拼接结果输入生成器 G。生成器 G 预测与 x对应的对抗扰动x_{adv}。判别器 D 用于区分生成器的输出分布与真实分布。我们还使用了替代模型的对抗损失（文稿中的公式 4）。为限制扰动的幅度，我们最小化L_{inf} 对抗扰动的范数上界。损失函数定义如下：

$$\mathcal{L} = \mathcal{L}_{\text{GAN}} + \alpha\mathcal{L}_{adv} + \beta\mathcal{L}_{inf}, \tag{10}$$

其中

$$\mathcal{L}_{GAN} = \mathbb{E}_x[\log \mathcal{D}(x) + \mathbb{E}_x \log(1 - \mathcal{D}(x + G(z, f(x))))], \tag{11}$$

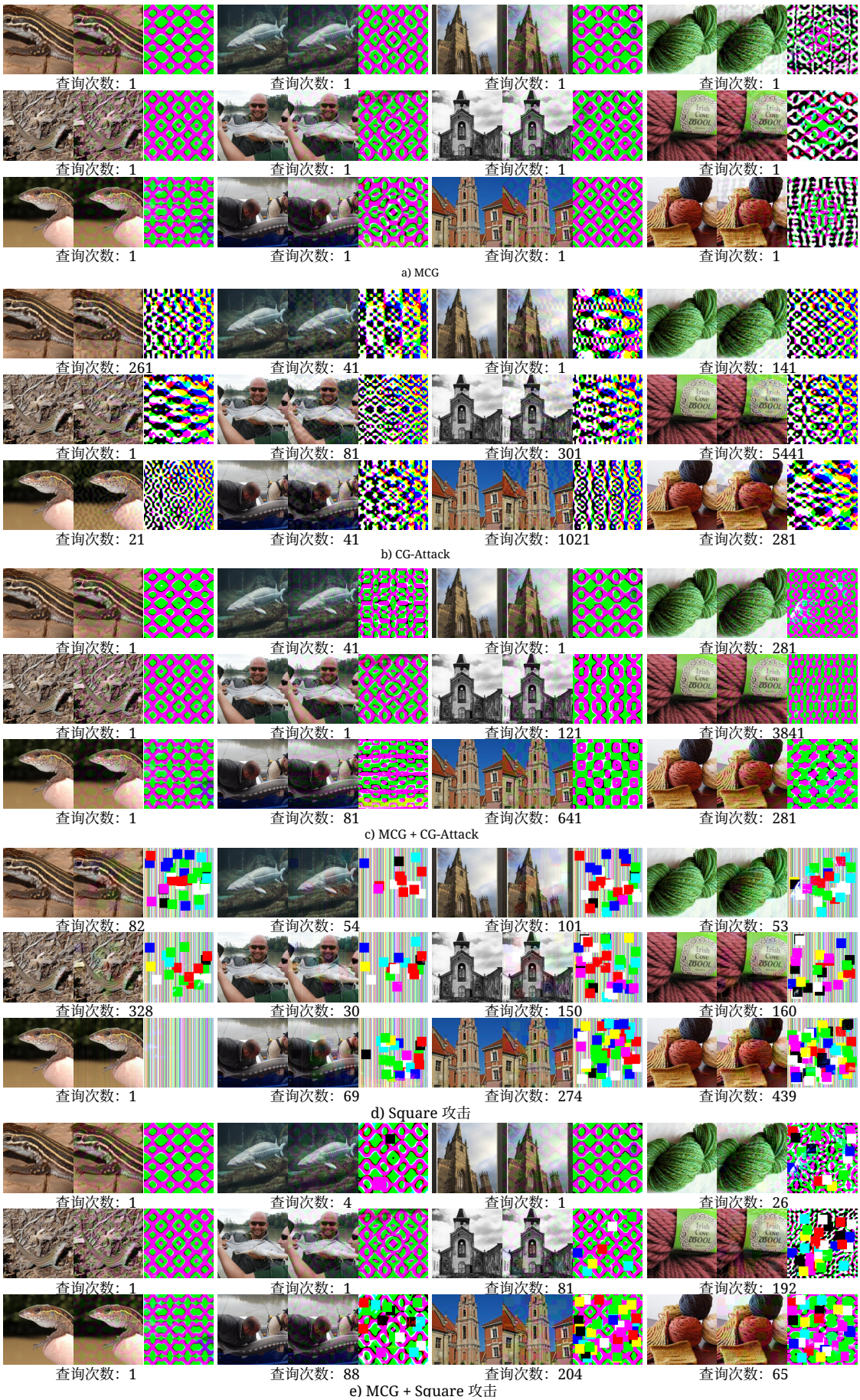


图 9. ImageNet 数据集生成的扰动可视化。在每个三元组块中，各列分别表示良性图像、对抗样本和扰动。扰动的原始范围为 $[-0.05, 0.05]$，为了便于观察，我们将其缩放至范围 $[0, 255]$。

表 9 与基于组合的方法在 CIFAR-10 数据集上的比较

目标模型 →	ResNet-PreAct-110			DenseNet-121			VGG-19			PyramidNet-110		
攻击方法 ↓	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数
无目标攻击												
TREMBA [23]	90.9%	120.7	64.0	97.8%	126.4	66.0	97.7%	125.5	63.0	97.9%	82.3	39.0
AdvFlow [22]	97.2%	841.4	600.0	**100.0%**	1025.3	740.0	98.2%	1079.1	860.0	99.7%	857.5	560.0
MCG +CG-Attack	**100.0%**	**41.8**	**1.0**	**100.0%**	**21.3**	**1.0**	**100.0%**	**38.8**	**1.0**	**100.0%**	**12.8**	**1.0**
有目标攻击												
TREMBA [23]	91.2	1125.3	868.0	92.3	1123.4	879.0	96.5	1331.5	1142.0	98.1	1082.4	759.0
AdvFlow [22]	98.6	911.7	820.0	96.3	1021.5	860.0	**97.4**	1144.1	940.0	**100.0**	908.1	820.0
MCG +CG-Attack	**100.0**	**430.8**	**181.0**	**99.8**	**608.5**	**241.0**	**97.4**	**944.1**	**381.0**	**100.0**	**290.6**	**141.0**

表 10 开放集场景下 ImageNet 数据集上的无目标攻击。

目标模型 →	ResNet-18			VGG-16			WRN-50			Inception-V3		
