Pluribus One
seeing one in many

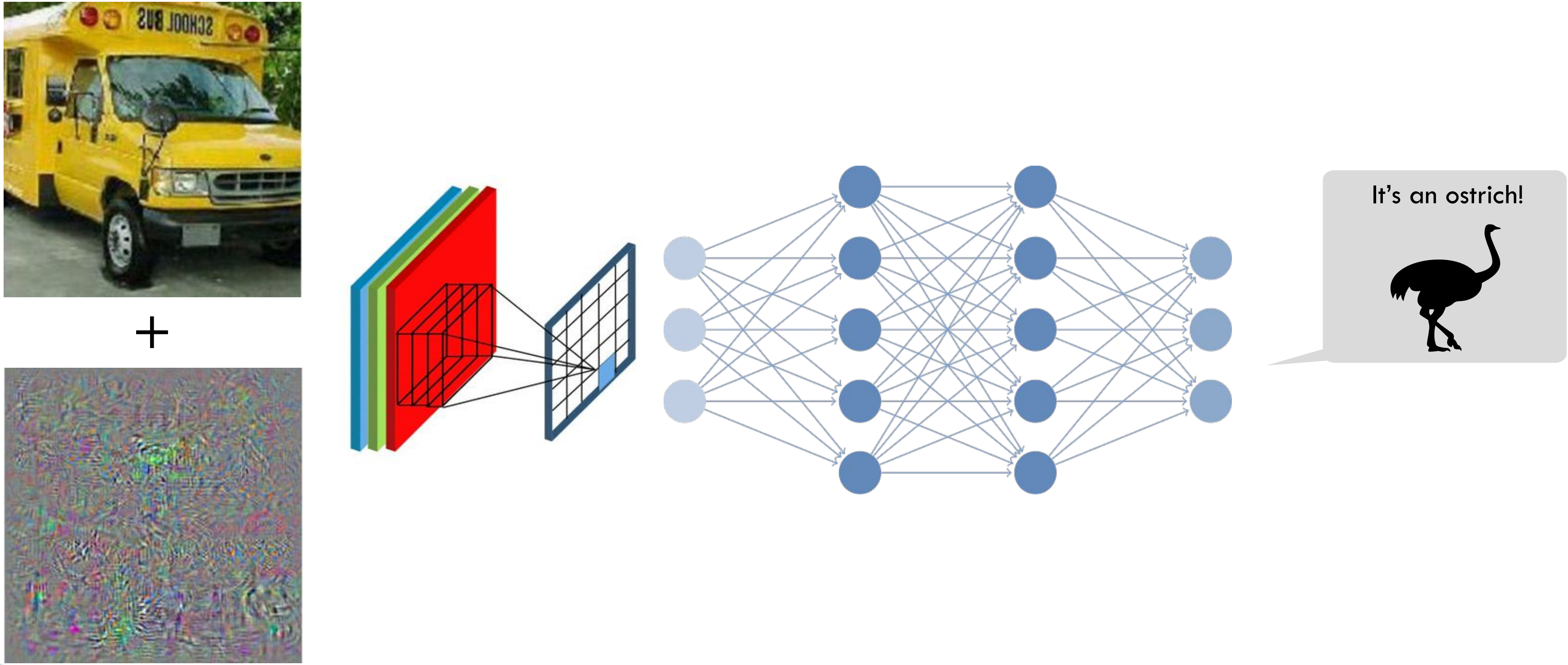
Pattern Recognition and Applications Lab

University of Cagliari, Italy

Indicators of Attack Failure: Visualizing and Debugging Optimization of Adversarial Examples

