REPORT: The impact of adversarial machine learning in security and privacy analytics: A survey

Student: Khoa Anh Do

ID: 223188874

Table Of Contents

1. Introduction

The introduction of Machine Learning has leveraged multiple disciplines that utilize data analytics, including the field of cyber-security. However, incorporating Machine Learning into real-life applications has its own set of drawbacks as it comes with multiple security vulnerabilities, from the dataset to the models themselves. This report aims to summarize and give an overview of currently known adversarial attacks, countermeasures, and their consequences in security and privacy analysis.

The structure of this report includes 6 chapters including this introduction. Chapter 2 will describe the current state of adversarial attacks: an overview of machine learning, the taxonomy of adversarial attacks, some current state-of-the-art models, and defense against adversarial AI. Chapter 3 will mention five different open-source or commercial tools for adversarial artificial intelligence attack and defense. Chapter 4 will summarize a real-life case study on the adversarial AI in the spam detection system of Facebook. Chapter 5 provides the discussion as well as the future direction of research for adversarial machine learning. Finally, we draw the conclusion from our findings and discussion.

2. Adversarial attacks

In this chapter, we will give an overview of Machine Learning, and the taxonomy of adversarial attacks. Then, we will discuss some of the current state-of-the-art attacks based on published literature and defense against adversarial attacks.

2.1. Overview of Machine Learning

Machine Learning is a part of Artificial Intelligence, a field that aims to bring human-level intelligence to machines in order to solve problems. Today, machine learning can broadly be classified into three categories: supervised learning, reinforcement learning, and unsupervised learning.

Supervised learning is the most popular form of machine learning that uses labeled data as input to ML models such as support vector machine, linear regression, logistic regression, etc. The goal of these models is either classification (assigning the input to a class) or regression (predicting a real value based on input features). Supervised learning targets many real-life problems, such as building a traffic sign recognition system (classification), predicting house pricing (regression), etc. Therefore, it has become the majority target for adversarial learning.

Reinforcement learning attempts to train the agent (model) without giving information; instead, the agent will learn to perform the correct action through a reward system: in a user-defined environment, agents take actions and receive feedback in the form of rewards or penalties based on their actions. Over time, they learn optimal strategies through trial and error, aiming to maximize the cumulative reward by adapting their decision-making policies. This process involves exploring different actions, evaluating their consequences, and refining their strategies through a balance of exploration and exploitation. For example, reinforcement learning can be used for smart autonomous vehicles: Each agent (vehicle) will learn to navigate through complex traffic situations by rewarding them for making safe and efficient driving decisions.

Unsupervised learning utilizes algorithms that analyze and extract patterns from unlabeled data, making it distinct from supervised learning and reinforcement learning. In unsupervised learning, the goal is to uncover hidden structures or relationships within the data without predefined target outcomes. Common techniques in this category include clustering, where data points are grouped into clusters based on similarities, and dimensionality reduction, which reduces the complexity of data by retaining its essential features. Unsupervised learning is an up-and-coming field, due to the amount of unlabeled data that is collected every day. One example of unsupervised learning is topic modeling, which can be used in the analysis of a large collection of news articles to automatically identify and extract distinct topics such as politics, sports, and entertainment, useful for tasks such as trend analysis.

2.2. Overview taxonomy of adversarial attacks

We first define the **attack surface** of a Machine Learning model. A Machine Learning lifecycle development process describes how raw information is transformed into actual prediction

outcomes.

