万能模拟器可行吗？模拟任意未知网络构造高效的黑盒攻击！

马晨
https://github.com/machanic

清華大學
Tsinghua University

1. 对抗攻击和meta-learning简介
2. 基于模拟器的黑盒对抗攻击
3. 我们可以数据不足情况下检测新型对抗样本吗?
4. 未来工作的展望
5. BugTorch开源攻击库的介绍

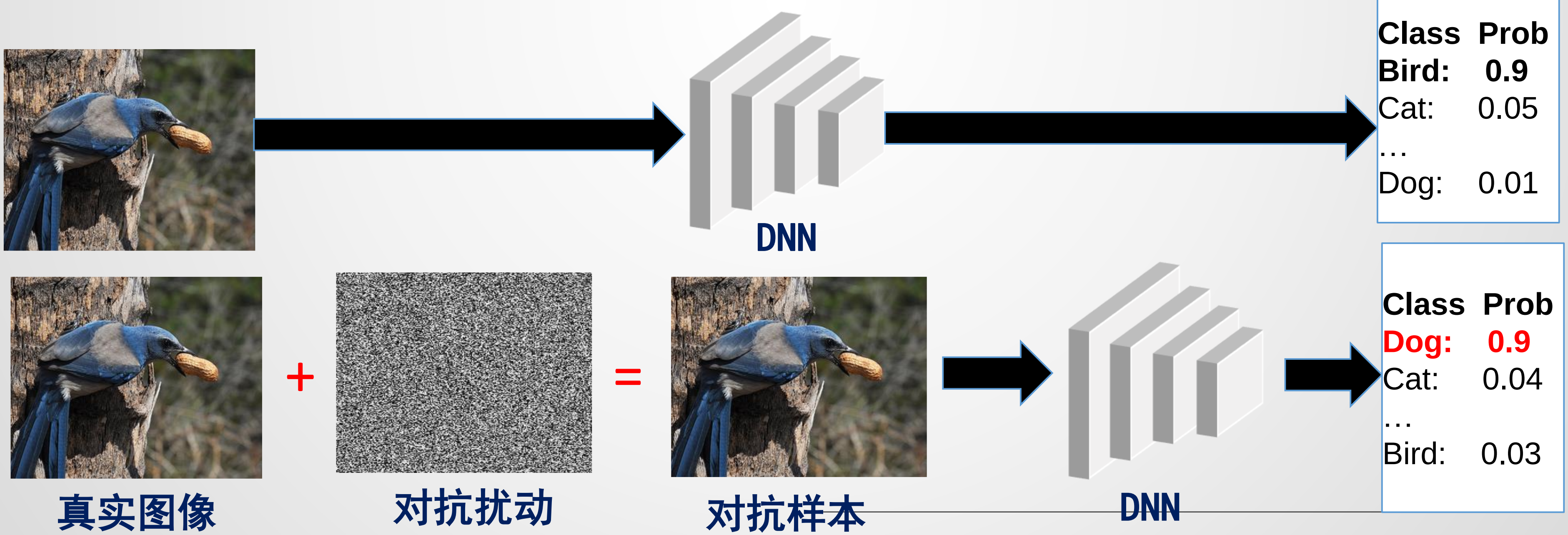
1. 对抗攻击简介

深度神经网络(DNN)的安全性课题

DNN在图像识别领域取得了显著的成绩。

DNN识别对抗样本会分类错误，**表现脆弱**。

对抗攻击是在输入干净图像上添加的微小扰动(perturbation)，肉眼无法识别。



Tsinghua University

1. 对抗攻击简介(黑盒攻击与白盒攻击)

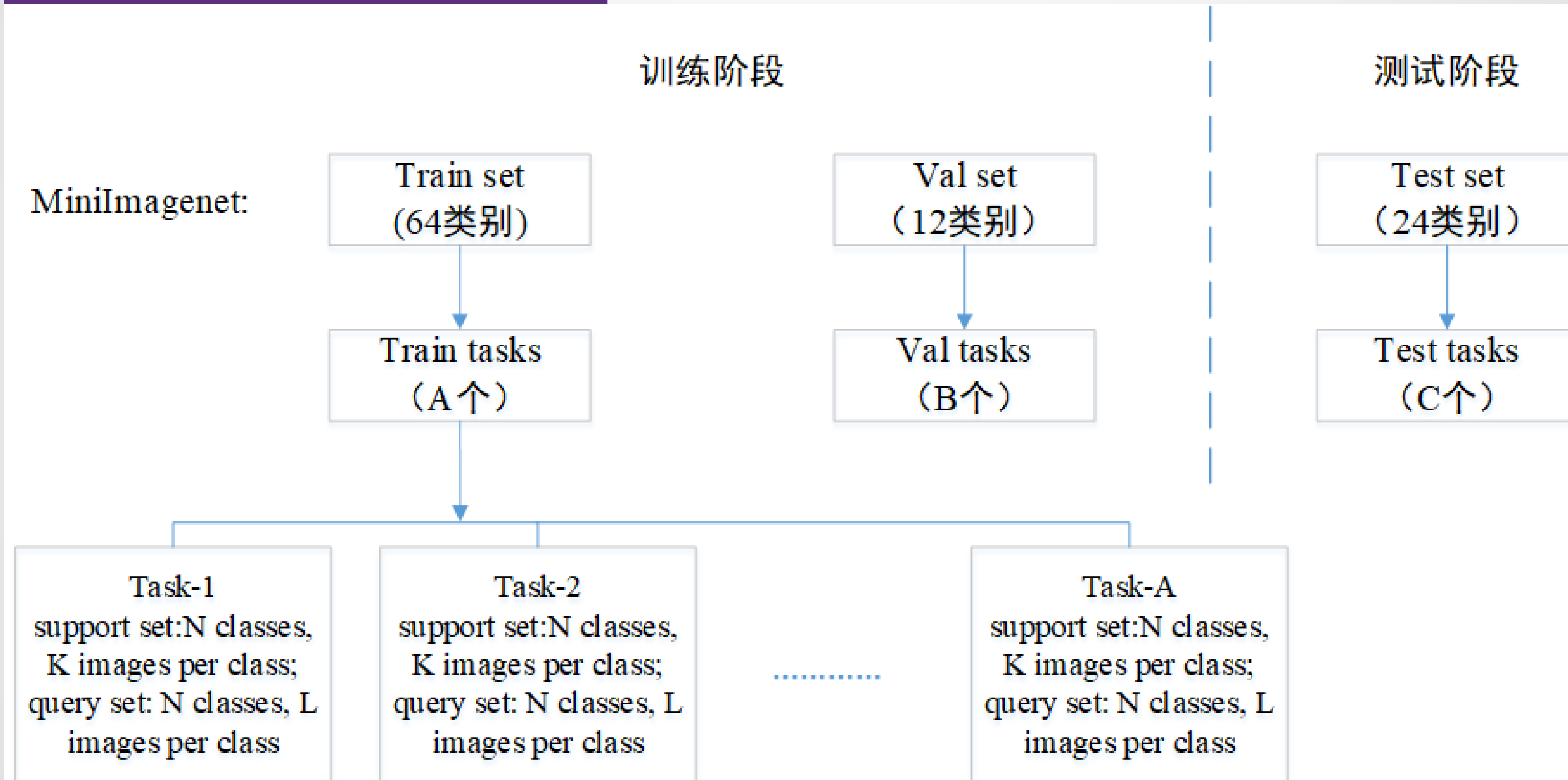
- 黑盒攻击：攻击者拿不到模型参数，模型结构以及梯度。
- 白盒攻击：攻击者可以拿到模型的内部信息：包括模型结构，梯度等。

黑盒攻击
- Query-based attacks
  - Score-based setting: 暴露目标模型的输出概率
  - Decision-based setting:暴露目标模型的输出label
- Transfer-based attacks: 攻击source model产生对抗样本来欺骗target model

Query-based attacks的目标是：如何以最少的query(查询次数)达到最高的攻击成功率。

$$\phi(x_{adv}) = \begin{cases} 1 \quad if\ \hat{y} = y_{adv}\ in\ the\ targeted\ attack \\ \quad or\ \hat{y} \neq y_{adv}\ in\ the\ untargeted\ attack \\ 0\ otherwise(\text{包括查询次数大于}10000) \end{cases}$$

1. 背景知识：meta-learning



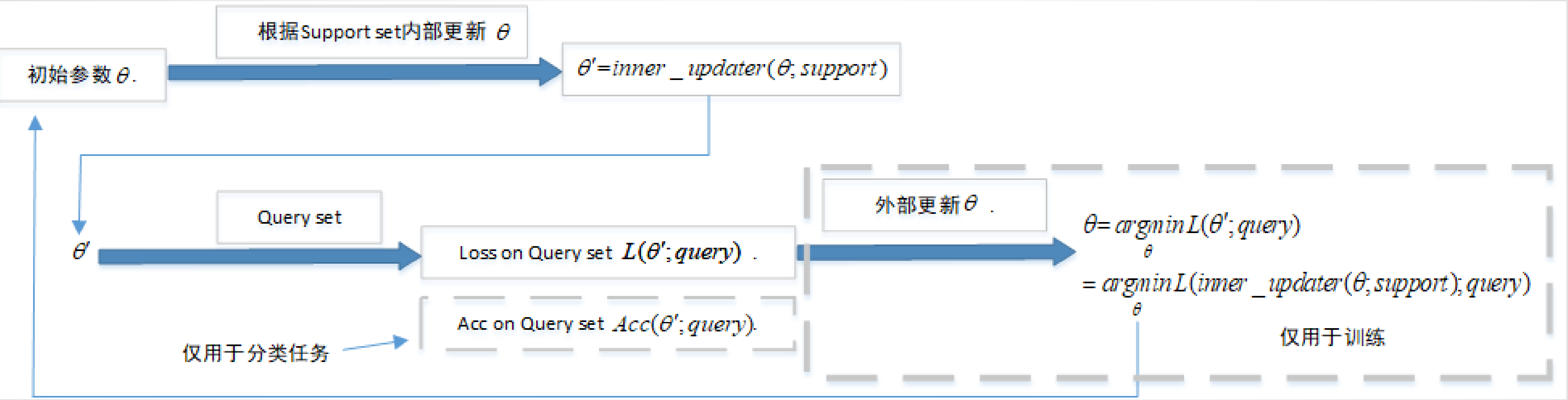
5way 1shot task: K=1, N=5, L是可调超参(L>=1)

以5分类任务为例： Task-1：电视，帽子，狮子，人，猫

Task-2：哈士奇，桌子，飞机，猫，杯子

...

