SoftwareX 18 (2022) 101095

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

secml: Secure and explainable machine learning in Python

Maura Pintor [a,b], Luca Demetrio [a,b], Angelo Sotgiu [a,b], Marco Melis [a], Ambra Demontis [a], Battista Biggio [a,b,*]

[a] DIEE, University of Cagliari, Via Marengo, Cagliari, Italy
[b] Pluribus One, Via Vincenzo Bellini, 9, Cagliari, Italy

ARTICLE INFO

Article history:
Received 5 January 2022
Received in revised form 21 April 2022
Accepted 25 April 2022

Keywords:
Machine learning
Security
Adversarial attacks
Explainability
Python3

ABSTRACT

We present secml, an open-source Python library for secure and explainable machine learning. It implements the most popular attacks against machine learning, including test-time evasion attacks to generate adversarial examples against deep neural networks and training-time poisoning attacks against support vector machines and many other algorithms. These attacks enable evaluating the security of learning algorithms and the corresponding defenses under both white-box and black-box threat models. To this end, secml provides built-in functions to compute security evaluation curves, showing how quickly classification performance decreases against increasing adversarial perturbations of the input data. secml also includes explainability methods to help understand why adversarial attacks succeed against a given model, by visualizing the most influential features and training prototypes contributing to each decision. It is distributed under the Apache License 2.0 and hosted at https://github.com/pralab/secml.

Code metadata

Current code version	v0.15
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00012
Permanent link to Reproducible Capsule	https://github.com/pralab/secml/releases/tag/v0.15
Legal Code License	Apache License Version 2.0
Code versioning system used	Git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	*numpy*, *sci-kit learn*, *matplotlib*, *scipy*, *joblib*, *Pillow*, *requests*, *python-dateutil* (*torch*, *torchvision*, *foolbox*, *cleverhans*, *tensorflow v1* as optional requirements)
If available Link to developer documentation/manual	https://secml.readthedocs.io/
Support email for questions	maura.pintor@unica.it, luca.demetrio93@unica.it

1. Introduction

Machine learning is vulnerable to well-crafted attacks. At test time, the attacker can stage evasion attacks (i.e., adversarial examples) [1–5], sponge examples [6], model stealing [7], and membership inference [8] attacks to violate system integrity, availability, or even its confidentiality. Similarly, at training time, the attacker may target either system integrity or availability via poisoning [9,10] and backdoor attacks [11]. The most studied attacks, namely evasion and poisoning, firstly explored by Biggio et al. [2,9], are formalized as constrained optimization problems, solved through gradient-based or gradient-free algorithms [10,12], depending on whether the attacker has white- or black-box access to the target system. Many libraries implement the former, however, they do not allow developers to assess machine learning models' security easily. Hence, we present secml, an open-source Python library that serves as a complete tool for evaluating and assessing the performance and robustness of

* Corresponding author at: DIEE, University of Cagliari, Via Marengo, Cagliari, Italy.
E-mail address: battista.biggio@unica.it (Battista Biggio).

https://doi.org/10.1016/j.softx.2022.101095

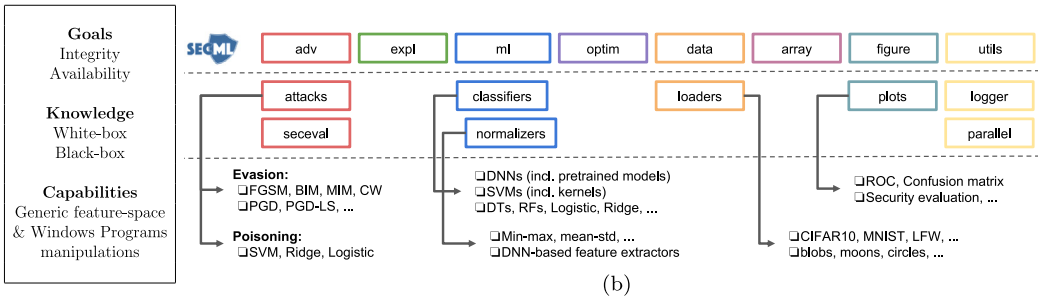

Fig. 1. Features available in `secml`. On the left, the supported threat models; on the right, the structure of the packages, and the supported strategies.

machine-learning models. To this end, `secml` implements: (*i*) a methodology for the empirical security evaluation of machine-learning algorithms under different evasion and poisoning attack scenarios; and (*ii*) explainable methods to help understand why and how these attacks are successful. With respect to other popular libraries that implement attacks almost solely against Deep Neural Networks (DNNs) [13–15], `secml` also implements training-time poisoning attacks and computationally-efficient test-time evasion attacks against many different algorithms, including support vector machines (SVMs) and random forests (RFs). It also incorporates both the feature-based and prototype-based explanation methods proposed by [16–18].