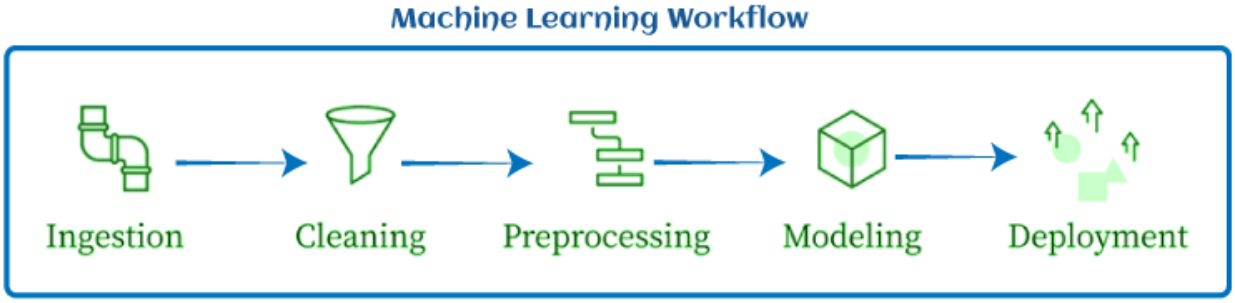


Figure 1: Machine learning system workflow
(source: https://www.javatpoint.com/machine-learning-pipeline)

Initially, raw data (e.g. images, recordings) is collected to create a dataset, which we can perform feature engineering steps such as data cleaning and feature extraction. Then, the processed data is fed into a Machine Learning algorithm, where it tries to learn the most helpful information to produce a predictive output. This learning process will be done on a training set, and the model performance will be evaluated on a separate testing set that has never been seen before. In recent advancements, these models do not stop learning after the initial training; instead, retraining is done when new data from actual users comes in to continue improving the model's accuracy. This process leaves numerous opportunities for attackers to tamper with the model's performance; in broad terms, there are two attack types: attack during the training phase and attack during the testing phase, with some few sub-categories **attack types** mentioned below:

- Poison attacks: occur only in the training phase, these attempts to compromise a model's performance by injecting carefully curated or manipulated data instances into its training dataset, potentially leading to misclassifications and security vulnerabilities.
- Evasion attacks: occur only in the testing phase, these attempt to exploit the model's misclassification, often with no influence over the training data.
- Exploratory attacks (testing phase): here, attackers attempt to extract as much information as possible through an exhaustive model testing process, to hopefully learn about the training set distribution and the models' underlying infrastructure.

To assess the scale of the aforementioned attacks, **adversarial goals** are used to measure what level and direction of damage is applied to the affected model, from reducing the confidence probability of correctly classified outputs to even misclassification. Here, there are two **specificities of attacks**: non-discriminant and discriminant. The former targets the general outcome of the model, regardless of labeling, such as an attempt to make the model predictions wrong for a wide range of inputs. The latter, also called targeted attacks, aims to focus on a specific group or individual label; for example, the attacker may try to make the model misclassify only a stop sign in a self-driving system, which can lead to consequences in real life.

Adversarial capabilities indicate the capabilities that an attacker has over the information and resources of the machine learning process, which directly influences their attacking abilities. These capabilities can be categorized further in the training phase and testing phase:

- Training phase: there are four levels of training phase manipulations. From the weakest to the strongest, we have: read access to the training set, data injection (the ability to add new corrupted data to the training set), data modification (the ability to directly read and write dataset instances), and logic corruption (read and write access to the model).
- Testing phase: The attackers' knowledge in the testing phases is measured by their knowledge of the machine learning system. In general view, we have whitebox, graybox, and blackbox attack scenarios. The first kind, whitebox, describes the scenario where the attackers have full knowledge of the model (architecture, parameters, pipeline), and training data distribution. In the third scenario, blackbox, the attack does not have access to any knowledge mentioned before. Finally, graybox refers to the scenario where some sort of knowledge is presented; however, not all knowledge is given.

Next, **attack strategy** defines how the attack is carried out, which is composed of non-colluding (attacker works alone) and colluding (attackers work toward the same system). The latter scenario may offer some benefits to attackers as it delegates the tasks among the group and may also help with covering their tracks.

Finally, **adversarial evaluation** is needed to perform measurement on the impact of before and after the adversarial attacks. With the aim to impact the classification outcome, by observing false positive and/or false negative rates, we can come up with a good estimation of the damage dealt to our model.

2.3. Current state-of-the-art adversarial attack techniques

In this section, we will discuss various attack techniques explored in cybersecurity literature and focus on the three mentioned attack types: poisoning attacks, evasion attacks, and exploratory attacks.