1. 背景知识：meta-learning



2. 模拟器攻击

1. 研究查询高效的黑盒攻击方法

背景：只可以通过查询黑盒模型获得反馈来攻击。
符合现实场景，实用价值高。

(1) 基于查询的黑盒攻击

Andrew Ilyas et al.ICLR 2019, Cheng et al. NeuralPS 2019, Andrew Ilyas et al. ICML 2018, Bhagoji et al. ECCV 2018, Ilyas et al. arXiv:1804.08598 , Moon et al. ICML 2019. Andriushche el al. arXiv:1912.00049.

特点：利用多次查询估计出梯度。

缺点：直接将查询施加在黑盒模型上，未使用代理模型，查询复杂度较高。

(2) 基于模仿的对抗攻击

Papernot et al. arXiv:1605.07277, Papernot ACCV 2017, Ma arXiv 2020. Wei et al. CVPR2020

特点: 训练代理模型，数据标签来自于黑盒模型的输出，再攻击代理模型去生成对抗样本。

缺点：训练需要大量查询，且用不同模型生成的样本无法成功迁移。

清華大學 Tsinghua University

2.模拟器攻击:动机

论文： Chen Ma, Li Chen, and Jun-Hai Yong. Simulating Unknown Target Models for Query-Efficient Black-box Attacks. In Conference on Computer Vision and Pattern Recognition 2021 CVPR 2021, Virtual, https://arxiv.org/abs/2009.00960

idea 的诞生源于我对Bandits攻击[1]的代码的观察，因为目标是减少查询query。

论文正文中正式写作的motivation是：现有的模型窃取攻击在训练一个替身模型的时候，需要查询目标模型。这仍然导致大量的查询，且可以被检测和防御。

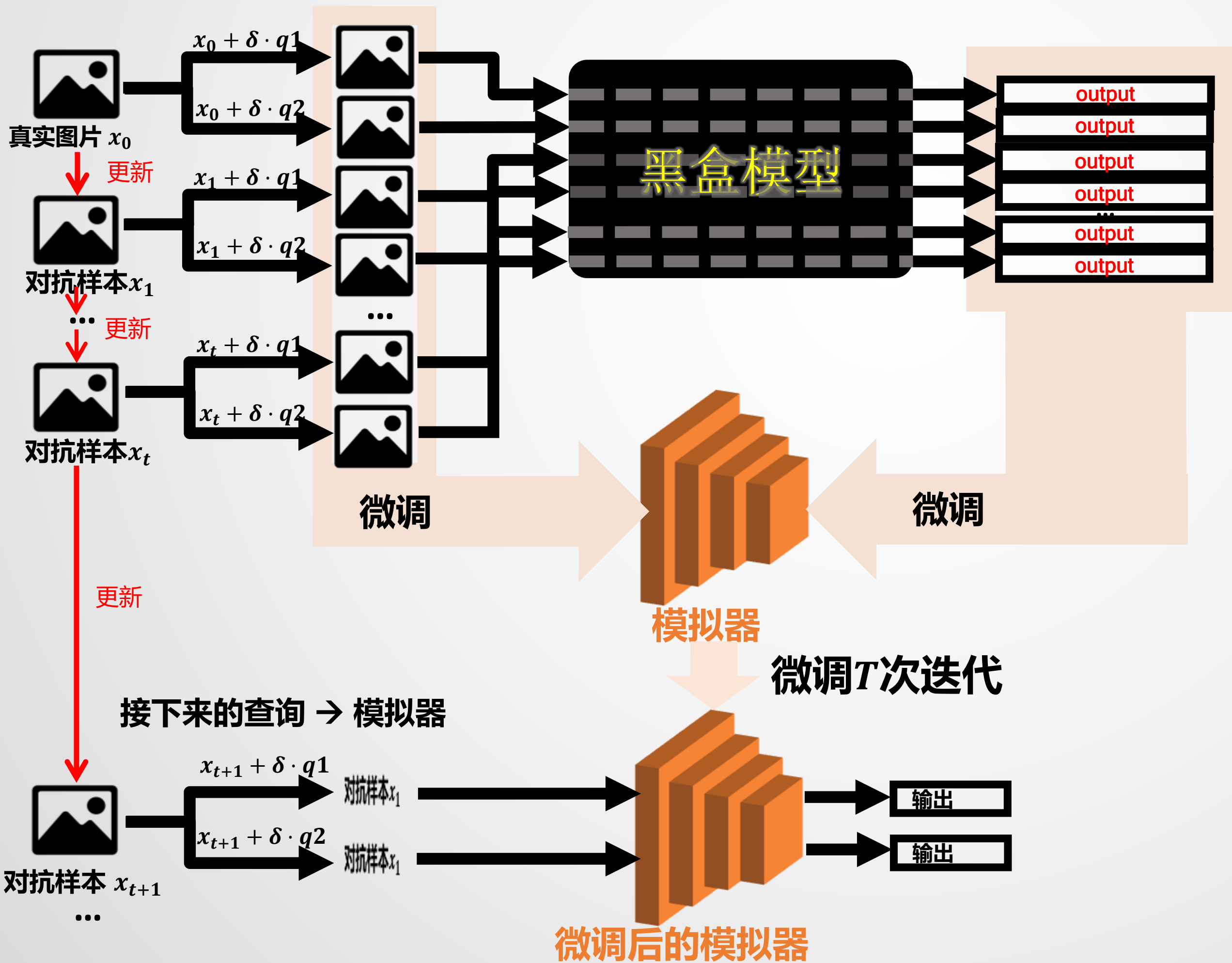
Bandits攻击的论文:

[1] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In International Conference on Learning Representations, 2019

```
for step_index in range(args.max_queries // 2):
    # Create noise for exporation, estimate the gradient, and take a PGD step
    exp_noise = args.exploration * torch.randn_like(prior) / (dim ** 0.5)  # parameterizes the exploration to be done around the prior
    # Query deltas for finite difference estimator
    exp_noise = exp_noise.cuda()
    q1 = upsampler(prior + exp_noise)  # 这就是Finite Difference算法， prior相当于论文里的v，这个prior也会更新，把梯度累积上去
        q2 = upsampler(prior - exp_noise)   # prior 相当于累积的更新量，用这个更新量，再去修改image，就会变得非常准
        # Loss points for finite difference estimator
    q1_images = adv_images + args.fd_eta * q1 / self.norm(q1)
    q2_images = adv_images + args.fd_eta * q2 / self.norm(q2)
    with torch.no_grad():
        q1_logits = target_model(q1_images)
        q2_logits = target_model(q2_images)
    l1 = criterion(q1_logits, true_labels, target_labels)
    l2 = criterion(q2_logits, true_labels, target_labels)
    # Finite differences estimate of directional derivative
    est_deriv = (l1 - l2) / (args.fd_eta * args.exploration)  # 方向导数 , l1和l2是loss
    # 2-query gradient estimate
    est_grad = est_deriv.view(-1, 1, 1, 1) * exp_noise  # B, C, H, W,
    # Update the prior with the estimated gradient
    prior = prior_step(prior, est_grad, args.online_lr)  # 注意，修正的是prior,这就是bandit算法的精髓
        grad = upsampler(prior)  # prior相当于梯度
        ## Update the image:
    # take a pgd step using the prior
    adv_images = image_step(adv_images, grad * correct.view(-1, 1, 1, 1), args.image_lr)  # prior放大后相当于累积的更新量，可以用来更新
        adv_images = proj_step(adv_images)
    adv_images = torch.clamp(adv_images, 0, 1)
```

2. 模拟器攻击

模拟器攻击(攻击过程)



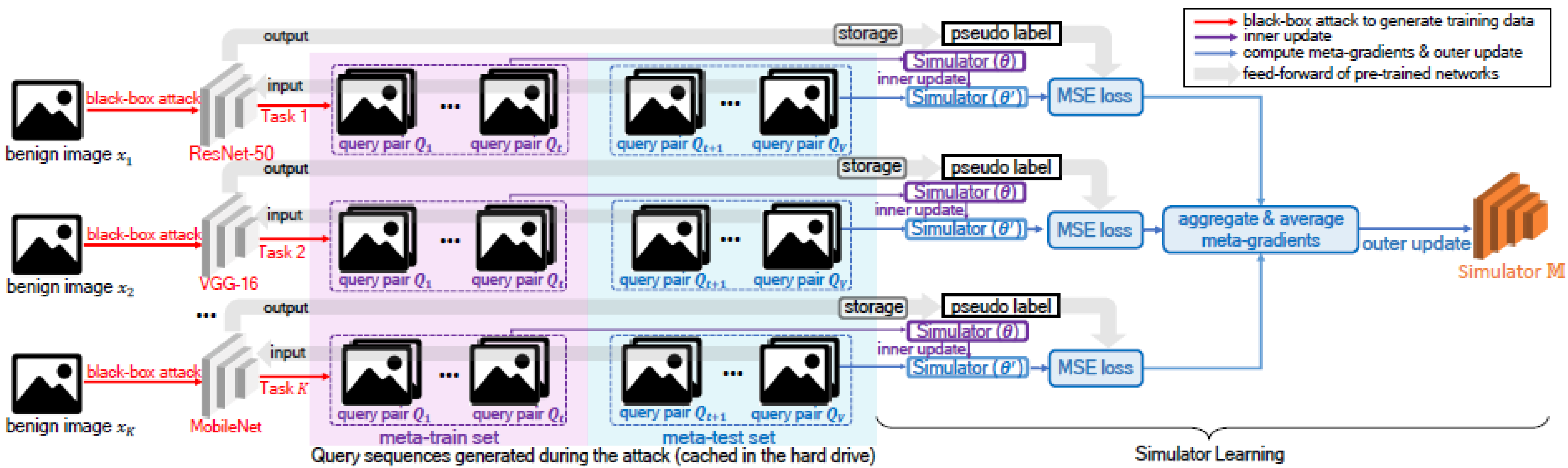
创新点：

1. 模拟器仅需少量样本微调即可模拟**任何**模型。
2. 大部分查询被迁移到元模拟器，减少查询。
3. 充分利用黑盒模型的查询反馈。

Tsinghua University

2. 模拟器攻击

模拟器攻击(模拟器的训练过程)