攻击方法 ↓	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数
NES [12]	66.7%	4684.9	4516.0	68.5%	4903.7	4778.5	74.0%	4199.8	3760.0	94.9%	1989.0	1439.5
MCG +NES	**77.5%**	**4113.1**	**3845.0**	**77.4%**	**4392.6**	**4517.0**	**78.9%**	**3503.1**	**3089.0**	**96.8%**	**1773.5**	**1251.5**
CG-Attack [53]	50.2%	524.8	61.0	36.3%	476.3	**41.0**	55.1%	615.6	141.0	64.4%	**497.3**	**41.0**
MCG +CG-Attack	**57.7%**	**446.2**	**21.0**	**41.2%**	**468.6**	61.0	**55.2%**	**591.3**	**61.0**	**65.0%**	521.4	**41.0**
SimBA-DCT [10]	20.9%	2123.4	2002.0	21.4%	2782.8	3202.0	33.3%	2947.2	3187.0	53.3%	2485.2	2283.0
MCG +SimBA-DCT	**42.9%**	**1182.9**	**829.0**	**35.2%**	**1006.6**	**669.0**	**52.4%**	**1293.4**	**1335.0**	**60.0%**	**1402.7**	**1343.0**
Signhunter [11]	**100.0%**	**142.2**	**60.0**	98.5%	391.6	145.0	**100.0%**	157.7	49.0	**100.0%**	129.5	**45.0**
MCG +Signhunter	**100.0%**	161.0	64.0	**99.2%**	**364.6**	**137.5**	**100.0%**	**153.0**	**44.0**	**100.0%**	**121.9**	46.0
Square [17]	**100.0%**	288.1	196.0	**100.0%**	632.6	336.0	**100.0%**	272.5	156.0	**100.0%**	207.1	126.0
MCG +Square	**100.0%**	**220.1**	**119.0**	**100.0%**	**572.3**	**271.0**	**100.0%**	**220.3**	**112.0**	**100.0%**	**175.8**	**97.0**
MetaAttack [20]	78.8%	5646.7	5951.0	75.7%	5485.2	5293.0	76.2%	5346.7	5505.5	93.6%	4168.6	4198.5
MCG +MetaAttack	**79.2%**	**4360.4**	**4416.0**	**76.6%**	**5078.9**	**5619.0**	**82.4%**	**4555.6**	**4412.0**	**94.5%**	**3896.9**	**3974.0**