2.3.1. Poisoning attacks

As mentioned above, poisoning attacks describe adversarial attacks that aim to alter the training data depending on the adversarial capability (read, injection, modification, and logic corruption). Some attack techniques include backdoor attacks and label flipping. **Label flipping** attempts to modify the classification labeling, thereby increasing the number of misclassifications. For example, Random Label Flipping naively altered an instance label which ultimately resulted in an increase of false positives and false negatives.

- Biggio et al. [1] have demonstrated the effectiveness of **Random label flipping** by showing the performance decrease of the Support Vector Machine model under the label flipping scenarios.

Backdoor attacks categorize a subset of attacks that includes a meticulously crafted backdoor key. Only in the presence of this chosen input key, the model which previously performed well on the clean training and validating will start to degrade.

- In 2017, Gu et al. [2] coined the term **BadNets** which refers to these backdoored neural networks, and demonstrated through a traffic sign detection system attack that will trigger backdoors in the appearance of a small Post-it note. It is worth noting that the average accuracy of this BadNets compared to its clean counterpart has little difference. However, when a backdoor trigger is presented like the image below:

Figure 2: Image misclassification due to a Post-it note backdoor trigger [2]

The stop sign shown has a 95% confidence prediction as a speed limit sign, which is detrimental in real life.

- In 2021, Souri et al. [3] published a new backdoor trigger attack called **Sleeper Agent** which is effective against untrained models. This proposed black-box approach combines gradient matching, poison selection, and adaptive retraining. Gradient matching is a technique that aligns gradients in optimization problems, commonly used in deep learning to manipulate model behavior in tasks like adversarial attacks. Poison selection, or data selection, is carried out by selecting a small number of training samples to tamper in order to maximize the influence of prediction. During the poisoning period, the model is constantly retrained and evaluated to see the poisoning effect. The below image describes how a sleeper agent is used: a subset of training samples is selected and

adds noises (perturbation) for training, which then leads the model to obtain an undetectable backdoor attack.