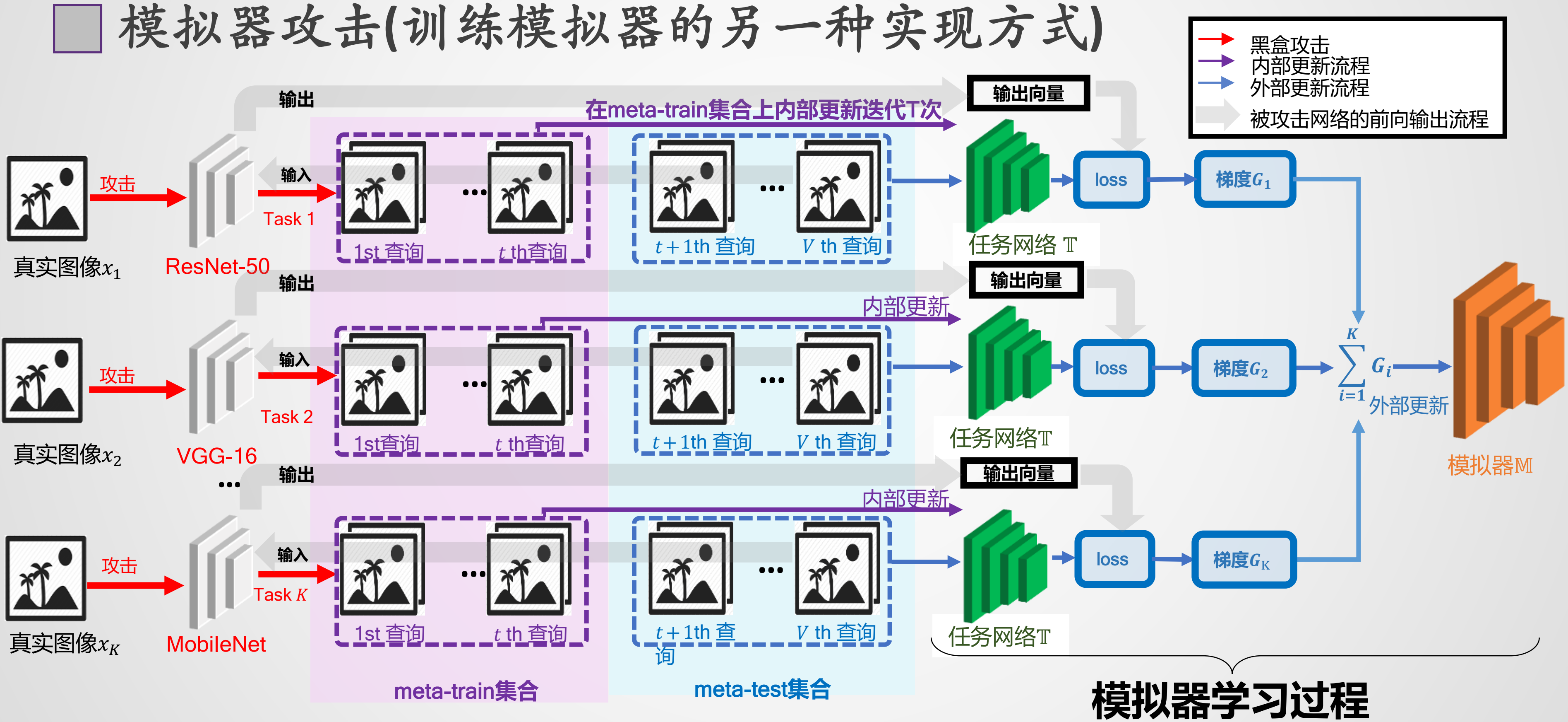


MSE损失函数

$$\mathcal{L}(\hat{\mathrm{p}}, \mathrm{p}) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{\mathrm{p}}_{Q_{i,1}} - \mathrm{p}_{Q_{i,1}}\right)^2 + \frac{1}{n}\sum_{i=1}^{n}\left(\hat{\mathrm{p}}_{Q_{i,2}} - \mathrm{p}_{Q_{i,2}}\right)^2 \tag{1}$$

Tsinghua University 清華大學

模拟器攻击(训练模拟器的另一种实现方式)



1.结合元学习，收集各种已有网络组成task训练数据。

2. 一个task：一个网络的数据。

3.T和M交替更新：T专注学习每个task，M学习跨越task泛化能力。

模拟器

Algorithm 1 Training procedure of the Simulator

Input: Training dataset D, Bandits attack algorithm $\mathcal{A}$, pre-trained classification networks $\mathbb{N}_1, \ldots, \mathbb{N}_n$, the Simulator network $\mathbb{M}$ and its parameters θ, feed-forward function f of $\mathbb{M}$, loss function $\mathcal{L}(\cdot, \cdot)$ defined in Eq. (1).

Parameters: Training iterations N, query sequence size V, meta-train set size t, batch size K, inner-update learning rate λ_1, outer-update learning rate λ_2, inner-update iterations T.

Output: The learned Simulator $\mathbb{M}$.

for $iter \leftarrow 1$ to N **do**
  sample K benign images $x_1, \ldots, x_K$ from D
  for $k \leftarrow 1$ to K **do** ▷ iterate over K tasks
    a network $\mathbb{N}_i \leftarrow$ sample from $\mathbb{N}_1, \ldots, \mathbb{N}_n$
    $Q_1, \ldots, Q_V \leftarrow \mathcal{A}(x_k, \mathbb{N}_i)$ ▷ query sequence
    $\mathcal{D}_{mtr} \leftarrow Q_1, \ldots, Q_t$
    $\mathcal{D}_{mte} \leftarrow Q_{t+1}, \ldots, Q_V$
    $\mathbf{p}_{\mathtt{train}} \leftarrow \mathbb{N}_i(\mathcal{D}_{mtr})$
    $\mathbf{p}_{\mathtt{test}} \leftarrow \mathbb{N}_i(\mathcal{D}_{mte})$ ▷ pseudo labels
    $\theta' \leftarrow \theta$ ▷ reinitialize $\mathbb{M}$'s weights
    for $j \leftarrow 1$ to T **do**
      $\theta' \leftarrow \theta' - \lambda_1 \cdot \nabla_{\theta'} \mathcal{L}\left(f_{\theta'}\left(\mathcal{D}_{mtr}\right), \mathbf{p}_{\mathtt{train}}\right)$
    end for
    $L_i \leftarrow \mathcal{L}\left(f_{\theta'}\left(\mathcal{D}_{mte}\right), \mathbf{p}_{\mathtt{test}}\right)$
  end for
  $\theta \leftarrow \theta - \lambda_2 \cdot \frac{1}{K} \sum_{i=1}^{K} \nabla_\theta L_i$ ▷ the outer update
end for
return $\mathbb{M}$

清華大学
Tsinghua University

文

□ 模拟器攻击(攻击

Algorithm 2 Simulator Attack under the ℓ_p norm constraint

Input: Input image $x \in \mathbb{R}^D$ where D is the image dimensionality, true label y of x, feed-forward function f of target model, Simulator $\mathbb{M}$, attack objective loss $\mathcal{L}(\cdot, \cdot)$.
Parameters: Warm-up iterations t, simulator-predict interval m, Bandits exploration τ, finite difference probe δ, OCO learning rate η_g, image learning rate η.
Output: x_{adv} that satisfies $\|x_{\text{adv}} - x\|_p \leq \epsilon$.

Initialize the adversarial example $x_{\text{adv}} \leftarrow x$
Initialize the gradient to be estimated $\mathbf{g} \leftarrow \mathbf{0}$
Initialize $\mathbb{D} \leftarrow deque(maxlen = t)$ ▷ a bounded double-ended queue with maximum length of t, adding a full $\mathbb{D}$ leads it to drop its oldest item automatically.
for $i \leftarrow 1$ to N **do**
  $\mathbf{u} \leftarrow \mathcal{N}(\mathbf{0}, \frac{1}{D}\mathbf{I})$ ▷ the same dimension with x
  $q1 \leftarrow \mathbf{g} + \tau\mathbf{u}, \quad q2 \leftarrow \mathbf{g} - \tau\mathbf{u}$
  $q1 \leftarrow q1/\|q1\|_2, \quad q2 \leftarrow q2/\|q2\|_2$
  if $i \leq t$ or $(i - t) \bmod m = 0$ **then**
    $\hat{y}_1 \leftarrow f(x_{\text{adv}} + \delta \cdot q1)$
    $\hat{y}_2 \leftarrow f(x_{\text{adv}} + \delta \cdot q2)$
    $\{x_{\text{adv}} + \delta \cdot q1, \hat{y}_1, x_{\text{adv}} + \delta \cdot q2, \hat{y}_2\}$ append $\mathbb{D}$
    if $i \geq t$ **then**
      Fine-tune $\mathbb{M}$ using $\mathbb{D}$ ▷ fine-tune $\mathbb{M}$ every m iterations after the warm-up phase.
    end if
  else
    $\hat{y}_1 \leftarrow \mathbb{M}(x_{\text{adv}} + \delta \cdot q1), \quad \hat{y}_2 \leftarrow \mathbb{M}(x_{\text{adv}} + \delta \cdot q2)$
  end if
  $\Delta_g \leftarrow \frac{\mathcal{L}(\hat{y}_1, y) - \mathcal{L}(\hat{y}_2, y)}{\tau\delta}\mathbf{u}$
  if $p = 2$ **then**
    $\mathbf{g} \leftarrow \mathbf{g} + \eta_g \cdot \Delta_g$
    $x_{\text{adv}} \leftarrow \prod_{\mathcal{B}_2(x,\epsilon)}(x_{\text{adv}} + \eta \cdot \frac{\mathbf{g}}{\|\mathbf{g}\|_2})$ ▷ $\prod_{\mathcal{B}_p(x,\epsilon)}$ denotes the ℓ_p norm projection under ℓ_p norm bound.
  else if $p = \infty$ **then** ▷ using the exponentiated gradient update [20] in the ℓ_∞ norm attack as follows.
    $\hat{\mathbf{g}} \leftarrow \frac{\mathbf{g}+\mathbf{1}}{2}$
    $\mathbf{g} \leftarrow \frac{\hat{\mathbf{g}} \cdot \exp(\eta_g \cdot \Delta_g) - (\mathbf{1} - \hat{\mathbf{g}}) \cdot \exp(-\eta_g \cdot \Delta_g)}{\hat{\mathbf{g}} \cdot \exp(\eta_g \cdot \Delta_g) + (\mathbf{1} - \hat{\mathbf{g}}) \cdot \exp(-\eta_g \cdot \Delta_g)}$
    $x_{\text{adv}} \leftarrow \prod_{\mathcal{B}_\infty(x,\epsilon)}(x_{\text{adv}} + \eta \cdot \text{sign}(\mathbf{g}))$
  end if
  $x_{\text{adv}} \leftarrow \text{Clip}(x_{\text{adv}}, 0, 1)$
end for
return x_{adv}

