BREAKING CERTIFIED DEFENSES: SEMANTIC ADVERSARIAL EXAMPLES WITH SPOOFED ROBUSTNESS CERTIFICATES

<第 7 組>

Team Members：

408530003 資管四 范綱彥
408530004 資管四 潘甫翰 (組長)
408530007 資管四 謝瀞瑩
408530028 資管四 楊宗軒

Introduction

In summary, we consider methods that attack a certified classifier in the following sense:

- Imperceptibility: the adversarial example “looks like” its corresponding natural base example,
- Misclassification: the certified classifier assigns an incorrect label to the adversarial example
- Strongly certified: the certified classifier provides a strong/large-radius certificate for the adversarial example.


setopt(conn, CURLOPT_URL, url);
"Failed to set URL [%s]\n",
setopt(conn, CURLOPT_FOLLOWLOCATION,
"Failed to set redirect option [%s]\n",
(conn, CURLOPT_WRITEFUNCTION,
"Failed to set writer [%s]\n",

BACKGOUND

Background

White-box Attack

The attacker knows the victim's network and parameters

Background

Adversarial perturbation are often constructed using:

- first-order gradient information
- approximations of the gradient

Background

The prevailing formulation for crafting attacks uses an additive adversarial perturbation, and perceptibility is minimized using an l_p-norm constraint. For example:

- l_∞-bounded attacks limit how much each pixel can move
- l_0 adversarial attacks limit the number of pixels that can be modified

Background

craft imperceptible attacks without using l_p bounds:

- shifting color channels
- Wasserstein ball/distance
- rotation and translation

Certifiable Defenses

Certifiable Defenses

Certified defenses, on the other-hand, provably make networks resist l_p-bounded perturbations of a certain radius. Both of these defenses produce a class label, and also a guarantee that the image could not have been crafted by making small perturbations to an image of a different label. Certified defenses can also benefit from adversarial training. For instance:

- randomized smoothing (Cohen et al., 2019) is a certifiable defense against l_2-norm bounded attacks
- CROWN-IBP (Zhang et al., 2019b) is a certifiable defense against l_∞-norm bounded perturbations

PGD Attack

easy_setopt(conn, CURLOPT_URL, url);
"Failed to set URL [%s]\n",
easy_setopt(conn, CURLOPT_FOLLOWLOCATION,
!= CURLE_OK)
"Failed to set redirect option [%s]\n",
return false;
CURLOPT_WRITEFUNCTION,
"Failed to set writer [%s]\n",

PGD Attack

PGD attack, which creates adversarial images by modifying a clean base image. Given a loss function L and an l_p-norm bound ϵ for some $p \geq 0$, PGD attacks solve the following optimization problem:

$$\max_{\delta} L(\theta, x + \delta)$$

$$s.t. \ ||\delta||_p \leq \epsilon,$$

- θ: network parameters
- δ: adversarial perturbation to be added to the clean input image x

Shadow Attack

Shadow Attack

Shadow Attack is a hybrid model that allows various kinds of attacks to be compounded together, resulting in perturbations of large radii. It can be seen as the generalization of the well-known PGD attack. We solve the following problem with a range of penalties:

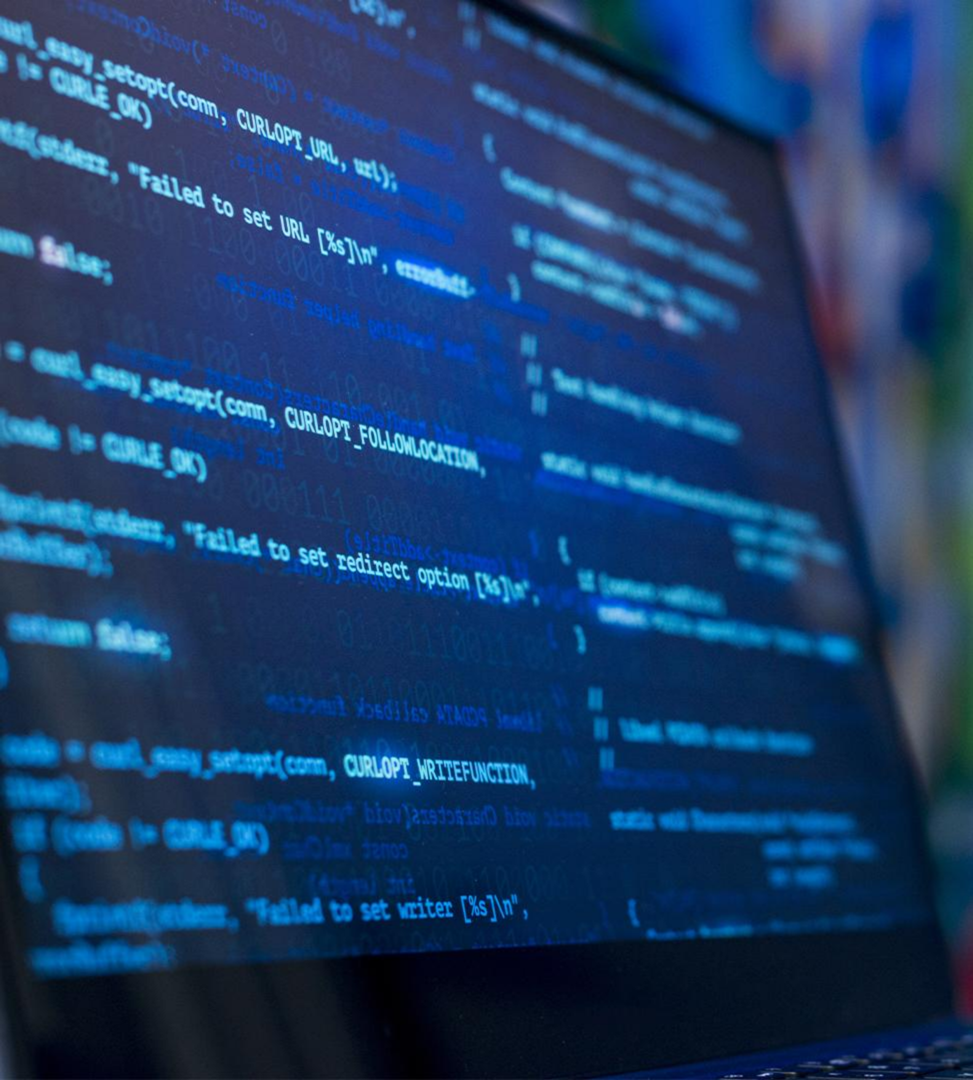
$$\max_{\delta} L(\theta, x+\delta) - \lambda_c C(\delta) - \lambda_{tv} TV(\delta) - \lambda_s Dissim(\delta)$$

- λ_c, λ_{tv}, λ_s: scalar penalty weights
- TV(δ): forces the perturbation δ to have small total variation (TV), and so appear more smooth and natural
- C(δ): limits the perturbation δ globally by constraining the change in the mean of each color channel c
- Dissim(δ): promotes perturbations δ that assume similar values in each color channel.

Shadow Attack

We suggest two ways of enforcing such similarity between RGB channels and we find both of them effective:

- **1-channel attack** strictly enforces $\delta_{R,i} \approx \delta_{G,i} \approx \delta_{B,i}, \forall i$ by using just one array to simultaneously represent each color channel $\delta_{W \times H}$. On the forward pass, we duplicate δ to make a 3-channel image. In this case, $Dissim(\delta) = 0$, and the perturbation is greyscale.
- **3-channel attack** uses a 3-channel perturbation $\delta_{3 \times W \times H}$, along with the dissimilarity metric $Dissim(\delta) = \|\delta_R - \delta_B\|_p + \|\delta_R - \delta_G\|_p + \|\delta_B - \delta_G\|_p$.