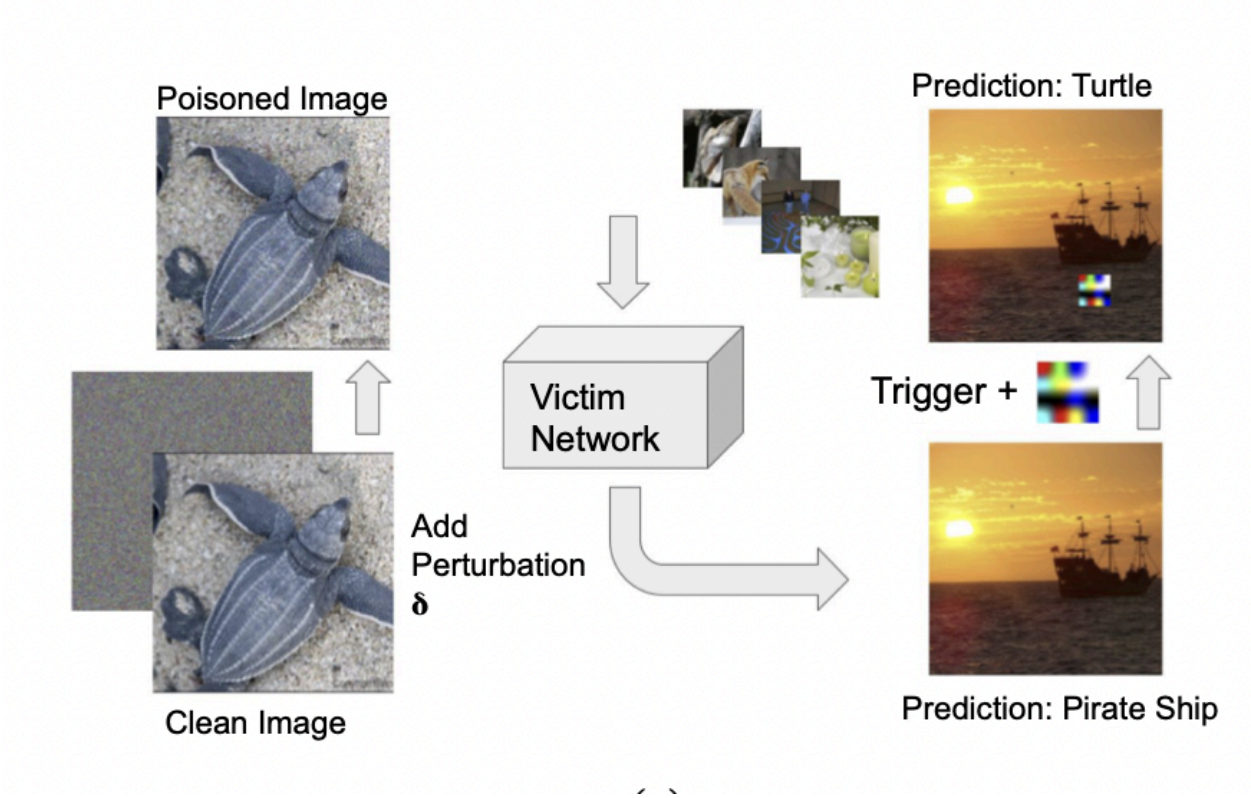


Figure 3: Visualization for Sleeper Agent model [3]

- In 2022, Chan et al. [4] proposed **BadDet**, a series of poisoning backdoor attacks focusing on the object detection problem. Specifically, an Object Generation attack follows a traditional pattern that falsely generates classification on the trigger. Regional Misclassification Attacks force the model to misclassify all instances of the image to be misclassified as a specific label within a region of the trigger. Global Misclassification Attacks will make the model misclassified to one specific label on a trigger. Finally, an

Object Disappearance Attack will make some classification go undetected.

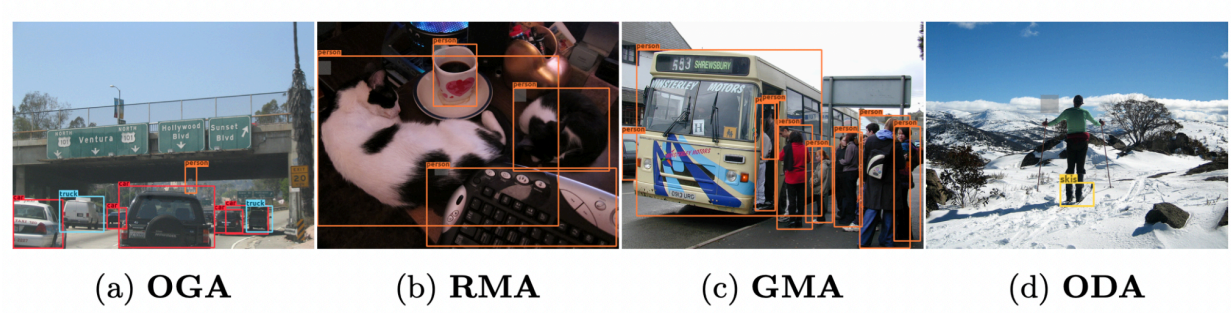


Figure 4: Example for BadDet Attacks [4]

2.3.1. Evasion attacks

Evasion attacks occur at the testing phase attempting to modify the testing sample by introducing false perception to the model, even though it may not be differentiable for the human eye. We will touch on evasion attacks and some state-of-the-art examples in two scenarios: whitebox (Carlini and Wagner) and blackbox (square attack).

- **Carlini and Wagner** whitebox attacks [5] is one of the strongest algorithms for evasion attack to date. This algorithm is accustomed to three different distance metrics scenarios: L_0, L_2, and L_∞. In the below example extracted from adversarial.js, a stop sign in the clean model has had a C&W filter applied, making the model classify it as a 120km/hr sign, which will have a detrimental consequence in real life.

