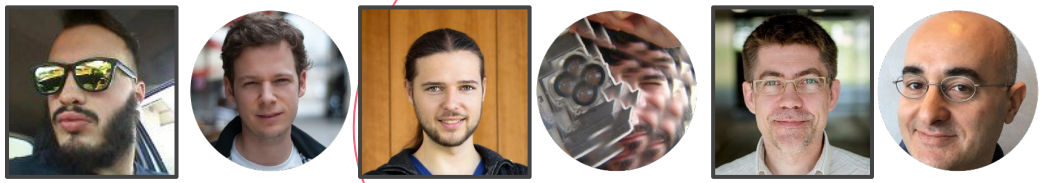
Evaluating the Adversarial Robustness of Adaptive Test-Time Defenses

Francesco Croce*, Sven Gowal*,
Thomas Brunner*, Evan Shelhamer*
Matthias Hein, Taylan Cemgil



Everyday Robots

Adversarial Robustness: Non-Adaptive Defenses

Deep nets are vulnerable to *adversarial attacks*: small input (x) perturbations ($\boldsymbol{\delta}$) that result in errors

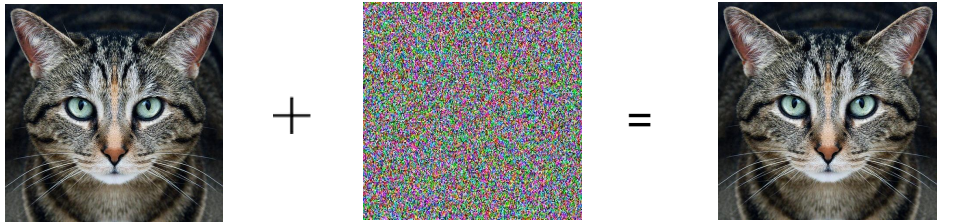


illustration credit: Athalye and Carlini

$$\boldsymbol{x} \qquad \boldsymbol{\delta} \text{ in } \{\boldsymbol{\delta} \mid \|\boldsymbol{\delta}\|_p \leq \epsilon\}$$

- **Train-Time Defenses:** devise models, losses, ..., and optimization algorithms for training classifier, e.g. *adversarial training* (Madry et al., 2018). They are *static* and do not change during testing

- **Test-Time Defenses:** alter inference during testing by altering the input x or model, e.g. test-time data augmentation. While these alter inference, they do not *adapt*, as their alterations do not depend on the input

Adaptive Test-Time Defenses

Adaptive test-time defenses

- alter inference and condition their changes on the input by optimization

- iteratively update (during inference) the input x or parameters θ of the network to improve robustness to adversarial attack

Potential benefits

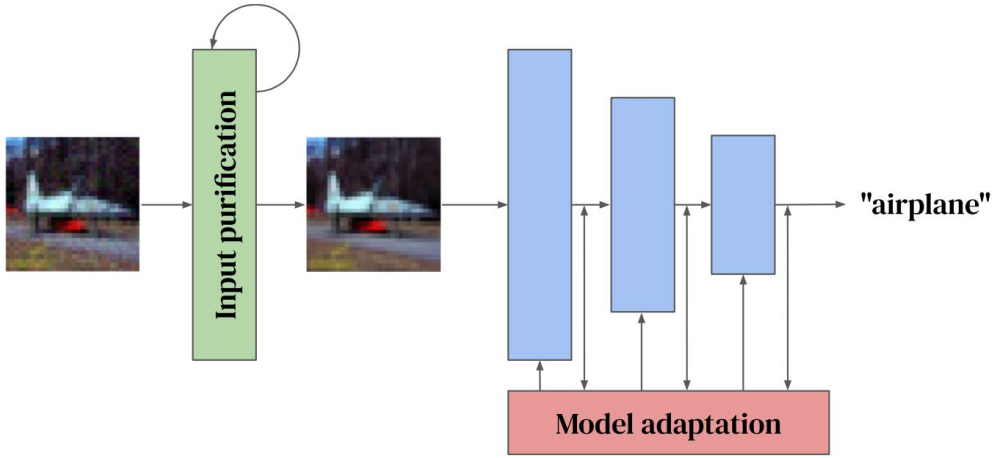
- preserving clean accuracy, by adapting differently to natural inputs and attacked inputs, unlike train-time defenses
- improving robust accuracy, by changing the defense as a function of the attack, unlike non-adaptive test-time defenses