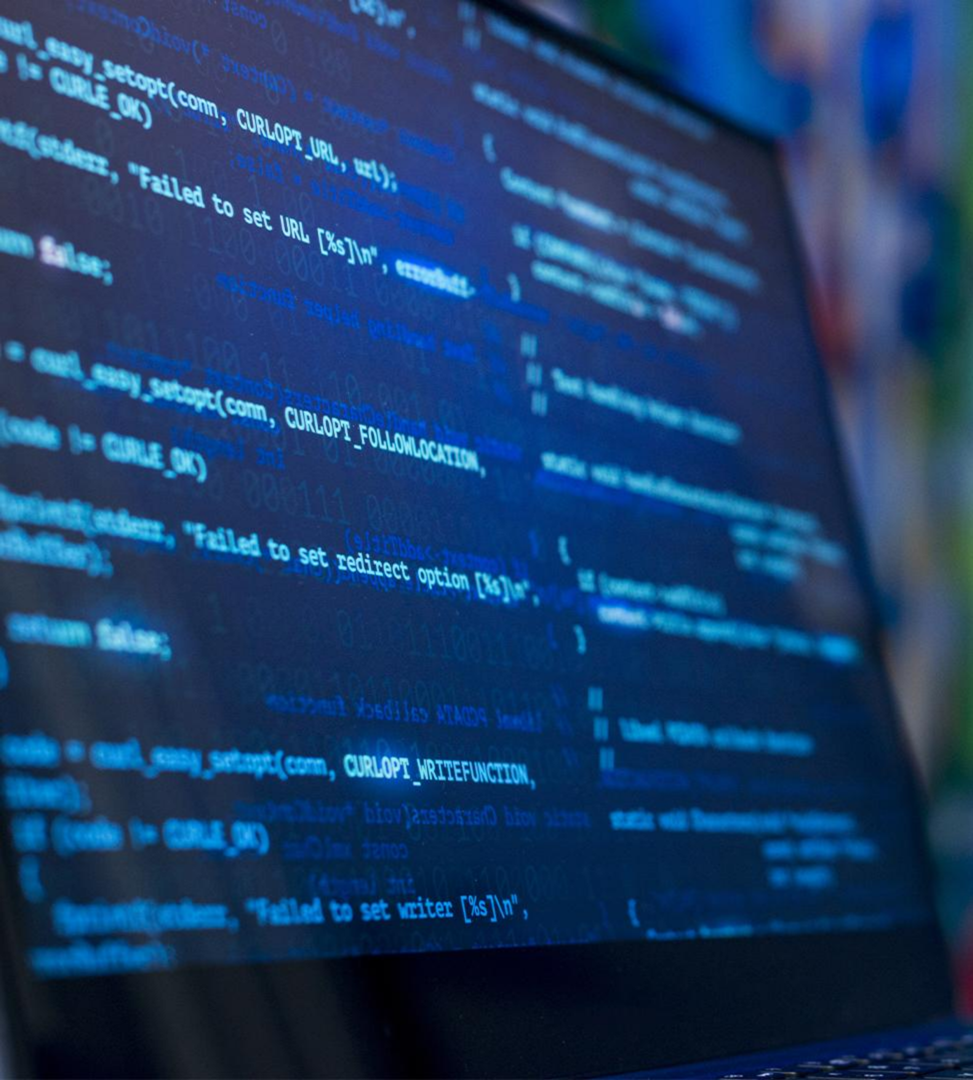
Creating Un-targeted Attack

Creating Un-targeted Attack

We focus on spoofing certificates for *untargeted* attacks, in which the attacker does not specify the class into which the attack image moves. To achieve this, we generate an adversarial perturbation for all possible wrong classes $\bar{y}$ and choose the best one as our strong attack:

$$\max_{\bar{y} \neq y, \delta} -L(\theta, x + \delta \| \bar{y}) - \lambda_c C(\delta) - \lambda_{tv} TV(\delta) - \lambda_s Dissim(\delta) \tag{4}$$

where y is the true label/class for the clean image x, and L is a spoofing loss that promotes a strong certificate. We examine different choices for L for different certificates below.