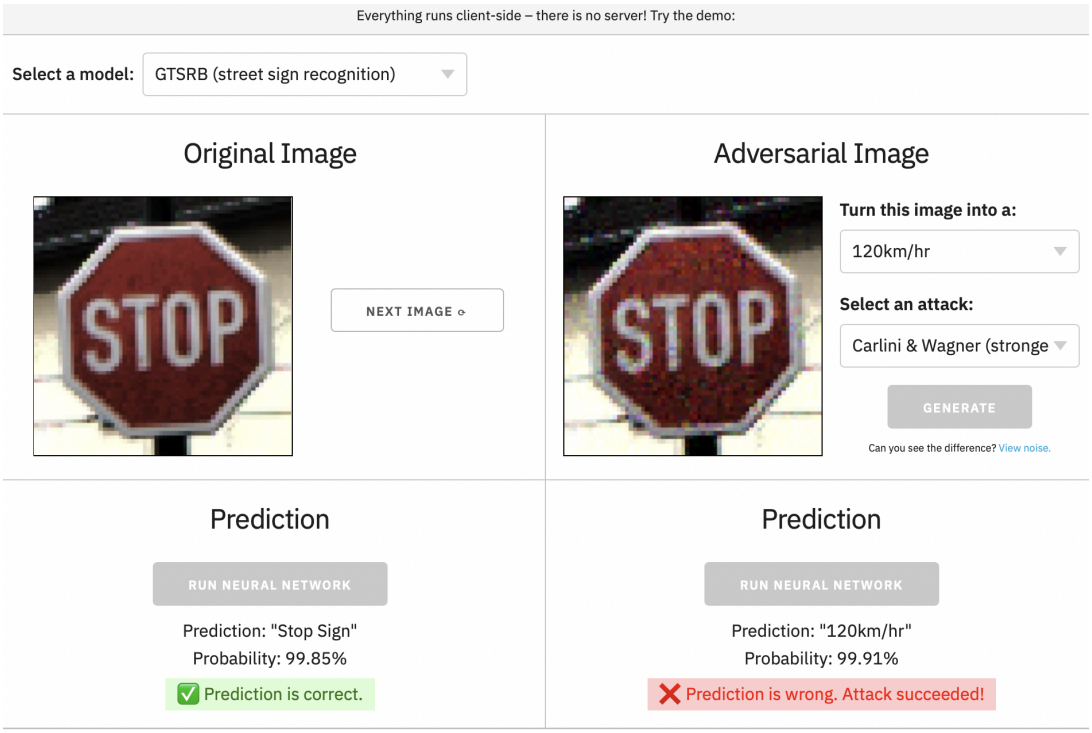


Figure 4: Example of Carlini & Wagner attack. Source: https://kennysong.github.io/adversarial.js/

- In 2020, Andriushchenko et al. [6] introduced **Square Attack**, a novel black box method designed to be effective in L_2 and L_∞. What sets it apart is its independence from local gradient information, making it immune to gradient masking. The Square Attack employs a randomized search strategy, selecting square-shaped perturbations at random positions. This approach keeps the perturbations near the boundary of feasible changes during each iteration.

2.4. Defense strategies against adversarial attacks

This section touches on the defense strategies against adversarial attacks. Firstly, we define the security evaluation mechanism, which is the general framework on how to improve the robustness of the Machine Learning model. Wang et al. [10] mentioned two methods for machine learning model security evaluation: reactive defense and proactive defense. A **reactive defense** mechanism involves one group of machine learning model designers against actual attackers. As the attacker continues to exploit the weakness of the model pipeline, the designer continues to improve through analysis and counteracting new attacks. In contrast, **proactive**

defense proposes a well-controlled evaluation schema where two teams of Machine learning specialist, actively tries to perform adversaries against the model whilst the other actively protects it.

Against poisoning attacks, **data sanitization** is a popular technique with instant results, which aims to detect and remove potential instances that may be a threat to the machine learning model. Moreover, designers of these systems may of for models that can generalize well, particularly ensemble methods such as Bagging and Random Forest. Finally, **robust statistics** aims to develop algorithms that can resist or mitigate corrupted data points and can be used to enhance the model's reliability.