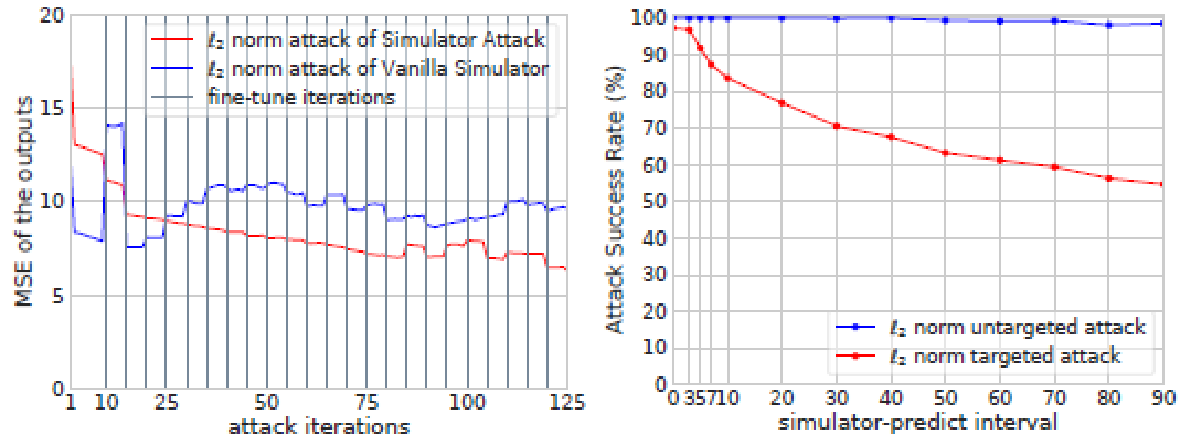
实验结果

Target Model	Method	Avg. Query	Med. Query	Max Query	Success Rate
PyramidNet-272	Rnd_init Simulator	105	52	1470	100%
	Vanilla Simulator	102	52	1374	100%
	Simulator Attack	92	52	834	100%

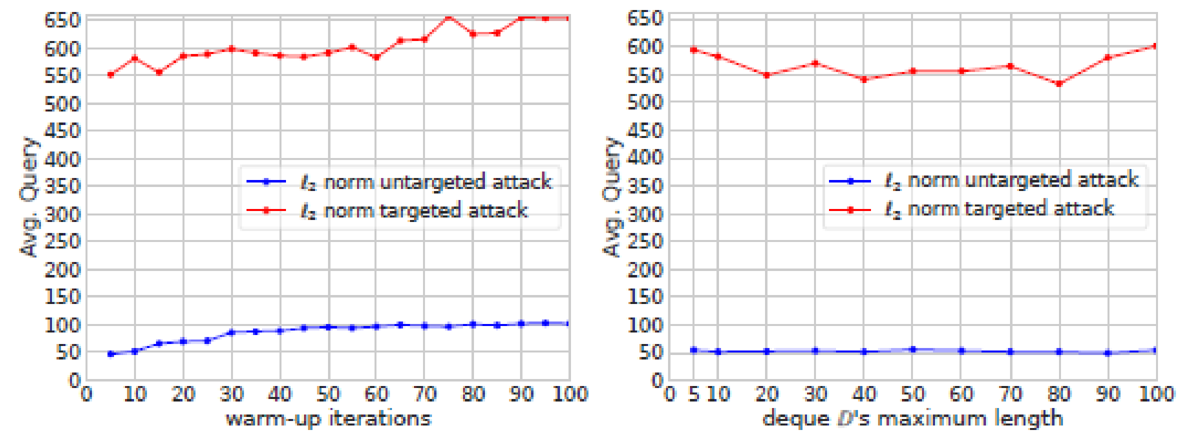
Table 2: Comparison of different simulators by performing ℓ_2 norm attack on the CIFAR-10 dataset. The Rnd_init Simulator uses an untrained ResNet-34 as the simulator; the Vanilla Simulator uses a ResNet-34 that is trained without using meta-learning as the simulator.

研究背景

- 实验



(a) simulation's precision study (b) simulator-predict interval



(c) warm-up study (d) deque's maximum length

Figure 3: We conduct ablation studies of the simulation's precision, simulator-predict interval, warm-up iterations, and deque 𝔻's maximum length by attacking a WRN-28 model in the CIFAR-10 dataset. The results indicate the following: (1) the meta training is beneficial for achieving an accurate simulation (Fig. 3a). (2) a difficult attack (e.g.

模拟器攻击：非目标攻击

实验结果：攻击普通的分类模型:PyramidNet-272,GDAS,WRN-28,WRN-40

Dataset	Norm	Attack	Attack Success Rate				Avg. Query				Median Query			
			PyramidNet-272	GDAS	WRN-28	WRN-40	PyramidNet-272	GDAS	WRN-28	WRN-40	PyramidNet-272	GDAS	WRN-28	WRN-40
CIFAR-10	ℓ_2	NES [19]	99.5%	74.8%	99.9%	99.5%	200	123	159	154	150	100	100	100
		RGF [31]	100%	100%	100%	100%	216	168	153	150	204	152	102	152
		P-RGF [8]	100%	100%	100%	100%	**64**	40	76	73	62	**20**	64	64
		Meta Attack [12]	99.2%	99.4%	98.6%	99.6%	2359	1611	1853	1707	2211	1303	1432	1430
		Bandits [20]	100%	100%	100%	100%	151	66	107	98	110	54	80	78
		Simulator Attack	100%	100%	100%	100%	92	**34**	**48**	**51**	**52**	26	**34**	**34**
	ℓ_∞	NES [19]	86.8%	71.4%	74.2%	77.5%	1559	628	1235	1209	600	300	400	400
		RGF [31]	99%	93.8%	98.6%	98.8%	955	646	1178	928	668	460	663	612
		P-RGF [8]	97.3%	97.9%	97.7%	98%	**742**	337	703	564	**408**	128	236	217
		Meta Attack [12]	90.6%	98.8%	92.7%	94.2%	3456	2034	2198	1987	2991	1694	1564	1433
		Bandits [20]	99.6%	100%	99.4%	99.9%	1015	391	611	542	560	166	224	228
		Simulator Attack	96.5%	99.9%	98.1%	98.8%	779	**248**	**466**	**419**	469	**83**	**186**	**186**
CIFAR-100	ℓ_2	NES [19]	92.4%	90.2%	98.4%	99.6%	118	94	102	105	100	50	100	100
		RGF [31]	100%	100%	100%	100%	114	110	106	106	102	101	102	102
		P-RGF [8]	100%	100%	100%	100%	54	46	54	73	62	62	62	62
		Meta Attack [12]	99.7%	99.8%	99.4%	98.4%	1022	930	1193	1252	783	781	912	913
		Bandits [20]	100%	100%	100%	100%	58	54	64	65	42	42	52	53
		Simulator Attack	100%	100%	100%	100%	**29**	**29**	**33**	**34**	**24**	**24**	**26**	**26**
	ℓ_∞	NES [19]	91.3%	89.7%	92.4%	89.3%	439	271	673	596	204	153	255	255
		RGF [31]	99.7%	98.8%	98.9%	98.9%	385	420	544	619	256	255	357	357
		P-RGF [8]	99.3%	98.2%	98%	97.8%	308	220	371	480	147	116	136	181
		Meta Attack [12]	99.7%	99.8%	97.4%	97.3%	1102	1098	1294	1369	912	911	1042	1040
		Bandits [20]	100%	100%	99.8%	99.8%	266	209	262	260	68	57	107	92
		Simulator Attack	100%	100%	99.9%	99.9%	**129**	**124**	**196**	**209**	**34**	**28**	**58**	**54**

Table 3: Experimental results of untargeted attack in CIFAR-10 and CIFAR-100 datasets.