2. Software description

`secml` has a modular architecture oriented to code reuse. We have defined abstract interfaces for all components, including loss functions, regularizers, optimizers, classifiers, and attacks. By separating the definition of the optimization problem from the algorithm used to solve it, one can easily define novel attacks or classifiers (in terms of constrained optimization problems) and then use different optimizers to obtain a solution. This is a great advantage with respect to other libraries like *CleverHans* [13], *Foolbox* [14,19], and *ART* [15] as, we can switch from white-box to black-box attacks by just changing the optimizer (from a gradient-based to a gradient-free solver), without re-defining the entire optimization problem.

`secml` integrates different components via well-designed wrapper classes. We integrate several attack implementations from *CleverHans* and *Foolbox*, by extending them to also track the values of the loss function and the intermediate points optimized during the attack iterations, as well as the number of function and gradient evaluations. This is useful to debug and compare different attacks, e.g., by checking their convergence to a local optimum and properly tuning their hyperparameters (e.g., step size and number of iterations). `secml` supports DNNs via a dedicated *PyTorch* wrapper, which can be extended to include other popular deep-learning frameworks, like *TensorFlow* and *Keras*, and it natively supports *scikit-learn* classifiers as well. This allows us to run attacks that are implemented in *CleverHans* and *Foolbox* to universally run on both *PyTorch* and *scikit-learn* models.

Main packages. The library supports different threat models (Fig. 1(a)), and it is organized in different packages (Fig. 1(b)). The `adv` package implements different adversarial attacks and provides the functionalities to perform security evaluations. It includes target and untargeted evasion attacks provided by *CleverHans* and *Foolbox* as well as our implementations of evasion and poisoning attacks [10]. These attacks target the feature space of a model, regardless of its domain of application (e.g. attacking Android malware detection[1]). Hence, developers can encode ad-hoc manipulations that correspond to perturbations in the input space of a targeted domain (e.g. Windows programs [20]), and then use the algorithms included inside `secml` to optimize the attack. All the included attacks are listed in Fig. 1(b).

The `ml` package imports classifiers from *scikit-learn* and DNNs from *PyTorch*. We have extended *scikit-learn* classifiers with the gradients required to run evasion and poisoning attacks, which have been implemented analytically. Our library also supports chaining different modules (e.g., scalers and classifiers) and can automatically compute the corresponding end-to-end gradient via the chain rule. The `explanation` package implements the feature- and prototype-based explanation methods [16–18]. The `optim` package provides an implementation of the projected gradient descent (PGD) algorithm, and a more efficient version of it that runs a bisect line search along the gradient direction (PGD-LS) to reduce the number of gradient evaluations [21]. Finally, `data` provides data loaders for popular datasets, integrating those provided by *scikit-learn* and *PyTorch*; `array` provides a higher-level interface for both dense (*numpy*) and sparse (*scipy*) arrays, enabling the efficient execution of attacks on sparse data representations; `figure` implements some advanced plotting functions based on *matplotlib* (e.g., to visualize and debug attacks); and *utils* provides functionalities for logging and parallel code execution. We also provide a model zoo, available on GitHub as well,[2] that contains pre-trained models to rapidly test newly implemented attacks and utilities.

We recap the main functionalities of `secml` in Table 1, where we also compare it with other relevant libraries. Notably, our library is the sole that provides attack loss inspection plots to choose the appropriate attacks' hyperparameters, and security evaluation plots, to ease the complexity of assessing the robustness of machine learning models. Moreover, the features offered by `secml` are not only related to attacking machine learning models, but they are gathered to elevate `secml` as a complete tool for attacking, inspecting, and assessing the performances of machine learning models.