Note: Test-time adaptation has already shown improvement for *natural* shifts, so why not adversarial? (Sun et al. ICLR'20, Schneider et al. NeurIPS'20, Wang et al. ICLR'21, ...)

Paradigms of Adaptive Defenses

input adaptation or "purification" optimizes the input then passes it to the model.

model adaptation or "test-time training" optimizes the parameters or latents of the model.



Both paradigms update by iterative optimization during testing

Elements of Adaptive Defenses

iteration (I) iteratively updating test-time optimization problems by e.g. gradient descent

auxiliary networks (AN) equipping the model with another network to define the loss or update for optimization

randomization (R) randomizing explicitly, by adding noise, or implicitly, by sampling data

external data (ED) exploiting additional data for prediction, so that it does not depend solely on test inputs

Elements of Adaptive Defenses: Iteration

iteration (I) iteratively updating test-time optimization problems by e.g. gradient descent

auxiliary networks (AN) equipping the model with another network to define the loss or update for optimization

randomization (R) randomizing explicitly, by adding noise, or implicitly, by sampling data

external data (ED) exploiting additional data for prediction, so that it does not depend solely on test inputs

$$x_{t+1} \leftarrow x_t - \alpha_t \nabla_{x_t} \ell(x_t)$$

gradient iteration

$$x_{t+1} \leftarrow x_t + \alpha_t g_\phi(x_t)$$

model iteration

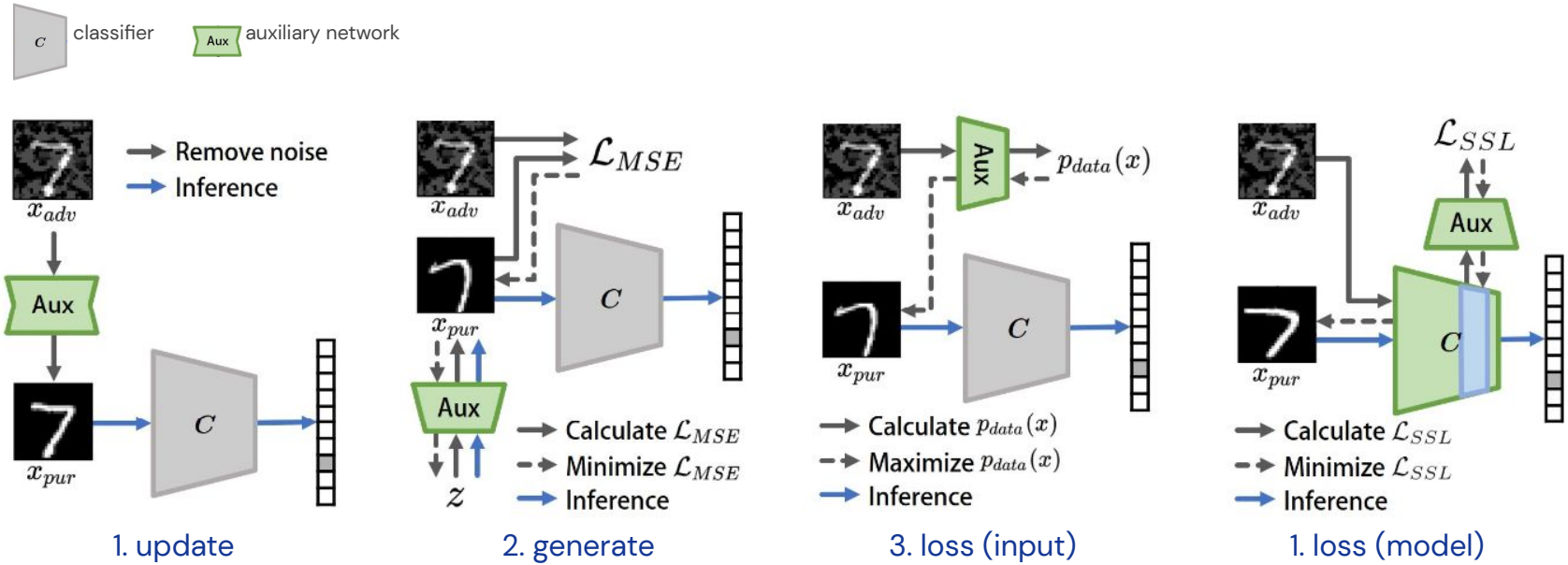
Elements of Adaptive Defenses: Auxiliary Networks