Against evasion and exploratory attacks (or attacks at the testing phase), several strategies and techniques can be employed to further improve the model's robustness. **Adversarial Training** involves augmenting the training dataset with adversarial examples generated during the training process. By adding adversarial inputs, the machine learning model can adapt and become more robust against similar attacks in the future. **Gradient Masking** is also a popular method that obscures gradient information during training to make it more challenging for attackers to manipulate the model effectively. In addition, data anonymization techniques, such as **differential privacy,** are also used for defense against testing attacks: this enforces a randomization technique that ensures the model output does not leave any information about the record in training data; however, this comes at the cost of utility (the information retained after anonymization is reduced).

3. Commercial and open-source tools for privacy analytics

In this section, three different tools will be mentioned for privacy analytics. These tools are the Adversarial Robustness Toolbox (ART), RobustML, Cleverhans, ARX - Data Anonymization Tool, and Diffprivlib.

3.1. Adversarial Robustness Toolbox

Adversarial Robustness Toolbox (ART) [7] is an open-sourced adversarial artificial attack and defense written in Python. There are 6 main categories of implementation in ART: attacks, defenses, estimators, evaluations, metrics, and preprocessing. The library is compatible with machine learning and deep learning Python framework (scikit-learn, tensorflow, PyTorch, etc.)

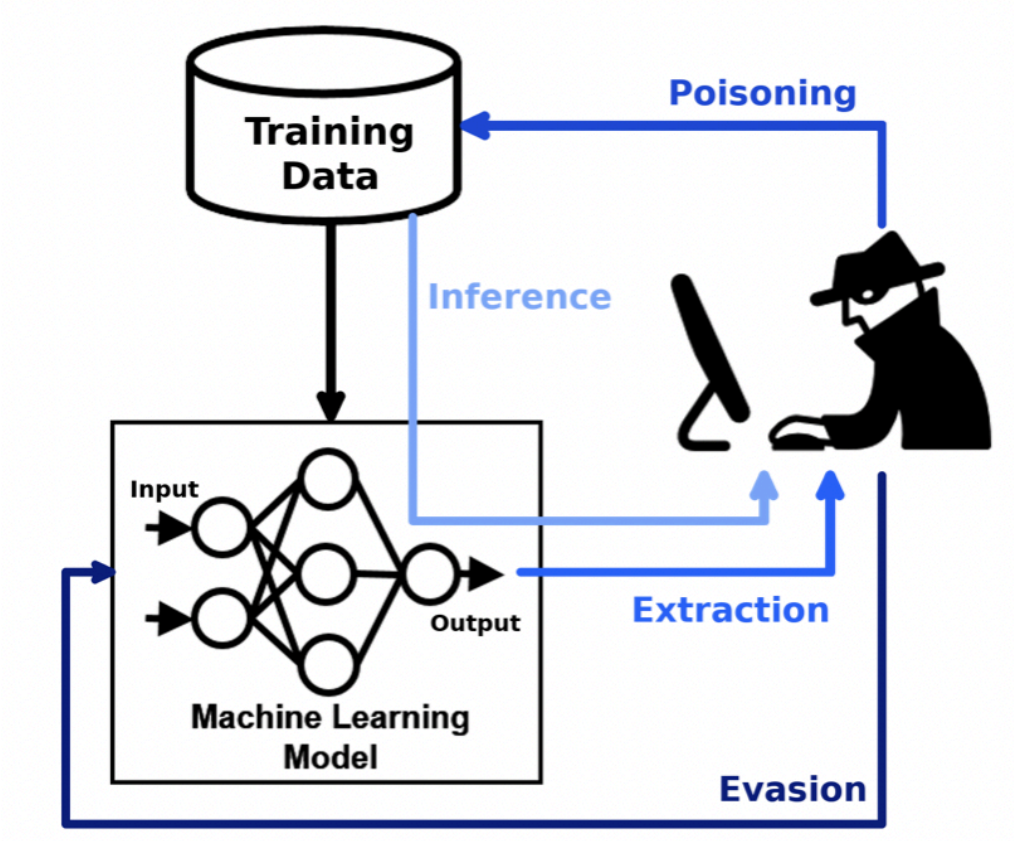


Figure 5: Attack types visualization. Source: https://github.com/Trusted-AI/adversarial-robustness-toolbox/wiki