Testing and documentation. We have run extensive tests on macOS X, Ubuntu 16.04, Debian 9 and 10, through the GitHub Actions infrastructure,[3] with also experimental support on Windows 10 platforms. The user documentation is available at https://secml.readthedocs.io/en/v0.15/, along with a basic developer guide detailing how to extend the `ml` package with other classifiers and deep-learning frameworks. The complete set of unit tests is available in our repository. A comprehensive view of the functionalities available in `secml` is included in tutorials available as Jupyter notebooks.

3. Impact

We now offer two examples extracted from `secml` to showcase its impact: evasion attacks against DNNs, and a poisoning attack against an SVM.

[1] https://github.com/pralab/secml/blob/master/tutorials/13-Android-Malware-Detection.ipynb.

[2] https://github.com/pralab/secml-zoo.

[3] https://docs.github.com/en/actions/automating-builds-and-tests.

Table 1
Comparison of `secml` and the other adversarial machine learning libraries. We show whether the library offer full (✓), partial (∼) or no (x) support of a particular feature.

	DL frameworks support	scikit-learn support	Built-in attack algorithms	Wraps adversarial frameworks	Dense/Sparse data support	Security evaluation plots	Attack loss inspection plots	Loss separated from Optimizer	Explainability	Model zoo	Comprehensive tutorials
`secml`	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Foolbox	✓	x	✓	x	x	∼	x	x	x	✓	∼
ART	✓	✓	✓	x	x	x	x	x	x	x	✓
CleverHans	✓	x	✓	x	x	x	x	x	x	x	x

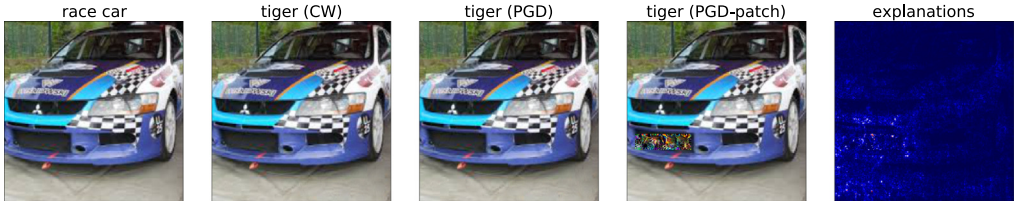

Fig. 2. Adversarial images (CW, PGD, and PGD-patch) representing a *race car* misclassified as a *tiger*. For PGD-patch, we also report explanations via integrated gradients.

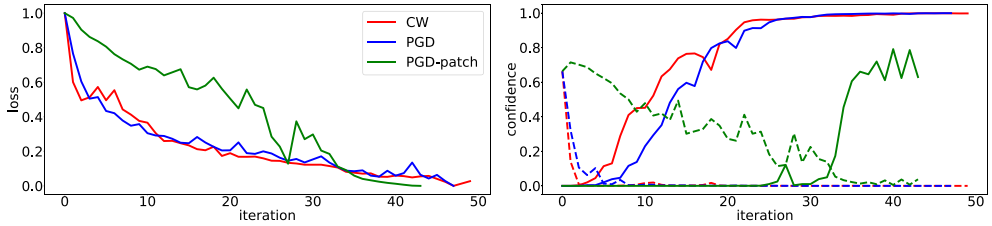

Fig. 3. Attack optimization. *Left*: loss minimization; *Right*: confidence of source class (*race car*, dashed lines) vs confidence of target class (*tiger*, solid lines), across iterations.

Evasion attacks. We show here how to use `secml` to run different evasion attacks against ResNet-18, a DNN pretrained on ImageNet, available from *torchvision*. This example demonstrates how `secml` enables running *CleverHans* attacks (implemented in *TensorFlow*) against *PyTorch* models. Our goal is perturbing the image of a race car to be misclassified as a tiger, using the ℓ_2-norm targeted Carlini–Wagner (CW) attack (from *CleverHans*), the ℓ_2 PGD attack implemented in `secml`, and PGD-patch, where the attacker can only change the pixels corresponding to the license plate [22].