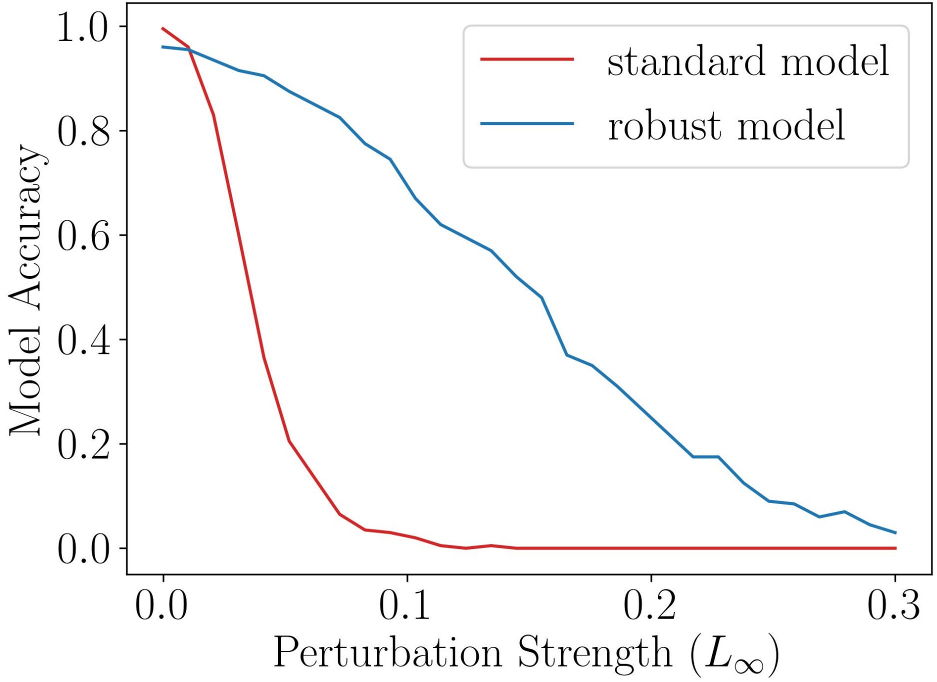
Maura Pintor, Luca Demetrio, Angelo Sotgiu, Giovanni Manca, Ambra Demontis, Nicholas Carlini, Battista Biggio, Fabio Roli

Adversarial Examples



Biggio et al., *Evasion Attacks Against Machine Learning at Test Time*, ECML PKDD 2013
Szegedy et al., *Intriguing Properties of Neural Networks*, ICLR 2014