The image above provides the attacking types supported by ART and visualizes the flow of four different attacks, including Evasion (e.g. Carlini & Wagner, Basic Iterative Method), Poisoning (e.g. Sleeper Agent, BadDet), Extraction (e.g. CopyCat CNN), and Inference (e.g. Attribute Inference White-box Decision Tree) attacks. It makes two further sub-classifications of explanatory attack called inference (extracting information from the training dataset), and extraction (extracting information from the machine learning model).

Each attack is carefully written with framework optimization in mind:

- all/Numpy: written in Numpy to support the majority or all framework
- Tensorflow: optimized for Tensorflow deep learning library
- PyTorch: optimized for PyTorch deep learning library

Adversarial Robustness Toolbox supports the Red team vs Blue team defense schema, where one team actively tries to attack the Machine Learning pipeline, and the other comes up with a solution against those attacks. Therefore, multiple defense mechanisms have also been implemented in ART. The main defense categories implementations are: preprocessors, postprocessors, trainers, transformers, and detectors.

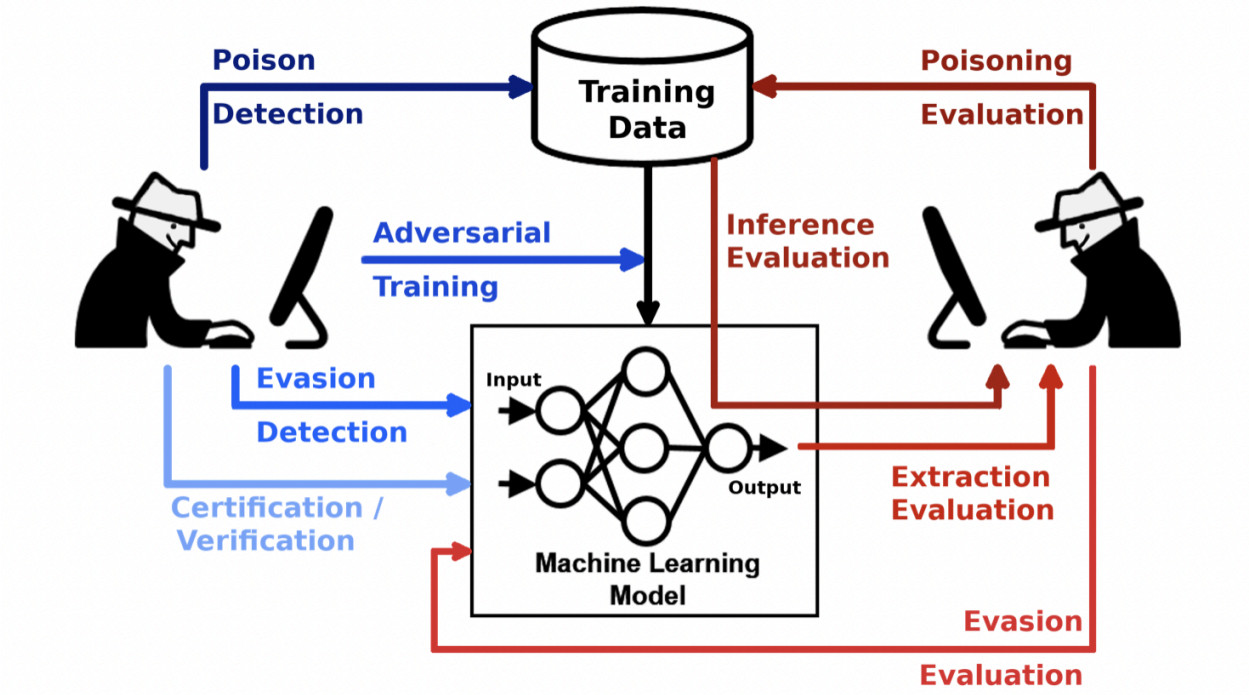


Figure 6: Attack and defense visualization by red and blue team schema.