We run all the attacks for 50 iterations, we set the confidence parameter of the CW attack $\kappa = 10^6$ to generate high-confidence misclassifications, and $c = 0.4$, yielding an ℓ_2 perturbation size $\epsilon = 1.87$. We bound PGD to create an adversarial image with the same perturbation size. For PGD-patch, we do not bound the perturbation size for the pixels that can be modified.

The resulting adversarial images are shown in Fig. 2. For PGD-patch, we also highlight the most relevant pixels used by the DNN to classify this image as a tiger, using the *integrated gradients* explanation method. The most relevant pixels are found around the perturbed region containing the license plate, unveiling the presence of potential adversarial manipulations.

We also visualize the performances of the attack in Fig. 3. The leftmost plot shows how the attack losses (scaled linearly in [0, 1] to enable comparison) iteration-wise, while the rightmost plot shows how the confidence assigned to class *race car* (dashed line) decreases in favor of the confidence assigned to class *tiger* (solid line) for each attack, across different iterations. By inspecting them, we can understand if these attacks have been correctly configured. For instance, by looking at the loss curves on the left, we can understand if the attacks reached convergence or not, thus facilitating tuning of either the step size or the number of iterations. Also, by looking at the plot on the right, it is clear that all the attacks are successful since the confidence of the target class exceeds the score of the original one.

Poisoning attacks. We also show the effect of a poisoning attack provided by `secml` applied to an SVM classifier implemented in *scikit-learn*. The experimental setting and code are available in one of our tutorials on GitHub.[4]

Results of the successful attack are represented in Fig. 4., highlighting the flexibility of `secml` in applying different strategies to third-party models as well, without the need of customizing them on a particular framework.

4. Conclusions and future work

The `secml` project was born more than five years ago and we open-sourced it in August 2019. Thanks to an emerging community of users and developers from our GitHub repository, we firmly believe that `secml` will soon become a reference

[4] https://github.com/pralab/secml/blob/master/tutorials/05-Poisoning.ipynb.

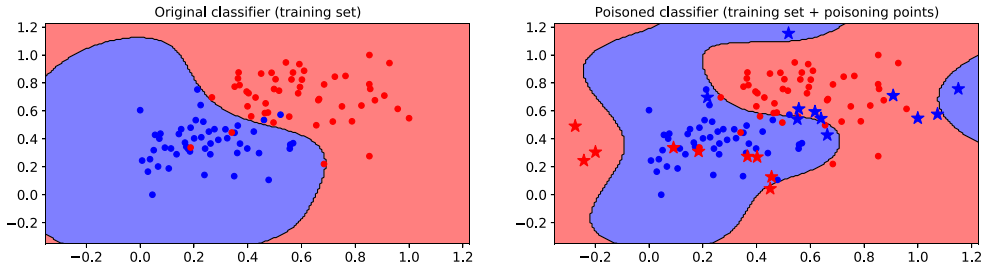

Fig. 4. Poisoning attacks against an SVM implemented with *scikit-learn*. The poisoning data (denoted with ⋆ in the right plot) induce the model to learn a worse decision boundary.

tool to evaluate the security of machine-learning algorithms. We are constantly working to enrich it with new functionalities, by adding novel defenses, wrappers for other third-party libraries, and more pretrained models to the `secml` zoo.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partly supported by the PRIN 2017 project RexLearn, funded by the Italian Ministry of Education, University and Research (grant no. 2017TWNMH2); by the project ALOHA, under the EU's H2020 programme (grant no. 780788); and by the project TESTABLE (grant no. 101019206), under the EU's H2020 research and innovation programme.