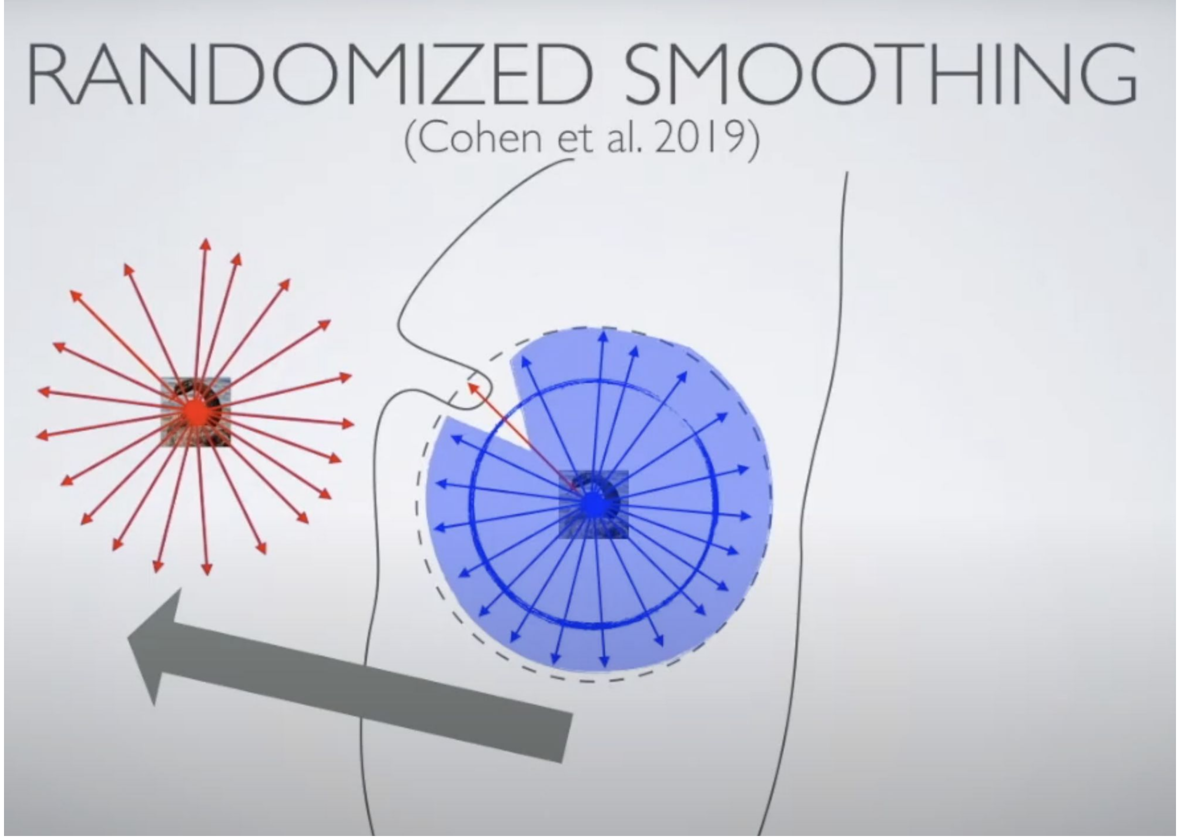


Attacks on Randomized Smoothing by Shadow Attack

Attacks on Randomized Smoothing by Shadow Attack

- It is an adversarial defense against l_2-norm bounded attacks
- If the variation is large, the smoothed classifier abstains from making a prediction.
- To spoof strong certificates (large certified radius) for an incorrect class, we must make sure that the majority of a batch of noisy images around the adversarial image are assigned the same (wrong) label. We do this by minimizing the cross entropy loss relative to a chosen (incorrect) label, averaged over a large set of randomly perturbed images. To this end, we minimize equation 4, where L is chosen to be the average cross-entropy over a batch of Gaussian perturbed copies.

Attacks on Randomized Smoothing by Shadow Attack



Attacks on Randomized Smoothing by Shadow Attack

Certified radii produced by the Randomized Smoothing method for Shadow Attack images and also natural images (larger radii means a stronger/more confident certificate).

Dataset	$\sigma(l_2)$	Unmodified/Natural Images Mean	Unmodified/Natural Images STD	Shadow Attack Mean	Shadow Attack STD
CIFAR-10	0.12	0.14	0.056	**0.22**	0.005
	0.25	0.30	0.111	**0.35**	0.062
	0.50	0.47	0.234	**0.65**	0.14
	1.00	0.78	0.556	**0.85**	0.442
ImageNet	0.25	0.30	0.109	**0.31**	0.109
	0.50	0.61	0.217	0.38	0.191
	1.00	1.04	0.519	0.64	0.322

Attacks on Crown-IBP by Shadow Attack

Attacks on Crown-IBP by Shadow Attack – Brief Introduction of IBP

Interval Bound Propagation (IBP) methods have been recently studied as a defense against l-infinity bounded attacks.

Attacks on Crown-IBP by Shadow Attack – Certificate

During testing, the user chooses an l–infinity perturbation bound, and error propagation is used to bound the magnitude of the largest achievable perturbation in network output. If the output perturbation is not large enough to **flip** the image label, then a certificate is produced. If the output perturbation is large enough to **flip** the image label, then a certificate is not produced.

Attacks on Crown-IBP by Shadow Attack – Attack

Although possessing such a certificate, it can still be vulnerable to a shadow attack in a whitebox scenario. An attacker simply needs to consider the certificate while training the adversarial example.

