

Train, but Verify: Towards Practical AI Robustness

Problem

The Beiler Taxonomy (2019) categorizes three ways a machine learning system can be attacked. The three matching security policies for a defender to enforce are:

1. **Learn** the right thing, even from adversary influenced data.
2. **Do** the right thing, even with adversarial examples present.
3. Never **Reveal** sensitive information about the model/data.

Existing defense research primarily focuses on only one of these security policies at a time. This is an important limitation, because recent research demonstrates that state of the art methods for enforcing do policies can lead to violations of reveal policies.

Train\Verify	Verify learn	Verify do	Verify reveal
Train for learn			
Train for do			
Train for reveal			Train, but Verify

Solution

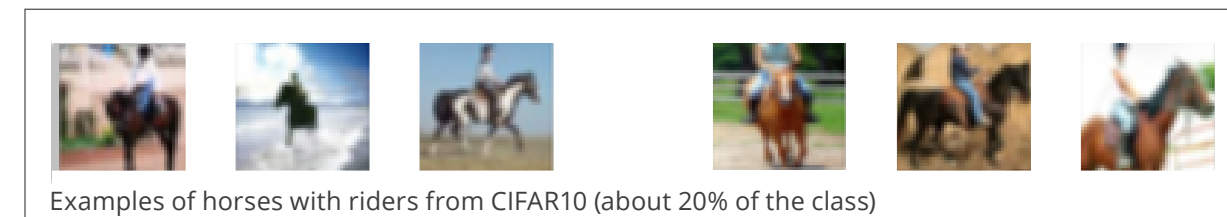
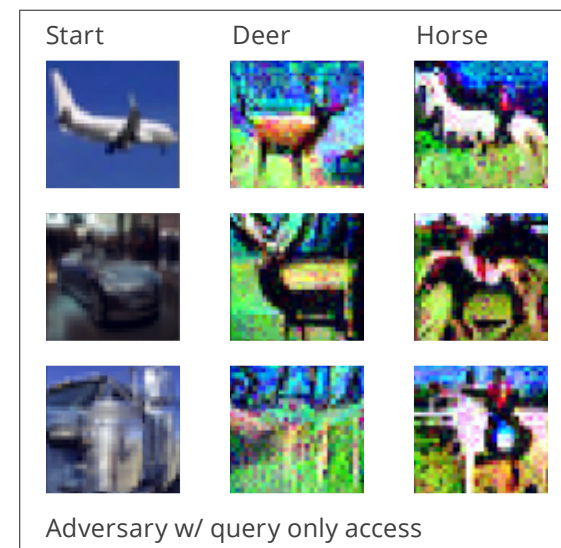
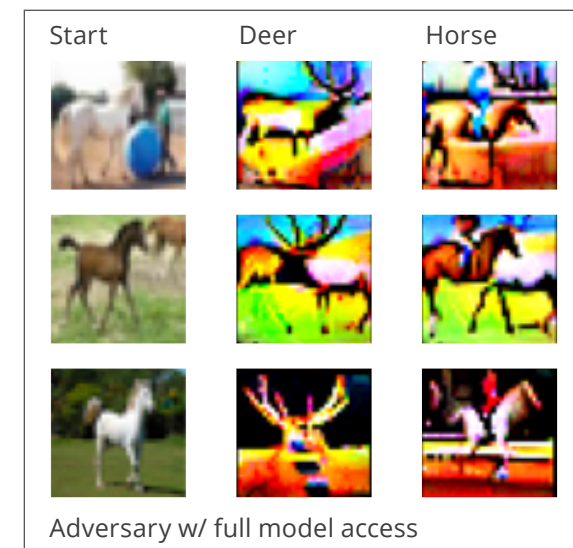
1. **Train** secure AI systems by training ML models to enforce at least two security policies.
2. **Verify** the security of AI systems by testing against realistic threat models across multiple policies.

Intended Impact (FY20-22)

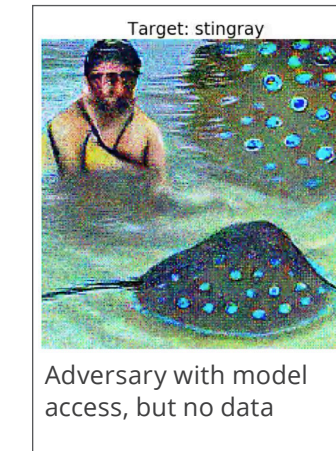
- Provide proof-of-concept defenses that either enforce multiple policies, or trade off between those policy goals.
- Provide proof-of-concept tooling to verify security policies across multiple policies.

An AI system **trained** for high-stakes decisions may **reveal critical information** about its training data.

For models trained on CIFAR 10 to enforce a do policy (TRADES, Zhang et al., 2019), adversaries with both full-model access and query-only access can recover the presence of riders on horses (about 20% of the class).



The ImageNet stingray class contains swimmers



... Cauliflower class contains purple cauliflower



CIFAR 10 data set documented in Krizhevsky, Alex. "Learning Multiple Layers of Features from Tiny Images." April 8, 2009.
ImageNet photos courtesy of ImageNet.

Copyright 2020 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

* These restrictions do not apply to U.S. government entities.

DM20-0904