References

[1] Huang L, Joseph AD, Nelson B, Rubinstein B, Tygar JD. Adversarial machine learning. In: 4th ACM workshop on artificial intelligence and security. 2011, p. 43–57.
[2] Biggio B, Corona I, Maiorca D, Nelson B, Šrndić N, Laskov P, et al. Evasion attacks against machine learning at test time. In: Blockeel Hendrik, Kersting Kristian, Nijssen Siegfried, Železný Filip, editors. Machine learning and knowledge discovery in databases. LNCS, vol. 8190, Springer Berlin Heidelberg; 2013, p. 387–402, Part III.
[3] Szegedy Christian, Zaremba Wojciech, Sutskever Ilya, Bruna Joan, Erhan Dumitru, Goodfellow Ian, et al. Intriguing properties of neural networks. In: International conference on learning representations. 2014.
[4] Papernot Nicolas, McDaniel Patrick, Jha Somesh, Fredrikson Matt, Berkay Celik Z, Swami Ananthram. The limitations of deep learning in adversarial settings. In: Proc. 1st IEEE European symposium on security and privacy. IEEE; 2016, p. 372–87.
[5] Carlini Nicholas, Wagner David A. Towards evaluating the robustness of neural networks. In: IEEE symposium on security and privacy. IEEE Computer Society; 2017, p. 39–57.
[6] Shumailov Ilia, Zhao Yiren, Bates Daniel, Papernot Nicolas, Mullins Robert, Anderson Ross. Sponge examples: Energy-latency attacks on neural networks. In: 2021 IEEE European symposium on security and privacy. IEEE; 2021, p. 212–31.
[7] Tramèr Florian, Zhang Fan, Juels Ari, Reiter Michael K, Ristenpart Thomas. Stealing machine learning models via prediction APIs. In: 25th USENIX security symposium. 2016, p. 601–18.
[8] Shokri Reza, Stronati Marco, Song Congzheng, Shmatikov Vitaly. Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security and privacy. IEEE; 2017, p. 3–18.
[9] Biggio Battista, Nelson Blaine, Laskov Pavel. Poisoning attacks against support vector machines. In: Langford John, Pineau Joelle, editors. 29th Int'L conf. on machine learning. Omni Press; 2012, p. 1807–14.
[10] Biggio B, Roli F. Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognit 2018;84:317–31.
[11] Gu Tianyu, Liu Kang, Dolan-Gavitt Brendan, Garg Siddharth. Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access 2019;7:47230–44.
[12] Joseph Anthony D, Nelson Blaine, Rubinstein Benjamin IP, Tygar JD. Adversarial machine learning. Cambridge University Press; 2018.
[13] Papernot Nicolas, Faghri Fartash, Carlini Nicholas, Goodfellow Ian, Feinman Reuben, Kurakin Alexey, et al. Technical report on the cleverhans V2.1.0 Adversarial examples Library. 2018, arXiv preprint arXiv:1610.00768.
[14] Rauber Jonas, Brendel Wieland, Bethge Matthias. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In: Reliable machine learning in the wild workshop, 34th international conference on machine learning. 2017.
[15] Nicolae Maria-Irina, Sinn Mathieu, Tran Minh Ngoc, Buesser Beat, Rawat Ambrish, Wistuba Martin, et al. Adversarial robustness toolbox v1.0.0. 2018, arXiv preprint arXiv:1807.01069.
[16] Ribeiro Marco Tulio, Singh Sameer, Guestrin Carlos. Why should I trust you?: Explaining the predictions of any classifier. In: 22nd ACM SIGKDD Int'L Conf. Knowl. Disc. data mining. New York, NY, USA: ACM; 2016 p. 1135–44.
[17] Sundararajan Mukund, Taly Ankur, Yan Qiqi. Axiomatic attribution for deep networks. In: International conference on machine learning. PMLR; 2017, p. 3319–28.
[18] Koh Pang Wei, Liang Percy. Understanding black-box predictions via influence functions. In: International conference on machine learning. PMLR; 2017, p. 1885–94.
[19] Rauber Jonas, Zimmermann Roland, Bethge Matthias, Brendel Wieland. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. J Open Source Softw 2020;5(53):2607.
[20] Demetrio Luca, Biggio Battista. Secml-malware: Pentesting windows malware classifiers with adversarial exemples in python. 2021, arXiv preprint arXiv:2104.12848.
[21] Demontis Ambra, Melis Marco, Pintor Maura, Jagielski Matthew, Biggio Battista, Oprea Alina, et al. Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. In: 28th USENIX security symposium USENIX security 19. USENIX Association; 2019.
[22] Melis Marco, Demontis Ambra, Biggio Battista, Brown Gavin, Fumera Giorgio, Roli Fabio. Is deep learning safe for robot vision? Adversarial examples against the icub humanoid. In: ICCVW vision in practice on autonomous robots. IEEE; 2017, p. 751–9.