表11 在 OpenImage 数据集上的无目标攻击评估。

目标模型 →	ResNet-18			VGG-16			WRN-50			Inception-V3		
攻击方法 ↓	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数
NES [12]	28.3%	4915.9	5041.0	40.4%	3894.7	3718.0	61.1%	4035.6	3823.0	86.9%	2671.2	1954.0
MCG +NES	**52.6%**	**1306.7**	**281.0**	**69.9%**	**764.3**	**1.0**	**74.0%**	**1327.8**	**1.0**	**92.9%**	**1712.8**	**23.0**
CG-Attack [53]	62.3%	2033.8	21.0	66.7%	2044.4	21.0	66.9%	1948.1	**81.0**	61.5%	2273.4	41.0
MCG +CG-Attack	**72.7%**	**1337.4**	**1.0**	**82.5%**	**1419.3**	**1.0**	**78.7%**	**1201.2**	101.0	**75.5%**	**1039.1**	**1.0**
SimBA-DCT [10]	90.4%	374.201	17	**94.9%**	254.7	10.0	**92.5%**	301.4	12.0	83.2%	320.180	11.0
MCG +SimBA-DCT	**90.8%**	**348.5**	**7.0**	94.7%	**226.2**	**1.0**	**92.5%**	**250.7**	**1.0**	**83.9%**	**249.6**	**1.0**
Signhunter [11]	**99.7%**	331.8	11.0	**100.0%**	149.2	7.0	**100.0%**	233.3	13.0	**100.0%**	259.9	8.0
MCG +Signhunter	98.6%	**267.9**	**6.0**	**100.0%**	**100.9**	**1.0**	**100.0%**	**159.9**	**1.0**	**100.0%**	**247.7**	**4.0**
Square [17]	**99.9%**	292.1	69.0	**100.0%**	247.0	68.5	**100.0%**	156.9	61.5	**100.0%**	178.9	56.0
MCG +Square	99.7%	**230.7**	**24.0**	**100.0%**	**180.9**	**1.0**	**100.0%**	**93.9**	**1.0**	**100.0%**	**123.9**	**8.5**
MetaAttack [20]	69.7%	4366.9	3964.0	79.9%	4029.4	3635.5	72.2%	4364.2	3754.0	81.4%	3689.8	3093.0
MCG +MetaAttack	**75.1%**	**3033.8**	**1670.0**	**84.7%**	**1637.9**	**1.0**	**76.9%**	**1722.0**	**1.0**	**86.9%**	**2351.1**	**41.0**

$$\mathcal{L}_{adv} = \mathbb{E}_x[g_w(x + G(z, f(x)), t)], \tag{12}$$

$$\mathcal{L}_{inf} = \mathbb{E}_x\|G(z, f(x))\|_\infty. \tag{13}$$

t 是目标类别， g_w 表示替代分类器。

基于 RNN 的生成器的实现细节。我们遵循[72] 重新实现基于 RNN 的模型。我们将输入的良性图像展平为一维特征向量，并将这些特征直接输入 RNN 网络 G。输入序列的初始隐藏状态 h 从可学习均值参数 z中采样。我们将输出序列从 G 重新变形为相应的空间维度，从而得到对抗扰动。同样，对抗损失 $\mathcal{L}_{adv}$ 和边界损失 $\mathcal{L}_{inf}$ 用于优化生成器。损失函数定义如下：

$$\mathcal{L} = \mathcal{L}_{adv} + \lambda\mathcal{L}_{inf}. \tag{14}$$

其余训练和测试策略的设置与我们的论文中相同。

实验结果。 实验结果如表 12 所示。我们使用 Square [17] 作为基线方法。可以观察到，基于 CNN 和基于 RNN 的生成器都能显著提升基线方法的性能。结果证明了我们框架的可扩展性，即，c-Glow 生成器可以替换为其他类型的生成器。比较三种生成器，基于流的 MCG 的性能优于另外两种。此外，总体而言，