iteration (I) iteratively updating test-time optimization problems by e.g. gradient descent

auxiliary networks (AN) equipping the model with another network to define the loss or update for optimization

randomization (R) randomizing explicitly, by adding noise, or implicitly, by sampling data

external data (ED) exploiting additional data for prediction, so that it does not depend solely on test inputs



1. update | 2. generate | 3. loss (input) | 1. loss (model)

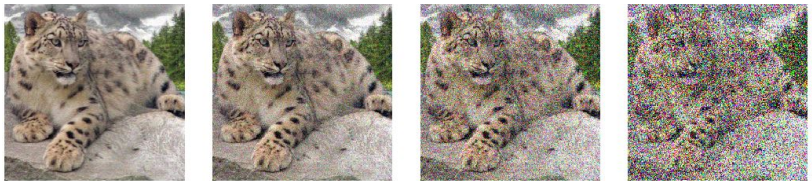
Elements of Adaptive Defenses: Randomization

iteration (I) iteratively updating test-time optimization problems by e.g. gradient descent

auxiliary networks (AN) equipping the model with another network to define the loss or update for optimization

randomization (R) randomizing explicitly, by adding noise, or implicitly, by sampling data

external data (ED) exploiting additional data for prediction, so that it does not depend solely on test inputs



explicit: adding noise

implicit: sampling for batching

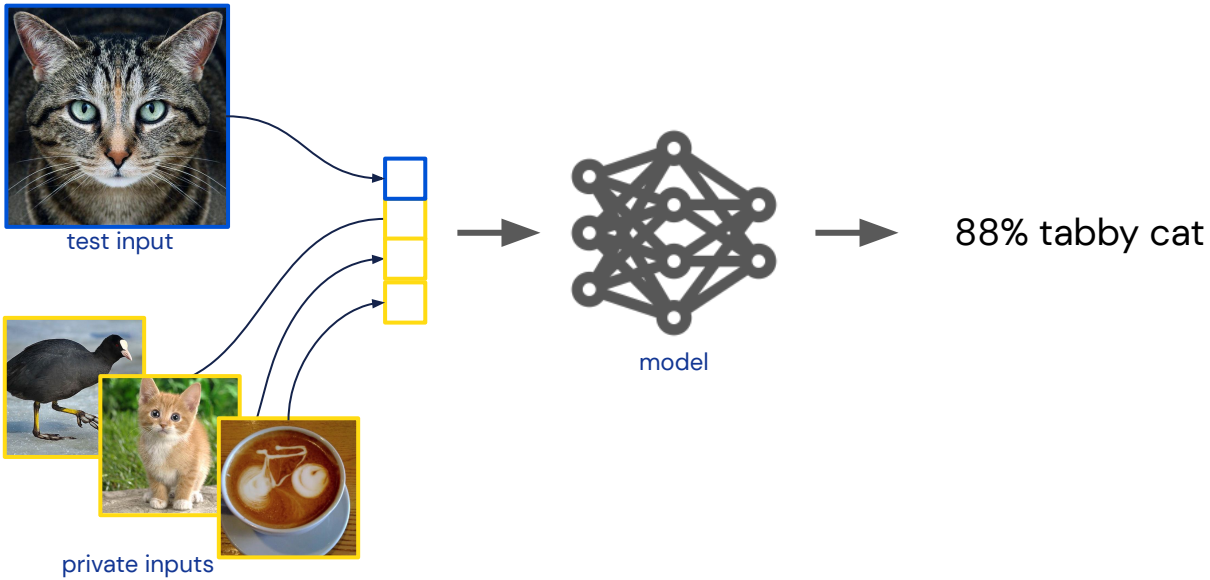
Elements of Adaptive Defenses: External Data