清華大學
Tsinghua University

模拟器攻击:目标攻击

- 实验结果：攻击普通的分类模型:PyramidNet-272,GDAS,WRN-28,WRN-40

Dataset	Norm	Attack	Attack Success Rate				Avg. Query				Median Query			
			PyramidNet-272	GDAS	WRN-28	WRN-40	PyramidNet-272	GDAS	WRN-28	WRN-40	PyramidNet-272	GDAS	WRN-28	WRN-40
CIFAR-10	ℓ_2	NES [19]	93.7%	95.4%	98.5%	97.7%	1474	1515	1043	1088	1251	999	881	882
		Meta Attack [12]	92.2%	97.2%	74.1%	74.7%	4215	3137	3996	3797	3842	2817	3586	3329
		Bandits [20]	99.7%	100%	97.3%	98.4%	852	718	1082	997	458	538	338	399
		Simulator Attack (m=3)	99.1%	100%	98.5%	95.6%	896	718	990	980	373	**388**	217	249
		Simulator Attack (m=5)	97.6%	99.9%	96.4%	94%	**815**	**715**	**836**	**793**	**368**	400	**206**	**245**
	ℓ_∞	NES [19]	63.8%	80.8%	89.7%	88.8%	4355	3942	3046	3051	3717	3441	2535	2592
		Meta Attack [12]	75.6%	95.5%	59%	59.8%	4960	3461	3873	3899	4736	3073	3328	3586
		Bandits [20]	84.5%	98.3%	76.9%	79.8%	2830	1755	2037	2128	2081	1162	1178	1188
		Simulator Attack (m=3)	80.9%	97.8%	83.1%	82.2%	2655	1561	1855	1806	1943	918	1010	1018
		Simulator Attack (m=5)	78.7%	96.5%	80.8%	80.3%	**2474**	**1470**	**1676**	**1660**	**1910**	**917**	**957**	**956**
CIFAR-100	ℓ_2	NES [19]	87.6%	77%	89.3%	87.6%	1300	1405	1383	1424	1102	1172	1061	1049
		Meta Attack [12]	86.1%	88.7%	63.4%	43.3%	4000	3672	4879	4989	3457	3201	4482	4865
		Bandits [20]	99.6%	100%	98.9%	91.5%	1442	847	1645	2436	1058	679	1150	1584
		Simulator Attack (m=3)	99.3%	100%	98.6%	92.6%	921	724	1150	1552	666	519	779	1126
		Simulator Attack (m=5)	97.8%	99.6%	95.7%	83.9%	**829**	**679**	**1000**	**1211**	**644**	**508**	**706**	**906**
	ℓ_∞	NES [19]	72.1%	66.8%	68.4%	69.9%	4673	5174	4763	4770	4376	4832	4357	4508
		Meta Attack [12]	80.4%	81.2%	57.6%	40.1%	4136	3951	4893	4967	3714	3585	4609	4737
		Bandits [20]	81.2%	92.5%	72.4%	56%	3222	2798	3353	3465	2633	2132	2766	2774
		Simulator Attack (m=3)	89.4%	94.2%	79%	64.3%	2732	2281	3078	3238	1854	1589	2185	2548
		Simulator Attack (m=5)	83.7%	91.4%	74.2%	60%	**2410**	**2134**	**2619**	**2823**	**1754**	**1572**	**2080**	**2270**

Table 4: Experimental results of targeted attack in CIFAR-10 and CIFAR-100 datasets, where m is simulator-predict interval.

- 实验结果：攻击普通的分类模型:DenseNet121,ResNeXT-101(32x4d),ResNeXT-101(64x4d)

Attack	Attack Success Rate			Avg. Query			Median Query		
	D_{121}	R_{32}	R_{64}	D_{121}	R_{32}	R_{64}	D_{121}	R_{32}	R_{64}
NES [19]	74.3%	45.3%	45.5%	1306	2104	2078	510	765	816
RGF [31]	96.4%	85.3%	87.4%	1146	2088	2087	667	1280	1305
P-RGF [8]	94.5%	83.9%	85.9%	883	1583	1581	448	**657**	**690**
Meta Attack [12]	71.1%	33.8%	36%	3789	4101	4012	3202	3712	3649
Bandits [20]	99.2%	94.1%	95.3%	964	1737	1662	520	954	1014
Simulator Attack	99.4%	96.8%	97.9%	**811**	**1380**	**1445**	**431**	850	878

Table 6: Experimental results of untargeted attack under ℓ_∞ norm in TinyImageNet dataset. D_{121}: DenseNet-121, R_{32}: ResNeXt-101 (32×4d), R_{64}: ResNeXt-101 (64×4d).

Attack	Attack Success Rate			Avg. Query			Median Query		
	D_{121}	R_{32}	R_{64}	D_{121}	R_{32}	R_{64}	D_{121}	R_{32}	R_{64}
NES [19]	88.5%	88%	88.2%	4625	4959	4758	4337	4703	4440
Meta Attack [12]	24.2%	21%	18.2%	5420	5440	5661	5506	5249	5250
Bandits [20]	85.1%	72.2%	72.4%	2724	3550	3542	1860	2700	2854
Simulator Attack	89.8%	84.9%	83.9%	**1959**	**2558**	**2488**	**1399**	**1966**	**1982**

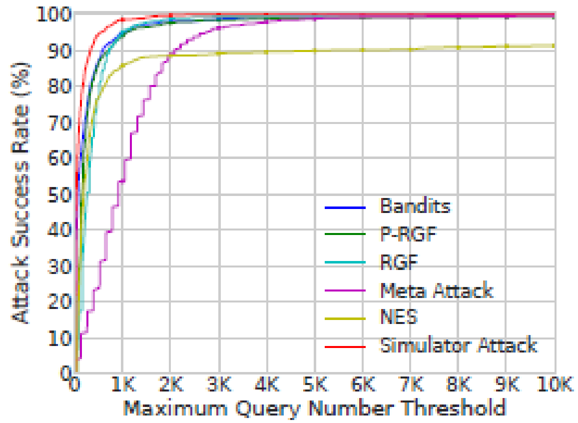
Table 7: Experimental results of targeted attack under ℓ_2 norm in TinyImageNet dataset. D_{121}: DenseNet-121, R_{32}: ResNeXt-101 (32×4d), R_{64}: ResNeXt-101 (64×4d).

模拟器攻击：攻击防御模型

实验结果(攻击防御模型)

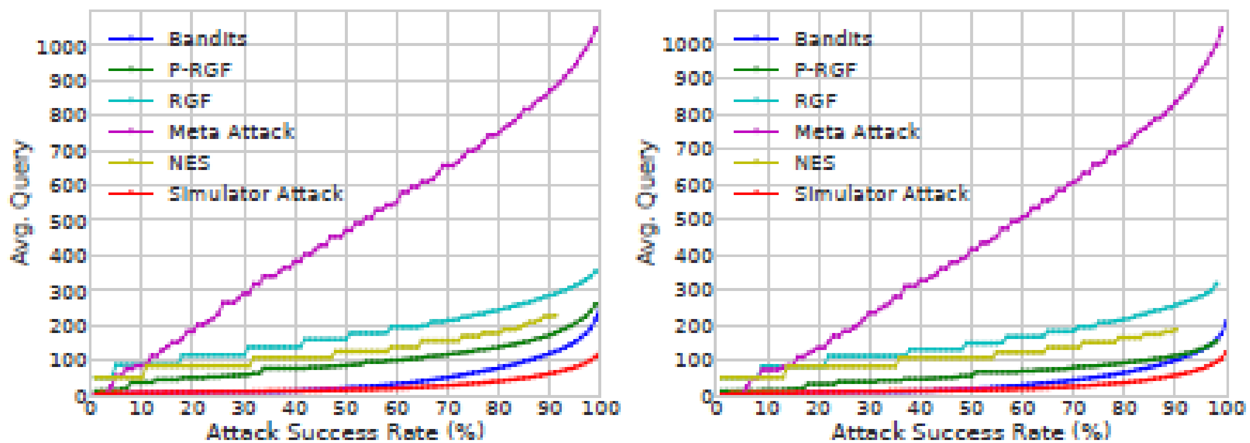
Dataset	Attack	Attack Success Rate				Avg. Query				Median Query			
		CD [21]	PCL [30]	FD [25]	Adv Train [28]	CD [21]	PCL [30]	FD [25]	Adv Train [28]	CD [21]	PCL [30]	FD [25]	Adv Train [28]
CIFAR-10	NES [19]	60.4%	65%	54.5%	16.8%	1130	728	1474	**858**	400	150	450	**200**
	RGF [31]	48.7%	82.6%	44.4%	22.4%	2035	1107	1717	973	1071	306	768	510
	P-RGF [8]	62.8%	80.4%	65.8%	22.4%	1977	1006	1979	1158	1038	230	703	602
	Meta Attack [12]	26.8%	77.7%	38.4%	18.4%	2468	1756	2662	1894	1302	1042	1824	1561
	Bandits [20]	44.7%	84%	55.2%	34.8%	786	776	832	1941	100	126	114	759
	Simulator Attack	54.9%	78.2%	60.8%	32.3%	**433**	**641**	**391**	1529	**46**	**116**	**50**	589
CIFAR-100	NES [19]	78.1%	87.9%	77.6%	23.1%	892	429	1071	**865**	300	150	250	**250**
	RGF [31]	50.2%	95.5%	62%	29.2%	1753	645	1208	1009	765	204	408	510
	P-RGF [8]	54.2%	96.1%	73.4%	28.8%	1842	679	1169	1034	815	182	262	540
	Meta Attack [12]	20.8%	93%	59%	27%	2084	1122	2165	1863	781	651	1043	1562
	Bandits [20]	54.1%	97%	72.5%	44.9%	786	321	584	1609	56	34	32	484
	Simulator Attack	72.9%	93.1%	80.7%	35.6%	**330**	**233**	**250**	1318	**30**	**22**	**24**	442
TinyImageNet	NES [19]	69.5%	73.1%	33.3%	23.7%	1775	863	2908	**945**	850	250	1600	**200**
	RGF [31]	31.3%	91.8%	9.1%	34.7%	2446	1022	1619	1325	1377	408	765	612
	P-RGF [8]	37.3%	91.8%	25.9%	34.4%	1946	1065	2231	1287	891	436	985	602
	Meta Attack [12]	4.5%	75.8%	3.7%	20.1%	1877	2585	4187	3413	912	1792	2602	2945
	Bandits [20]	39.6%	95.8%	12.5%	49%	893	909	1272	1855	85	206	193	810
	Simulator Attack	43%	84.2%	21.3%	42.5%	**377**	**586**	**746**	1631	**32**	**148**	**157**	632

Table 5: Experimental results after performing the ℓ_∞ norm attacks on defensive models, where CD represents ComDefend [21], FD is Feature Distillation [25], and PCL is prototype conformity loss [30].



(a) PyramidNet-272 in CIFAR-100 (b) R_{32} in TinyImageNet

Figure 4: Comparison of the attack success rate at different limited maximum queries in untargeted attack under ℓ_∞ norm, where R_{32} indicates ResNext-101 (32×4d).



(a) PyramidNet-272 in CIFAR-100 (b) GDAS in CIFAR-100

Figure 5: Comparisons of the average query at different success rates under the untargeted ℓ_∞ norm attack. More results are presented in the supplementary material.