Adversarial Robustness

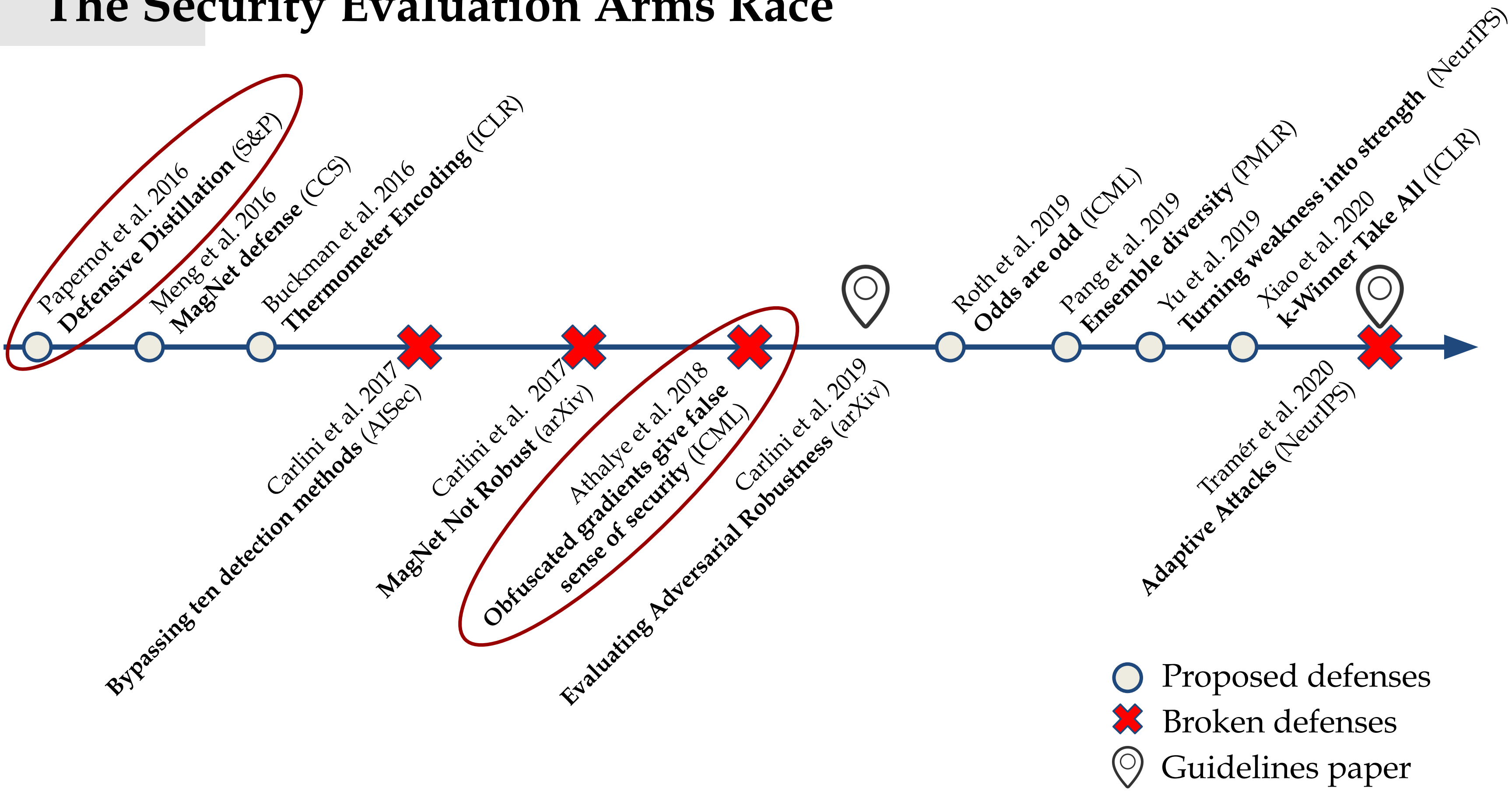


Evaluating adversarial robustness amounts to finding adversarial examples with a given perturbation budget

We have to rely on empirical evaluation

What is the problem of using empirical evaluations?

The Security Evaluation Arms Race



Robustness evaluations and pitfalls

The security evaluation is a way to estimate the robust accuracy of a model

Literature provides approaches to improve robustness evaluations

Limits: manual process, qualitative metrics, only suggestions and "best practices"

Currently, there is no debugging tool for adversarial attacks

Our contributions:

- Different point of view for debugging a security evaluation: per-point, inspect path
- Provide computable indicators
- Propose systematic protocol for improving security evaluations

Where can attacks fail and how can we detect the failures?

Pitfalls of Gradient-based Attacks

Algorithm 1: General formalization for *untargeted* (Equation 1 and 2) and *targeted* attacks (Equation 3 and 4)

Input : $\boldsymbol{x}$, the initial point; y_t, the target (true) class label if the attack is targeted (untargeted); n, the number of iterations; α, the learning rate; f, the target model; $(\boldsymbol{x}_{lb}, \boldsymbol{x}_{ub})$, the bounds of the input space; Δ, the considered region.

Output : $\boldsymbol{x}^\star$, the solution found by the algorithm

$\boxed{\boldsymbol{x}_0 \leftarrow init(\boldsymbol{x})}$ ▷ `Initialize starting point`

$\boxed{\hat{\boldsymbol{\theta}} \leftarrow approximation(\boldsymbol{\theta})}$ ▷ `Approximate model's parameters`

$\boldsymbol{\delta}_0 \leftarrow \mathbf{0}$ ▷ `Initial` δ

$\boxed{\textbf{for } i \in [1, n] \textbf{ do}}$

$\quad \boxed{\boldsymbol{\delta}' \leftarrow \boldsymbol{\delta}_i + \alpha \nabla_{\boldsymbol{x}_i} L(\boldsymbol{x}_0 + \boldsymbol{\delta}_i, y_t; \hat{\boldsymbol{\theta}})}$ ▷ `Compute optimizer step`

$\quad \boldsymbol{\delta}_{i+1} \leftarrow apply\_constraints(\boldsymbol{x}_0, \boldsymbol{\delta}', \Delta, \boldsymbol{x}_{lb}, \boldsymbol{x}_{ub})$ ▷ `Apply constraints`