Attacks on Crown-IBP by Shadow Attack – Result

Table 2: "Robust error" for natural images, and "attack error" for Shadow Attack images using the CIFAR-10 dataset, and CROWN-IBP models. Smaller is better.

$\epsilon(l_\infty)$	Model Family	Method	Robustness Errors		
			Min	Mean	Max
2/255	9 small models	CROWN-IPB	52.46	57.55	60.67
		Shadow Attack	**45.90**	**53.89**	65.74
	8 large models	CROWN-IBP	52.52	53.9	56.05
		Shadow Attack	**46.21**	**49.77**	**51.79**
8/255	9 small models	CROWN-IBP	71.28	72.15	73.66
		Shadow Attack	**63.43**	**66.94**	**71.02**
	8 large models	CROWN-IBP	70.79	71.17	72.29
		Shadow Attack	**64.04**	**67.32**	**71.16**

Code

https://github.com/Trusted-AI/adversarial-robustness-toolbox/blob/main/art/attacks/evasion/shadow_attack.py

Code – Perturbation Initialization

```
perturbation = (
    np.random.uniform(
        low=self.estimator.clip_values[0], high=self.estimator.clip_values[1], size=x.shape
    ).astype(ART_NUMPY_DTYPE)
    - (self.estimator.clip_values[1] - self.estimator.clip_values[0]) / 2
)
```

Code – Training, Updating Perturbation Overview

```
for _ in trange(self.nb_steps, desc="Shadow attack", disable=not self.verbose):
    gradients_ce = np.mean(
        self.estimator.loss_gradient(x=x_batch + perturbation, y=y_batch, sampling=False)
        * (1 - 2 * int(self.targeted)),
        axis=0,
        keepdims=True,
    )
    gradients = gradients_ce - self._get_regularisation_loss_gradients(perturbation)
    perturbation += self.learning_rate * gradients
```

Code – _get_regularisation_loss_gradients : 3 channel loss

```python
if perturbation_t.shape[1] == 1:
    loss_s = 0.0
elif perturbation_t.shape[1] == 3:
    loss_s = tf.norm(
        (perturbation_t[:, 0, :, :] - perturbation_t[:, 1, :, :]) ** 2
        + (perturbation_t[:, 1, :, :] - perturbation_t[:, 2, :, :]) ** 2
        + (perturbation_t[:, 0, :, :] - perturbation_t[:, 2, :, :]) ** 2,
        ord=2,
        axis=(1, 2),
    )
else:
    raise ValueError("Value for number of channels in `perturbation_t.shape` not recognized.")
```

Code – _get_regularisation_loss_gradients : Loss

```
loss = torch.mean(self.lambda_tv * loss_tv + self.lambda_s * loss_s + self.lambda_c * loss_c)
```