模拟器攻击

遗留的问题（未来的工作）：

1. fine-tune较慢。

2.如何避免预训练（因为训练要先生成**query sequences**数据）？

清華大學
Tsinghua University

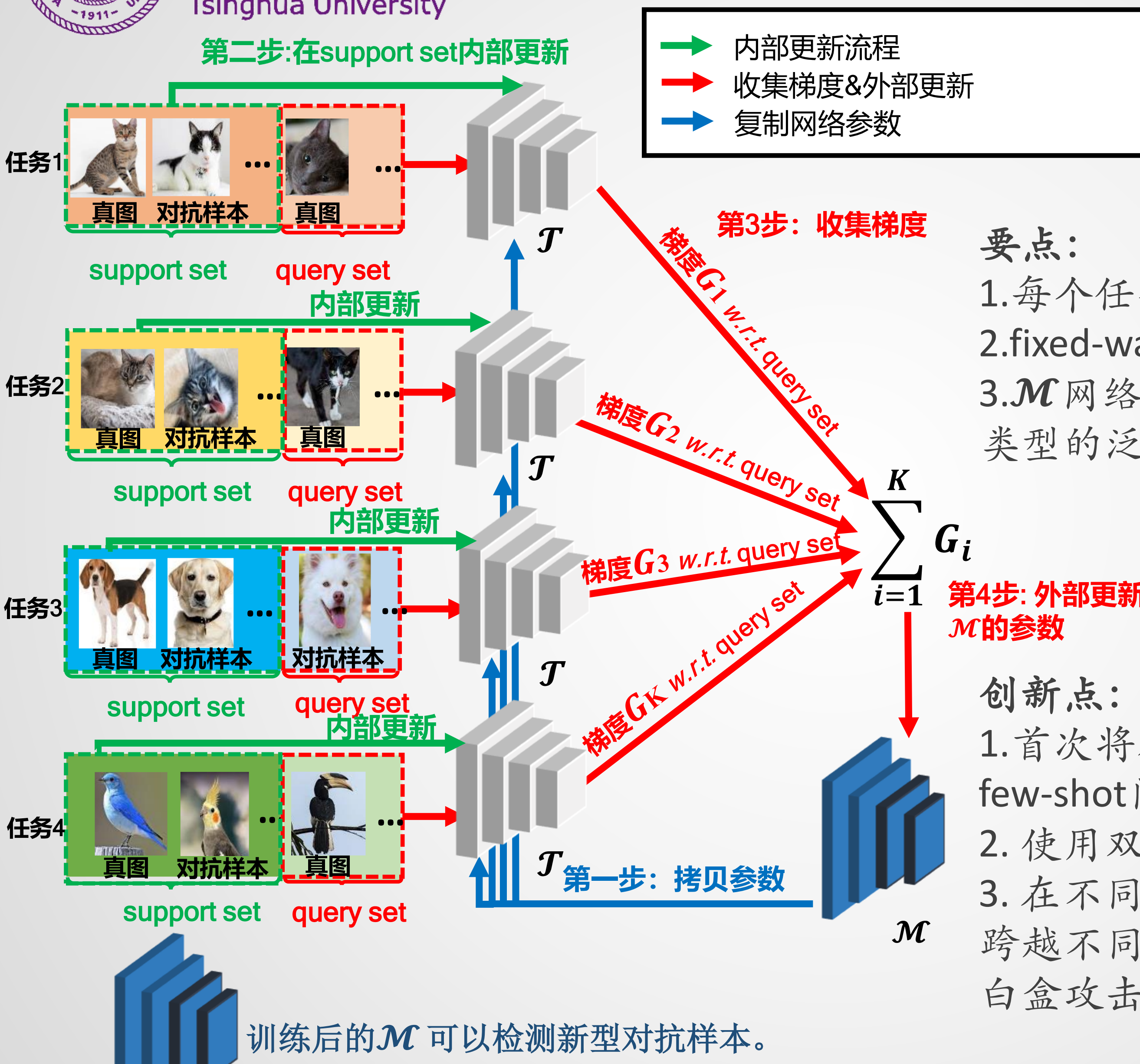
MetaAdvDet: ACM MM 2019

2.我们可以数据不足情况下检测新型对抗样本吗?

论文： Chen Ma, Chenxu Zhao, Hailin Shi, Li Chen, Junhai Yong, and Dan Zeng. MetaAdvDet: Towards Robust Detection of Evolving Adversarial Attacks. In Proceedings of the 27th ACM International Conference on Multimedia. Association for Computing Machinery, New York, NY, USA, 692–701. ACM MM 2019, Nice, France.

动机： 为了安全，区分出对抗样本和真实样本。但是
1. 新型对抗攻击数据标注成本高。
2. 收集样本速度慢。
3. 已有方法都需要上万样本训练。

创新点： 将这种检测问题定义为一个few-shot问题，提出基于元学习的检测方法:MetaAdvDet。
优点： 仅需几个标注新攻击样本，就可以检测。



要点：

1. 每个任务包含一种攻击类型。
2. fixed-way设置。
3. $\mathcal{M}$网络学到跨越不同任务和攻击类型的泛化检测能力。

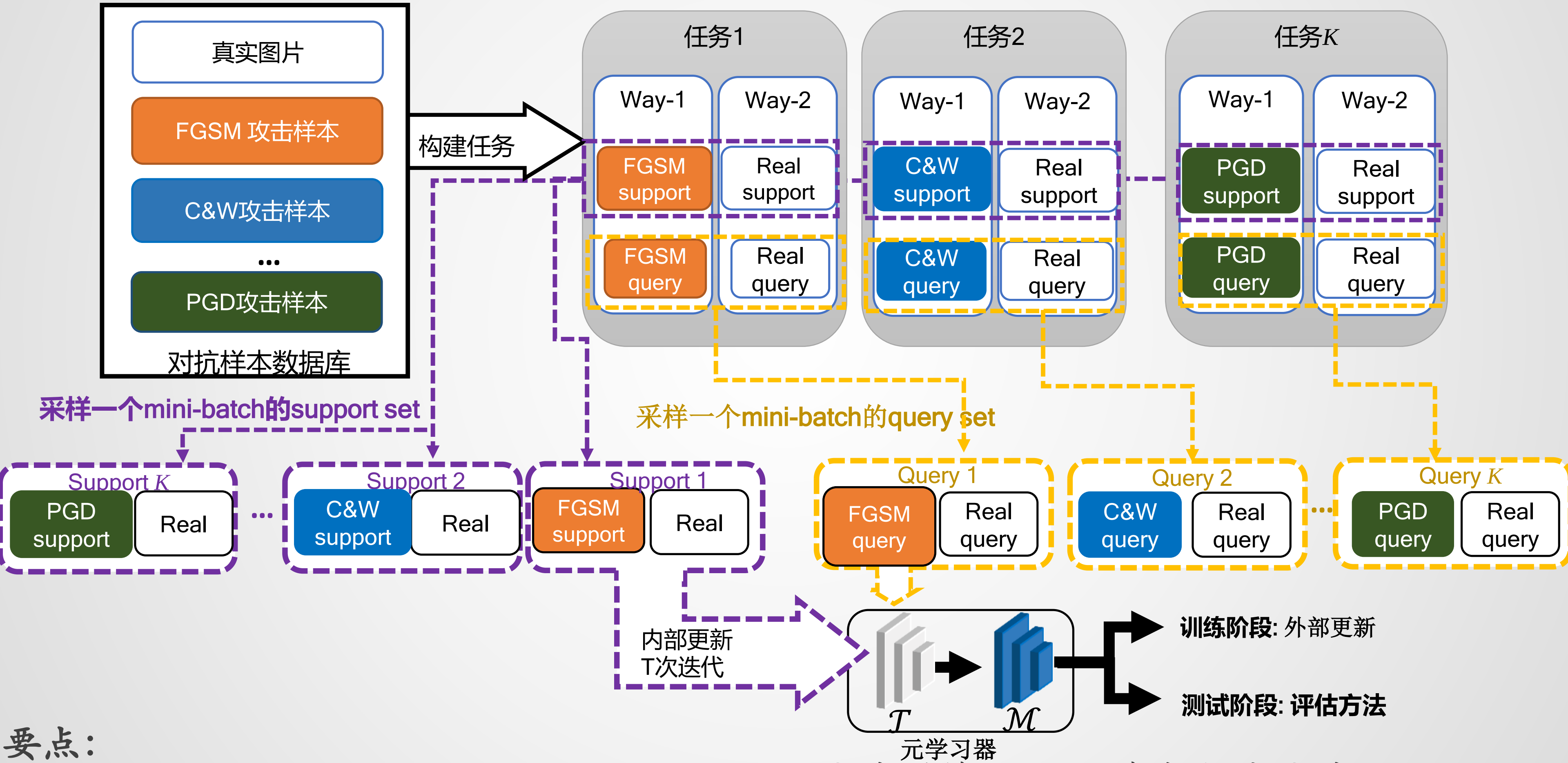
创新点：

1. 首次将对抗样本检测问题定位为few-shot问题。
2. 使用双网络结构检测对抗样本。
3. 在不同测试基准(跨越攻击类型，跨越不同数据库，跨越不同网络结构，白盒攻击）均取得领先性能。

图9. MetaAdvDet在一个mini-batch的学习过程

清華大學 Tsinghua University

MetaAdvDet



MetaAdvDet方法训练所需task数据切分方法

要点：

1.收集不同类型的攻击样本。

2.切分task，每个task一种攻击样本。

3. fixed-way设置，label 0：真实样本 label 1：对抗样本。

Algorithm 1 MetaAdvDet training procedure

Input: master network $\mathcal{M}$ and its parameters $\mathcal{M}_\theta$, task-dedicated network $\mathcal{T}$ and its parameters $\mathcal{T}_\theta$, the feed-forward function $f_{\mathcal{T}_\theta}$ of $\mathcal{T}$, max iterations N, inner-update learning rate λ_1, outer-update learning rate λ_2, inner updates iteration T, the multi-task format dataset $\mathcal{D}$, cross entropy loss function $\mathcal{L}$.