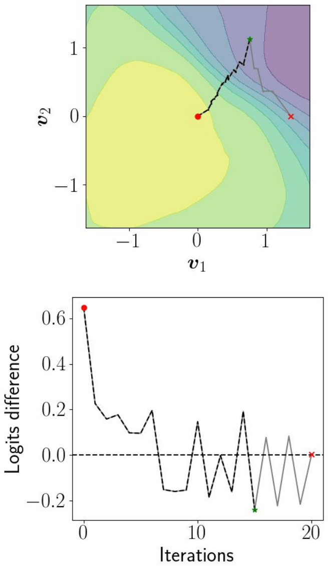
$\boxed{\boldsymbol{\delta}^\star \leftarrow best(\boldsymbol{\delta}_0, ..., \boldsymbol{\delta}_n)}$ ▷ `Choose best perturbation`

return $\boldsymbol{\delta}^\star$

Attack Failures

Bad implementation

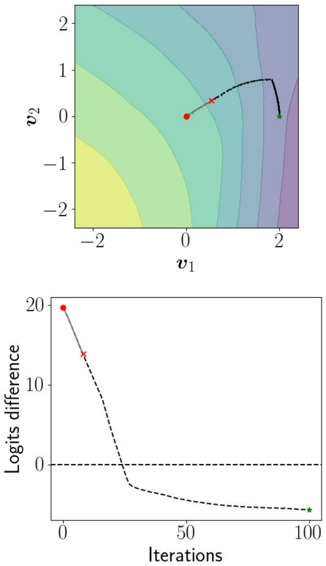
$$\boldsymbol{\delta}^\star \leftarrow best(\boldsymbol{\delta}_0, ..., \boldsymbol{\delta}_n)$$



Attack is not converging

for $i \in [1, n]$ **do**

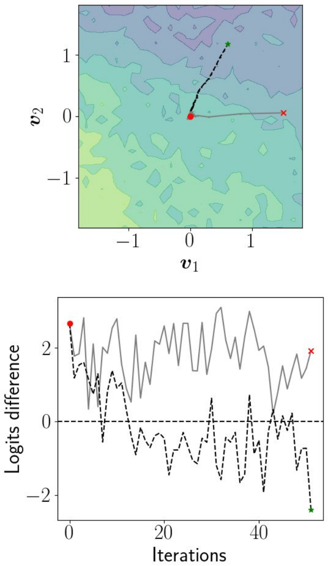
$$\boldsymbol{\delta}' \leftarrow \boldsymbol{\delta}_i + \alpha \nabla_{\boldsymbol{x}_i} L(\boldsymbol{x}_0 + \boldsymbol{\delta}_i, y_t; \hat{\boldsymbol{\theta}})$$



Bad local optimum

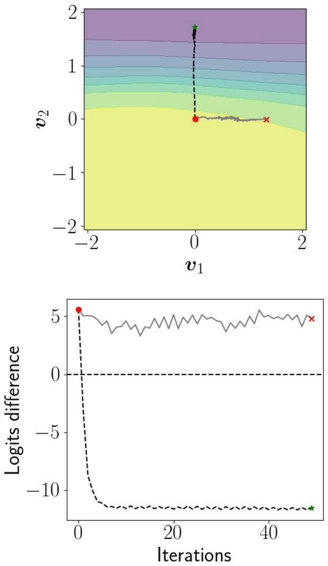
$$\boldsymbol{x}_0 \leftarrow init(\boldsymbol{x})$$

$$\hat{\boldsymbol{\theta}} \leftarrow approximation(\boldsymbol{\theta})$$



Attack is not adaptive

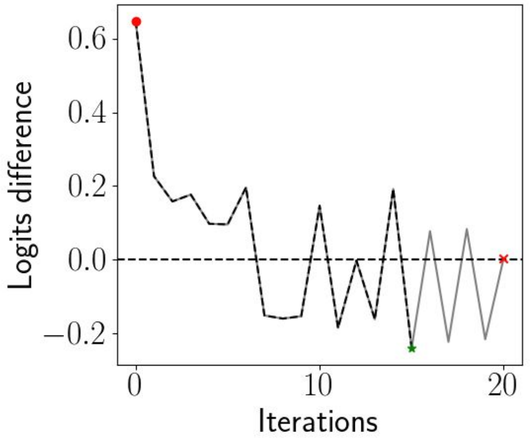
$$\hat{\boldsymbol{\theta}} \leftarrow approximation(\boldsymbol{\theta})$$