表 12 在 CIFAR-10 数据集上使用卷积神经网络和基于 RNN 的生成器进行无目标攻击的比较。

目标模型 →	ResNet-PreAct-110			DenseNet-121			VGG-19			PyramidNet-110		
攻击方法 ↓	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数
Square	**100.0%**	227.3	144.5	**100.0%**	260.3	159.0	**100.0%**	342.0	175.5	**100.0%**	165.5	100.5
卷积神经网络 MCG + Square	**100.0%**	67.4	**1.0**	**100.0%**	58.7	**1.0**	**100.0%**	120.8	24.0	**100.0%**	39.3	**1.0**
循环神经网络 MCG + Square	**100.0%**	49.9	**1.0**	**100.0%**	69.4	6.0	**100.0%**	97.9	20.0	**100.0%**	33.1	**1.0**
c-Glow MCG + Square	**100.0%**	**39.7**	**1.0**	**100.0%**	**47.6**	**1.0**	**100.0%**	**57.9**	**1.0**	**100.0%**	**29.6**	**1.0**

表 13 基于 MAML 元学习策略在 CIFAR-10 数据集上的无目标攻击比较。

目标模型 →	ResNet-PreAct-110			DenseNet-121			VGG-19			PyramidNet-110		
攻击方法 ↓	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数	ASR	Mean	中位数
Square [17]	**100.0%**	227.3	144.5	**100.0%**	260.3	159.0	**100.0%**	342.0	175.5	**100.0%**	165.5	100.5
MCG-MAML + Square	**100.0%**	42.2	**1.0**	**100.0%**	**45.7**	**1.0**	**100.0%**	107.1	18.0	**100.0%**	**27.2**	**1.0**
MCG-REPTILE + Square	**100.0%**	**39.7**	**1.0**	**100.0%**	47.6	**1.0**	**100.0%**	**57.9**	**1.0**	**100.0%**	29.6	**1.0**

表 14 使用 SGM 对扩展进行验证。在 ImageNet 数据集上的无目标攻击评估。

目标模型 →	ResNet-18			VGG-16			WRN-50			Inception-V3		
攻击方法 ↓	FASR	Mean	中位数	FASR	Mean	中位数	FASR	Mean	中位数	FASR	Mean	中位数
MCG +Square	60.1%	31.7	**1.0**	71.3%	24.8	**1.0**	51.3%	59.9	**1.0**	30.0%	123.8	24.0
SGM + MCG +Square	**67.9%**	**22.9**	**1.0**	**76.2%**	**22.6**	**1.0**	**65.5%**	**41.0**	**1.0**	**43.9%**	**88.2**	**6.0**

基于 RNN 的生成器略优于基于 CNN 的生成器。

B.4 与基于 MAML 的元学习方法比较

MAML 和 REPTILE 都是强大的元学习算法，旨在优化初始表示，使其能够有效地进行微调。MAML 会展开梯度下降算法的计算图并对其求导，而 Reptile 只是在每个任务上执行随机梯度下降，因此计算量和内存占用都比 MAML 少。我们在 CIFAR-10 数据集上开展了无目标攻击实验，对比基于 REPTILE 的 MCG 与基于 MAML 的 MCG。基线方法为 Square [17]。结果如表 13 所示。可以看出，“ MCG-MAML +Square”和“MCG-REPTILE +Square”的性能相近。这一比较也体现了在所提出的框架中使用不同元学习算法的灵活性。

B.5 使用跳跃梯度方法的扩展实验

跳跃梯度方法（SGM） [73] 是一种基于迁移的攻击方法，通过挖掘跳跃连接分支的内部梯度流，生成更具可迁移性的扰动。由于我们利用替代模型的模型级对抗可迁移性，因此可以将 SGM 策略引入我们的框架，以提升攻击性能。与 SGM 类似，我们调整替代模型中跳跃连接分支的反向梯度权重来训练生成器。在攻击过程中，我们对替代模型应用相同的策略，以微调元生成器。我们以 ResNet-50 作为替代模型，在 ImageNet 上开展了无目标攻击实验。结果如表 14 所示。“MCG + Square”是我们原始的方法。“SGM + MCG + Square”表示我们额外引入了 SGM 策略。结果表明，SG M 策略有助于提升模型的 FASR 并减少查询次数。