Code – Back to training loop : total loss

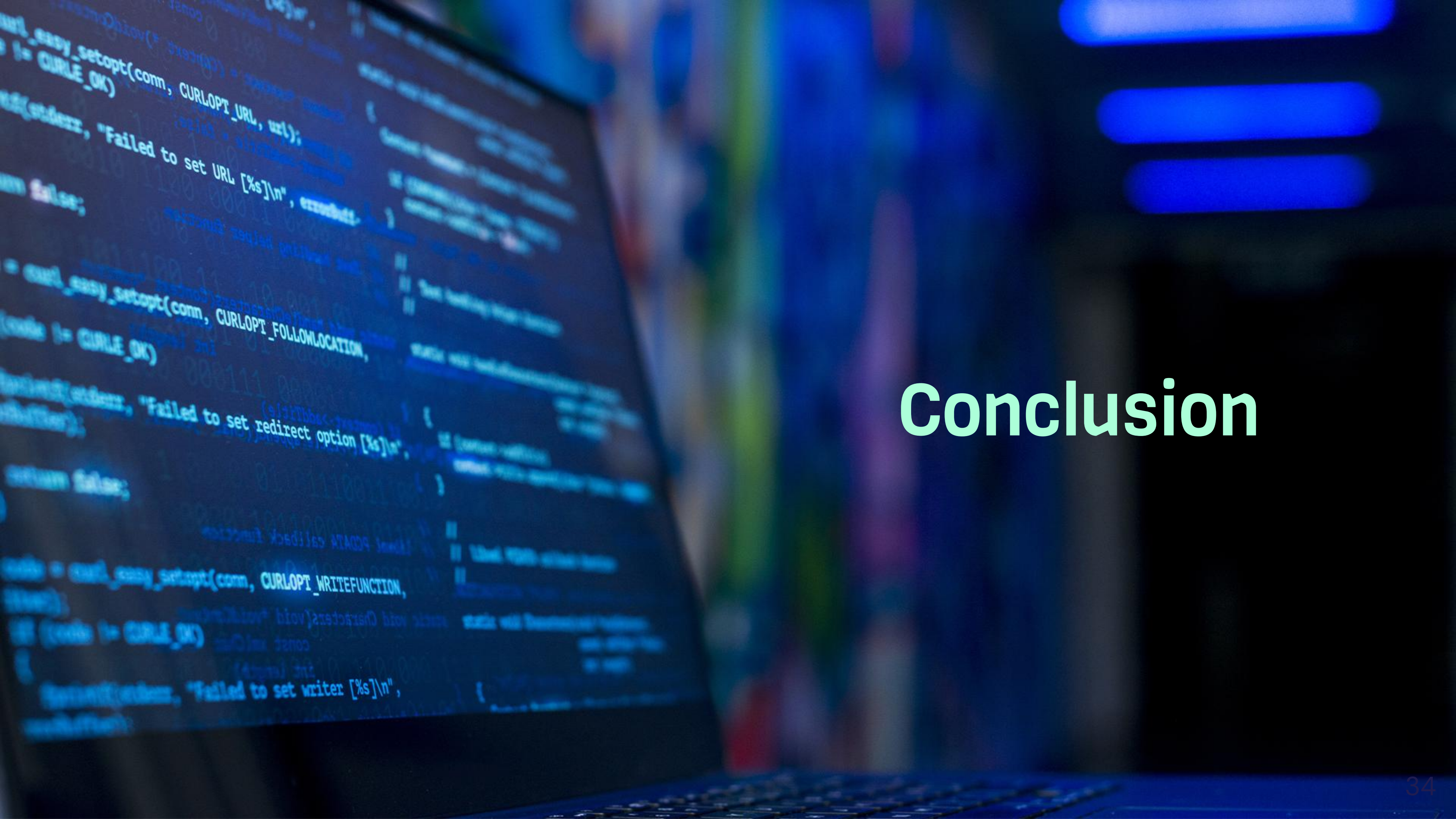
```
for _ in trange(self.nb_steps, desc="Shadow attack", disable=not self.verbose):
    gradients_ce = np.mean(
        self.estimator.loss_gradient(x=x_batch + perturbation, y=y_batch, sampling=False)
        * (1 - 2 * int(self.targeted)),
        axis=0,
        keepdims=True,
    )
    gradients = gradients_ce - self._get_regularisation_loss_gradients(perturbation)
    perturbation += self.learning_rate * gradients
```

Code – Return Adversarial Example : x_adv

```python
x_p = x + perturbation
x_adv = np.clip(x_p, a_min=self.estimator.clip_values[0], a_max=self.estimator.clip_values[1]).astype(
    ART_NUMPY_DTYPE
)

return x_adv
```

Conclusion



curl_easy_setopt(conn, CURLOPT_URL, url);
"Failed to set URL [%s]\n",
curl_easy_setopt(conn, CURLOPT_FOLLOWLOCATION,
"Failed to set redirect option [%s]\n",
CURLOPT_WRITEFUNCTION,
"Failed to set writer [%s]\n",

Conclusion

It is demonstrated that it is possible to produce adversarial examples with "spoofed" certified robustness by using large-norm perturbations. The adversarial examples are built using the Shadow Attack, which produces smooth and natural-looking perturbations that are often less perceptible than those of the commonly used lp-bounded perturbations, while being large enough in norm to escape the certification regions of state-of-the-art principled defenses. This work suggests that the certificates produced by certifiably robust classifiers, while mathematically rigorous, are not always good indicators of robustness or accuracy.

curl_easy_setopt(conn, CURLOPT_URL, url);
"Failed to set URL [%s]\n",
curl_easy_setopt(conn, CURLOPT_FOLLOWLOCATION,
CURLE_OK)
"Failed to set redirect option [%s]\n",
return false;
CURLOPT_WRITEFUNCTION,
"Failed to set writer [%s]\n",

Reference

Reference

- https://arxiv.org/pdf/2003.08937.pdf
- https://arxiv.org/pdf/1902.02918.pdf
- https://www.youtube.com/watch?v=hvemlq8pjno

Thanks for your listening!