Silent success

Measures if the attack is returning a wrong result

Computed by looking inside the optimization path



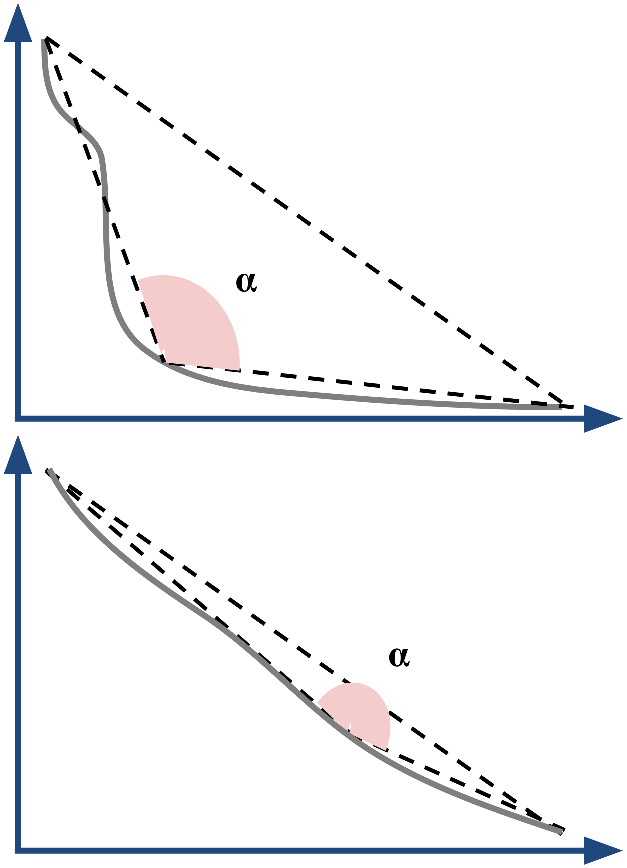
Break-point angle

Measures the alignment between the improvement of the loss over the iterations and the expected decreasing behaviour

Computed as the absolute value of cos **α**

90° → loss has the correct shape

180° → loss is still decreasing

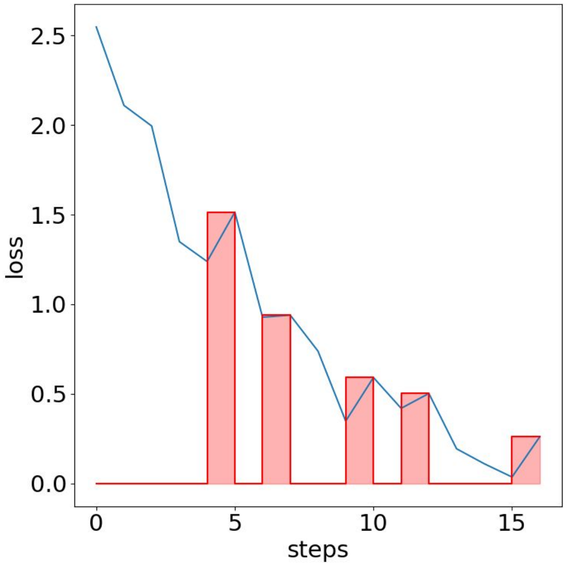


Increasing loss

Measures the increment of the loss due to jumps inside the space

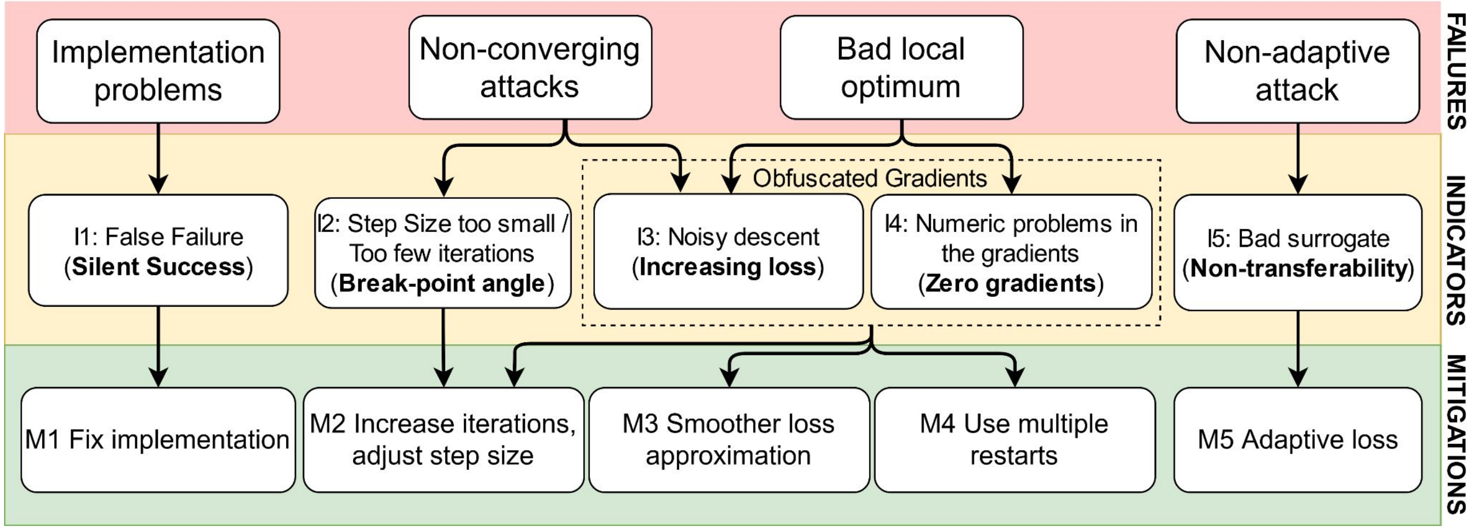
Computed as area under positive contributions in the attack loss