iteration (I) iteratively updating test-time optimization problems by e.g. gradient descent

auxiliary networks (AN) equipping the model with another network to define the loss or update for optimization

randomization (R) randomizing explicitly, by adding noise, or implicitly, by sampling data

external data (ED) exploiting additional data for prediction, so that it does not depend solely on test inputs



Case Study: Summary of Results

Our evaluation of the studied adaptive defenses on CIFAR-10, Linf-threat model, ε=8/255

I = Input Adaptation, **M** = Model Adaptation **IA** = Iterative Algorithm, **AN** = Auxiliary Network, **R** = Randomization, **ED** = Extra Data

Defense	Venue	Paradigm		Elements				Attack	Robust Accuracy Theirs / Ours (Base)	Inference Time
		I	M	IA	AN	R	ED			
Kang et al. '21	NeurIPS		●	●	●			Transfer APGD	57.8 / 52.2 (53.9)	2x
Chen et al. '21*	ICLR		●	●	●			APGD+BPDA	34.5*/ 5.6 (0.0)	59x
Wu et al. '21	arXiv	●		●		●		Transfer APGD+BPDA+EoT	65.7 / 61.0 (63.0)	46x
Alfarra et al. '22	AAAI	●		●				RayS	79.2 / 66.6 (66.6)	8x
Shi et al. '21	ICLR	●		●	●			APGD+BPDA*	51.0 / 3.7 (0.0)	518x
Qian et al. '21	arXiv	●		●	●			APGD	65.1 / 12.6 (7.7)	4x
Hwang et al. '21	ICMLW	●		●	●			APGD+BPDA	52.7 / 43.8 (49.3)	40x
Mao et al. '21**	ICCV	●		●	●	●	●	APGD+EoT	63.8 / 58.4 (59.4)	407x
Yoon et al. '21	ICML	●		●	●	●		APGD+EoT	69.7 / 33.7 (0.0)	176x

* Chen et al.: this paper and our evaluation use ε=2/255. ** Mao et al.: our evaluation uses batch size 50, and not the original 512, for computational reasons.

Case Study: results and observations

- Complex defenses are often more difficult to evaluate (Tramèr, arXiv'20)
- There is not a single evaluation method which works across all defenses, but we combine several existing techniques, depending on the characteristics of the defense
- Adaptive test-time defenses that make use of a robust static model do not improve its robustness, and often degrade it
- Adaptive test-time defenses using a nominal, non-robust static model might yield some robustness, but less than the degree obtained by e.g. adversarial training
- Test-time adaptation increases inference time by multiplicative factors, or even orders of magnitude(!)

Best Practices for Attacking Adaptive Defenses

- Transfer attacks from the static defense to the adaptive defense
- Evaluate by gradient-based attacks on the full defense when possible (e.g. APGD +iterations +multiple losses +multiple restarts) – current frameworks like JAX and PyTorch enable easy gradient computation
- Evaluate the defense with score-based and decision-based black-box attacks
- Approximate non-differentiable components by BPDA (either with identity or more complex functions) and attack randomization by EoT
- Remember that attacks developed for static classifiers, e.g. AutoAttack, might be not effective in presence of test-time adaptation (but can provide a baseline)