Output: the learned network $\mathcal{M}$.

for $iter \leftarrow 1$ to N **do**
  sample K tasks $\mathbb{T}_{i,i\in\{1,\cdots,K\}}$ from $\mathcal{D}$
  for $i \leftarrow 1$ to K **do**
    S_i and $Q_i \leftarrow$ support set and query set of $\mathbb{T}_i$
    $\mathcal{T}_\theta \leftarrow \mathcal{M}_\theta$ ▷ copy parameters from
    $\mathcal{T}_{\theta'} \leftarrow \mathcal{T}_\theta$ ▷ $\mathcal{T}_\theta$ will be used in the oute
    for $t \leftarrow 1$ to T **do**
      Calculate $\nabla_{\mathcal{T}_{\theta'}} \mathcal{L}(f_{\mathcal{T}_{\theta'}})$ by using S_i
      $\mathcal{T}_{\theta'} \leftarrow \mathcal{T}_{\theta'} - \lambda_1 \nabla_{\mathcal{T}_{\theta'}} \mathcal{L}(f_{\mathcal{T}_{\theta'}})$ ▷ inner
    end for
    $G_i \leftarrow \nabla_{\mathcal{T}_\theta} \mathcal{L}(f_{\mathcal{T}_{\theta'}})$ by using Q_i
  end for
  $\mathcal{M}_\theta \leftarrow \mathcal{M}_\theta - \lambda_2 \sum_{i=1}^{K} G_i$ ▷ oute
end for
return $\mathcal{M}$

$$\text{recall} = \frac{TP}{TP + FN}, \text{precision} = \frac{TP}{TP + FP}$$

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Algorithm 2 MetaAdvDet testing procedure

Input: master network $\mathcal{M}$ and its learned parameters $\mathcal{M}_\theta$, task-dedicated network $\mathcal{T}$ and its parameters $\mathcal{T}_\theta$, the feed-forward function $f_{\mathcal{T}_\theta}$ of $\mathcal{T}$, fine-tune iterations T, learning rate λ, test tasks $\mathbb{T}_{i,i\in\{i,\cdots,N\}}$ which is obtained by reorganizing the test set, cross entropy loss $\mathcal{L}$, ground truth $Y_{i,i\in\{i,\cdots,N\}}$ of the query set.

Output: the average F1 score over all tasks.

for $\mathbb{T}_i \leftarrow \mathbb{T}_1$ to $\mathbb{T}_N$ **do** ▷ iterate over all test tasks
  S_i and $Q_i \leftarrow$ support set and the query set of $\mathbb{T}_i$
  $\mathcal{T}_\theta \leftarrow \mathcal{M}_\theta$ ▷ copy parameters to ensure each task is tested independently
  for $t \leftarrow 1$ to T **do**
    Calculate $\nabla_{\mathcal{T}_\theta} \mathcal{L}(f_{\mathcal{T}_\theta})$ by using S_i
    $\mathcal{T}_\theta \leftarrow \mathcal{T}_\theta - \lambda \nabla_{\mathcal{T}_\theta} \mathcal{L}(f_{\mathcal{T}_\theta})$ ▷ fine-tune step
  end for
  $\hat{Y}_i \leftarrow f_{\mathcal{T}_\theta}(Q_i)$ ▷ get prediction of query set of task i
  $\text{score}_i \leftarrow \text{F1}(\hat{Y}_i, Y_i)$
end for
F1 score $\leftarrow \frac{1}{N} \sum_{i=1}^{N} score_i$
return F1 score

Tsinghua University

研

Benchmark	Test Protocols	
Datasets	CIFAR-10, MNIST and FashionMNIST	
Cross-Adversary Benchmark (simulate the situation of evolving attacks)	**Train Adversary**	**Test Adversary**
	FGSM, MI-FGSM, BIM, PGD, C&W, JSMA, SPSA, VAT, MaxConfidence	EAD, semantic, DeepFool, Spatial Transformation, NewtonFool
Cross-Domain Benchmark	**Train Domain**	**Test Domain**
	MNIST FashionMNIST	FashionMNIST MNIST
Cross-Architecture Benchmark (evaluate the detection of adversarial examples with new architecture)	**Train Architecture**	**Test Architecture**
	ResNet-10 ResNet-18 Conv-4 ResNet-10	ResNet-18 ResNet-10 ResNet-10 Conv-4
White-box benchmark	将分类器和检测器组合成一个大的分类器，再进行攻击，每种方法样本独立产生。	白盒攻击样本是测试阶段的样本。

MetaAdvDet的网络参数配置

name	default value	description
shots	1	number of examples in a way, MetaAdvDet should set the same shots in both training and testing.
ways	2	alias of class number, data of the same way come from using the same adversary to attack the same category's images.
train query set size	70	number of examples of a query set in training.
test query set size	30	number of examples of a query set in testing.
task number K	30	number of tasks in each mini-batch.
inner update times	12	iteration times of inner update during training
fine-tune times	20	iteration times of fine-tune during testing.
total tasks	20,000	total tasks in the constructed tasks.
inner learning rate	0.001	learning rate of inner update.
outer learning rate	0.0001	learning rate of outer update.
dataset	AdvCIFAR	the dataset for ablation study
backbone	conv-3	the backbone of MetaAdvDet & compared methods
benchmark	cross-adversary	the benchmark for ablation study

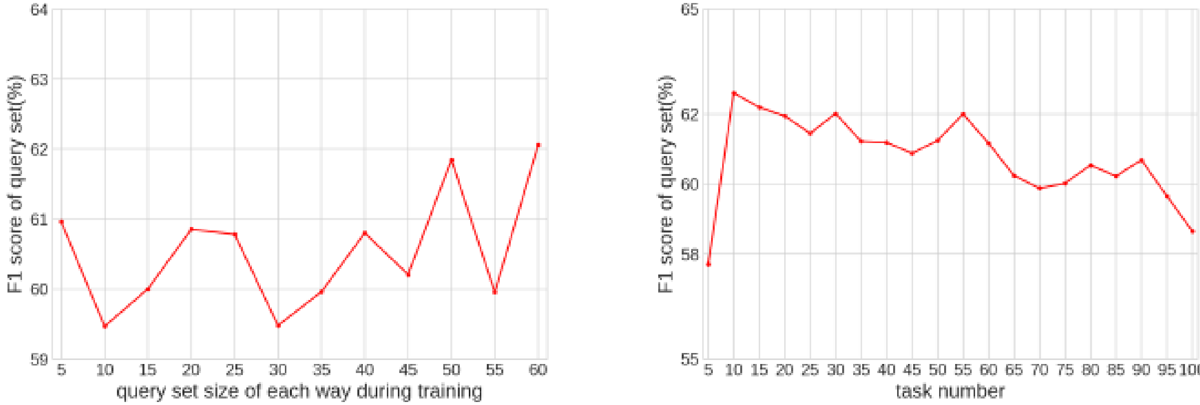
Tsinghua University

测试指标

$$\text{recall} = \frac{TP}{TP + FN}, \text{precision} = \frac{TP}{TP + FP}$$
$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2}$$

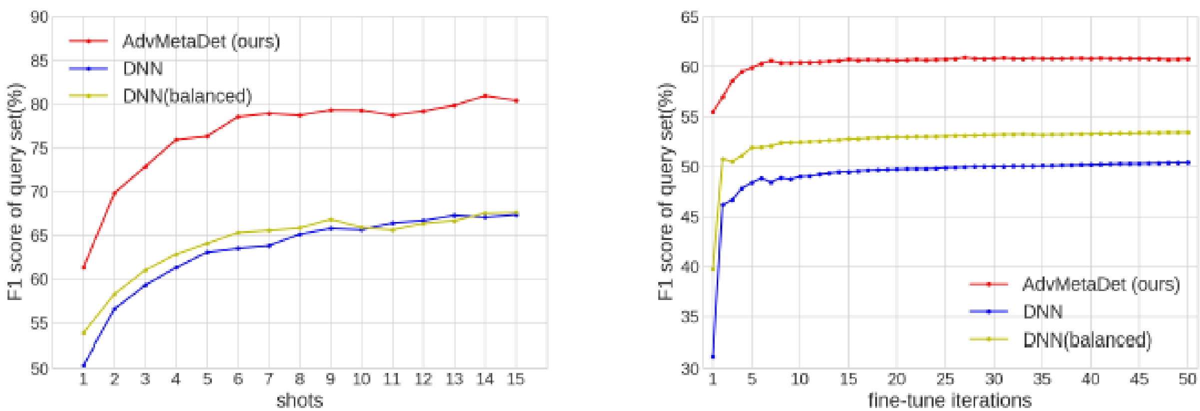
We use label 1 to represent the real example and 0 to represent the adversarial example, so TP is the number of correctly detected real examples, FN is the number of real examples that are incorrectly detected as adversarial examples, and FP is the number of adversarial images that are detected as real examples. Note that the final F1 score is obtained via averaging F1 scores of all tasks (Algorithm 2).

剥离实验



(a) train query set size study (b) task number K study

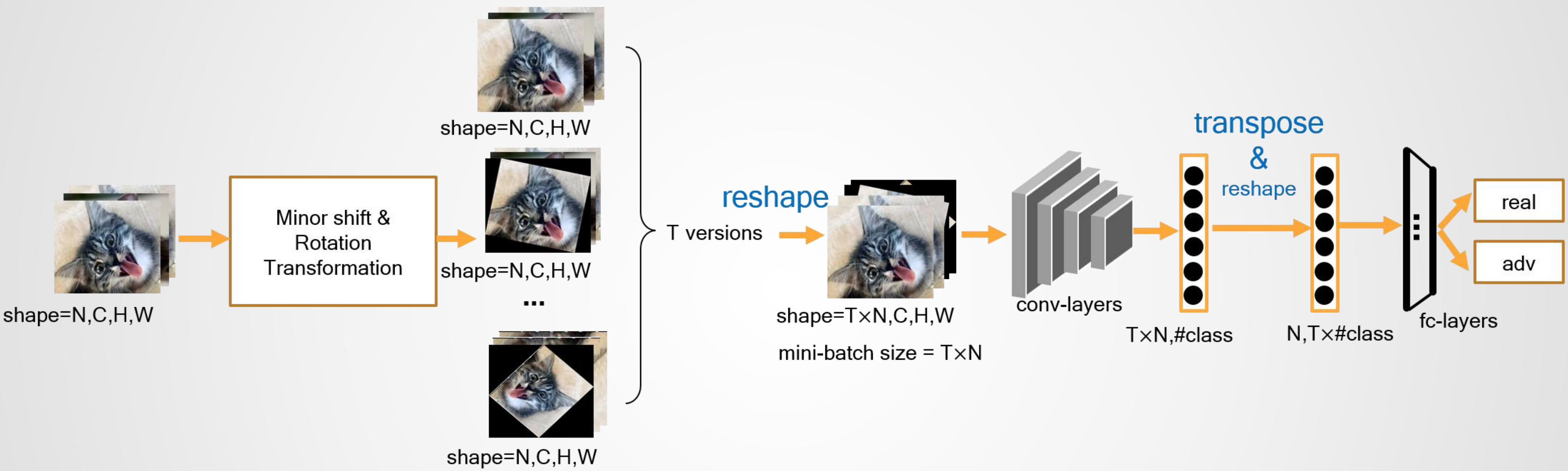
Figure 3: Ablation study results of train query set size and task number of a training mini-batch.