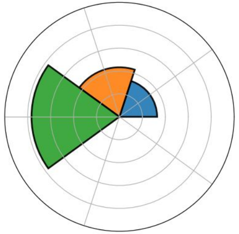
If indicator is close to 1, it means the loss is not decreasing consistently



And how do we fix them?

Indicators and mitigations

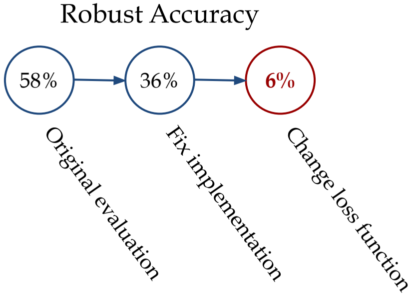


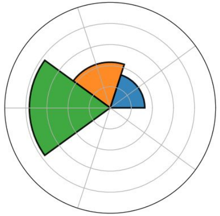


These charts help us understand when indicators are triggered, and which mitigations to apply

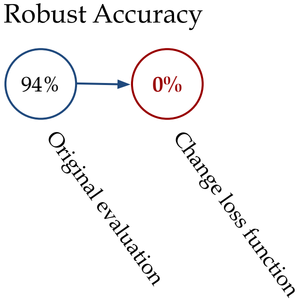
Experiments

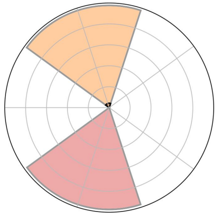
k-Winners
Take All



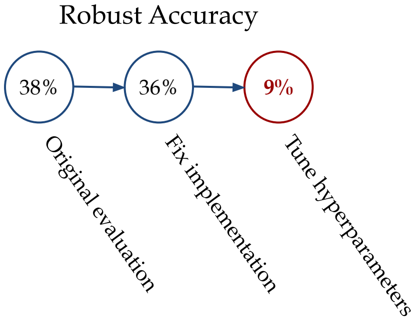


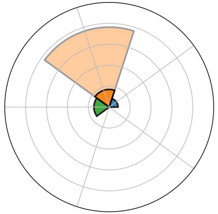
Distillation



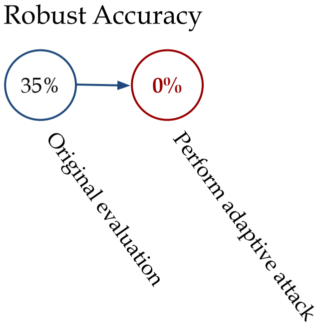


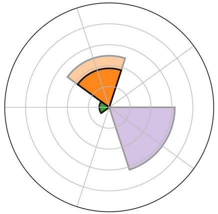
Ensemble Diversity





Turning a
Weakness into a
Strength





I_1: Silent Success I_2: Break-Point Angle I_3: Increasing Loss
I_4: Zero Gradients I_5: Non-transferability

Conclusions

- Enable debugging faulty-conducted security evaluations
- Empirical evaluation in 4 case-studies
- Paper available https://arxiv.org/abs/2106.09947
- Open source code https://github.com/pralab/IndicatorsOfAttackFailure

Future works

- Integrate in benchmarks
- Further automatization - ML*Sec*Ops

https://secml.gitlab.io/