(a) shots study (b) fine-tune iterations study

Figure 4: Ablation study results of shots and fine-tune iterations. MetaAdvDet outperforms the baseline DNN and DNN (balanced) by a large margin.

对比方法：基于旋转的TransformDet



清華大学
Tsinghua University

Cross-Adversary实验结果

Dataset	Method	F1 score	
		1-shot	5-shot
AdvCIFAR	DNN	0.495	0.639
	DNN (balanced)	0.536	0.643
	NeuralFP [8]	**0.698**	0.700
	TransformDet [45]	0.662	0.697
	MetaAdvDet (ours)	0.685	**0.791**
AdvMNIST	DNN	0.812	0.852
	DNN (balanced)	0.797	0.808
	NeuralFP [8]	0.780	0.906
	TransformDet [45]	0.840	0.904
	MetaAdvDet (ours)	**0.987**	**0.993**
AdvFashionMNIST	DNN	0.782	0.885
	DNN (balanced)	0.744	0.850
	NeuralFP [8]	0.798	0.817
	TransformDet [45]	0.712	0.879
	MetaAdvDet (ours)	**0.848**	**0.944**

清華大學
Tsinghua University

Cross-Adversary实验结果:不同的对抗攻击的F1 score

Table 10: F1 score of representative adversaries on the Adv-CIFAR dataset, cross-adversary benchmark.

Dataset	Adversary	Method	F1 score 1-shot	F1 score 5-shot
AdvCIFAR	Spatial Transformation [49]	DNN	0.498	0.599
		DNN (balanced)	0.529	0.589
		NeuralFP [8]	0.708	0.696
		TransformDet [45]	0.633	0.660
		MetaAdvDet (ours)	**0.811**	**0.920**
	semantic [17]	DNN	0.488	0.644
		DNN (balanced)	0.529	0.657
		NeuralFP [8]	0.698	0.700
		TransformDet [45]	0.662	0.688
		MetaAdvDet (ours)	**0.763**	**0.855**
	NewtonFool [19]	DNN	0.511	0.664
		DNN (balanced)	0.542	0.670
		NeuralFP [8]	**0.696**	0.696
		TransformDet [45]	0.658	0.716
		MetaAdvDet (ours)	0.647	**0.735**

清華大学 Tsinghua University

Cross-Domain实验结果

Train Domain	Test Domain	Method	F1 score	
			1-shot	5-shot
AdvMNIST	AdvFashionMNIST	DNN (balanced)	0.698	0.813
		NeuralFP [8]	0.748	0.811
		TransformDet [45]	0.664	0.808
		MetaAdvDet (ours)	**0.799**	**0.870**
AdvFashionMNIST	AdvMNIST	DNN (balanced)	0.950	0.977
		NeuralFP [8]	0.775	0.836
		TransformDet [45]	0.934	0.940
		MetaAdvDet (ours)	**0.956**	**0.981**

Cross-Architecture实验结果

Table 12: F1 score of cross-architecture benchmark.

Dataset	Train Arch	Test Arch	Method	F1 score 1-shot	F1 score 5-shot
AdvCIFAR	ResNet-10	ResNet-18	NeuralFP [8]	0.713	0.709
			TransformDet [45]	0.758	0.880
			DNN (balanced)	0.702	0.768
			MetaAdvDet (ours)	**0.832**	**0.902**
	ResNet-18	ResNet-10	NeuralFP [8]	0.712	0.703
			TransformDet [45]	0.788	0.874
			DNN (balanced)	0.711	0.752
			MetaAdvDet (ours)	**0.840**	**0.889**
	conv-4	ResNet-10	NeuralFP [8]	0.712	0.703
			TransformDet [45]	0.763	0.868
			DNN (balanced)	0.723	0.779
			MetaAdvDet (ours)	**0.835**	**0.885**
	ResNet-10	conv-4	NeuralFP [8]	0.709	0.702
			TransformDet [45]	0.766	0.885
			DNN (balanced)	0.739	0.790
			MetaAdvDet (ours)	**0.854**	**0.918**
AdvMNIST	ResNet-10	ResNet-18	NeuralFP [8]	0.906	0.882
			TransformDet [45]	0.973	0.988
			DNN (balanced)	0.943	0.972
			MetaAdvDet (ours)	**0.984**	**0.993**
	ResNet-18	ResNet-10	NeuralFP [8]	0.894	0.738
			TransformDet [45]	0.967	0.990
			DNN (balanced)	0.912	0.953
			MetaAdvDet (ours)	**0.981**	**0.991**
	conv-4	ResNet-10	NeuralFP [8]	0.894	0.738
			TransformDet [45]	**0.972**	**0.985**
			DNN (balanced)	0.897	0.959
			MetaAdvDet (ours)	0.963	0.983
	ResNet-10	conv-4	NeuralFP [8]	0.917	0.961
			TransformDet [45]	0.984	0.992
			DNN (balanced)	0.958	0.978
			MetaAdvDet (ours)	**0.990**	**0.996**
AdvFashionMNIST	ResNet-10	ResNet-18	NeuralFP [8]	0.813	0.856
			TransformDet [45]	0.936	0.974
			DNN (balanced)	0.848	0.932
			MetaAdvDet (ours)	**0.960**	**0.979**
	ResNet-18	ResNet-10	NeuralFP [8]	0.820	0.838
			TransformDet [45]	0.935	0.972
			DNN (balanced)	0.829	0.918
			MetaAdvDet (ours)	**0.957**	**0.976**
	conv-4	ResNet-10	NeuralFP [8]	0.820	0.838
			TransformDet [45]	0.946	0.970
			DNN (balanced)	0.920	0.968
			MetaAdvDet (ours)	**0.946**	**0.975**
	ResNet-10	conv-4	NeuralFP [8]	0.817	0.911
			TransformDet [45]	0.945	0.979
			DNN (balanced)	0.886	0.945
			MetaAdvDet (ours)	**0.967**	**0.982**

清华大学 Tsinghua University

White-box attack Benchmark实验结果

Dataset	Method	I-FGSM Attack		C&W Attack	
		1-shot	5-shot	1-shot	5-shot
CIFAR-10	DNN (balanced)	0.466	0.537	0.459	0.527
	TransformDet [45]	**0.593**	**0.728**	0.443	0.502
	MetaAdvDet (ours)	0.553	0.633	**0.548**	**0.607**
MNIST	DNN (balanced)	0.857	0.956	0.814	0.913
	TransformDet [45]	0.864	0.952	0.775	0.893
	MetaAdvDet (ours)	**0.968**	**0.994**	**0.920**	**0.990**
FashionMNIST	DNN (balanced)	0.745	0.890	0.726	0.853
	TransformDet [45]	0.837	0.920	0.747	0.853
	MetaAdvDet (ours)	**0.849**	**0.963**	**0.882**	**0.967**

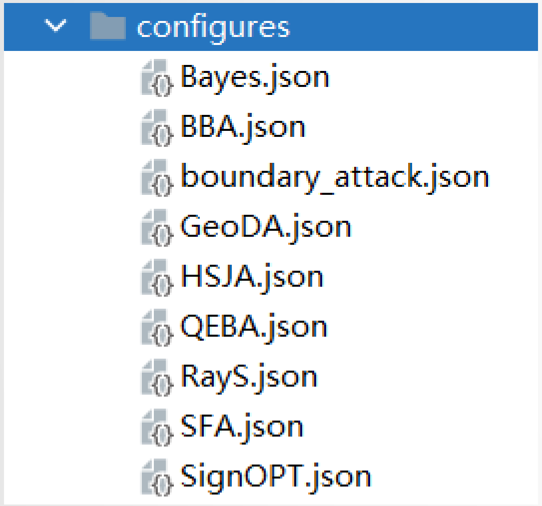
Method	DNN	NeuralFP [8]	TransformDet [45]	MetaAdvDet (ours)
Inference time (ms)	1.53 ± 0.01	2185.12 ± 18.10	69.17 ± 2.97	4.07 ± 4.40

BugTorch开源软件，目前收集了大量的黑盒攻击算法: bugtorch.org(还未上线)或https://github.com/machanic/bugtorch 包括score-based和decision-based setting

- adversarial_defense
- autozoom_attack
- bandits
- benign_image_classifier
- bundle_attack
- cifar_models
- cifar_models_myself
- configures
- corr_attack
- dataset
- frank_wolfe_attack(fail)
- imagenet_models
- LaMCTS
- LeBA
- meta_attack
- meta_simulator_bandits
- meta_simulator_benign_images
- meta_simulator_square_attack
- MGA_attack
- NES_attack

hard_label_attacks D:\work\hard_label_attacks

- adversarial_defense
- bayes_attack
- biased_boundary_attack
- boundary_attack
- cifar_models
- cifar_models_myself
- configures
- dataset
- evolutionary
- GeoDA
- hop_skip_jump_attack
- LogBarrier
- models
- OPT
- policy_driven_attack
- QEBA
- RayS
- sign_flip_attack


configures
Bayes.json
BBA.json
boundary_attack.json
GeoDA.json
HSJA.json
QEBA.json
RayS.json
SFA.json
SignOPT.json

Thank you!

Q&A