Source: https://github.com/Trusted-AI/adversarial-robustness-toolbox/wiki

3.2. RobustML

RobustML is a Python community-driven library focusing on Robust machine learning for computer vision. The project is maintained by various scholars from prestigious institutes such as Google Brain, UC Berkeley, MIT, etc. It provides implementation of popular open-sourced defenses (and attacks against those defenses mentioned in the paper) and analysis against adversarial machine learning. Specifically, these include defenses that target whitebox attacks, the most vulnerable attack capability compared to gray-box and black-box attacks. These implementations may either be published or unpublished, and below are examples of implemented defense, including the published papers, published venues, threat models, and natural accuracy versus claimed accuracy (for reduction reference after applying mentioned defense against mentioned attack).

Defense	Venue	Dataset	Threat Model	Natural Accuracy	Claims	Analyses
Provably Robust Boosted Decision Stumps and Trees against Adversarial Attacks (Andriushchenko & Hein) (code)	NeurIPS 2019	MNIST	$\ell_\infty(\epsilon = 0.3)$	97.32% accuracy	87.54% accuracy (certified)	
		FMNIST	$\ell_\infty(\epsilon = 0.1)$	85.85% accuracy	76.83% accuracy (certified)	
Harnessing the Vulnerability of Latent Layers in Adversarially Trained Models (Sinha et al.) (code)	IJCAI 2019	CIFAR-10	$\ell_\infty(\epsilon = 0.03)$	87.8% accuracy	53.82% accuracy	
Provable Robustness of ReLU networks via Maximization of Linear Regions (Croce et al.) (code)	AISTATS 2019	MNIST	$\ell_\infty(\epsilon = 0.1)$	98.81% accuracy	96.42% accuracy (empirical), 96.37% accuracy (certified)	
		FMNIST	$\ell_\infty(\epsilon = 0.1)$	85.50% accuracy	(on first 1000 test points) 73.4% accuracy (empirical), 69.3% accuracy (certified)	
		GTS	$\ell_2(\epsilon = 0.2)$	84.65% accuracy	(on first 1000 test points) 67.9%	

Figure 7: Example of defense in RobustML. Source: https://www.robust-ml.org/defenses/

In order for defense or analysis to be added to this curated list, it must be

- Open-sourced
- Implemented in robustML API style guide
- Defense for computer vision
- Attack against that defense (if needed)
- Tested against specific paper (currently MNIST, Fashion-MNIST, CIFAR-10, GTS, ImageNet ILSVRC 2012) [8]

3.3. Cleverhans

Cleverhans is written and maintained by the Cleverhans Laboratories. Contributors to this project have various backgrounds, including Google Brain researchers, University of Toronto researchers, etc. This project aims at the implementation of state-of-the-art attacks and defenses for adversarial artificial intelligence, while the general goal is constructing and building defenses and benchmarks. Similar to RobustML, this project welcomes contributions from outsiders; the difference is the focus on defense: whereas Cleverhans focus on general high-performing algorithms, RobustML only focuses on whitebox attacks.

The project officially supports three libraries: JAX, PyTorch, and Tensorflow 2. In addition, a blog written by Ian Goodfellow and Nicolas Papernot called **cleverhans-blog** includes various resources, and blog posts on the implementations of the project.

3.4. ARX - Data Anonymization Tool

Within this chapter, ARX is the only tool that solely focuses on Data Anonymization, a process of processing and transforming sensitive data and enhancing privacy using various privacy models. The privacy models can be categorized into:

- Syntactic privacy model: k-anonymity, l-diversity, t-closeness, etc.
- Statistical privacy model: k-map, the threshold on average risks, etc.
- Semantic privacy model: differential privacy, game-theoretic de-identification.

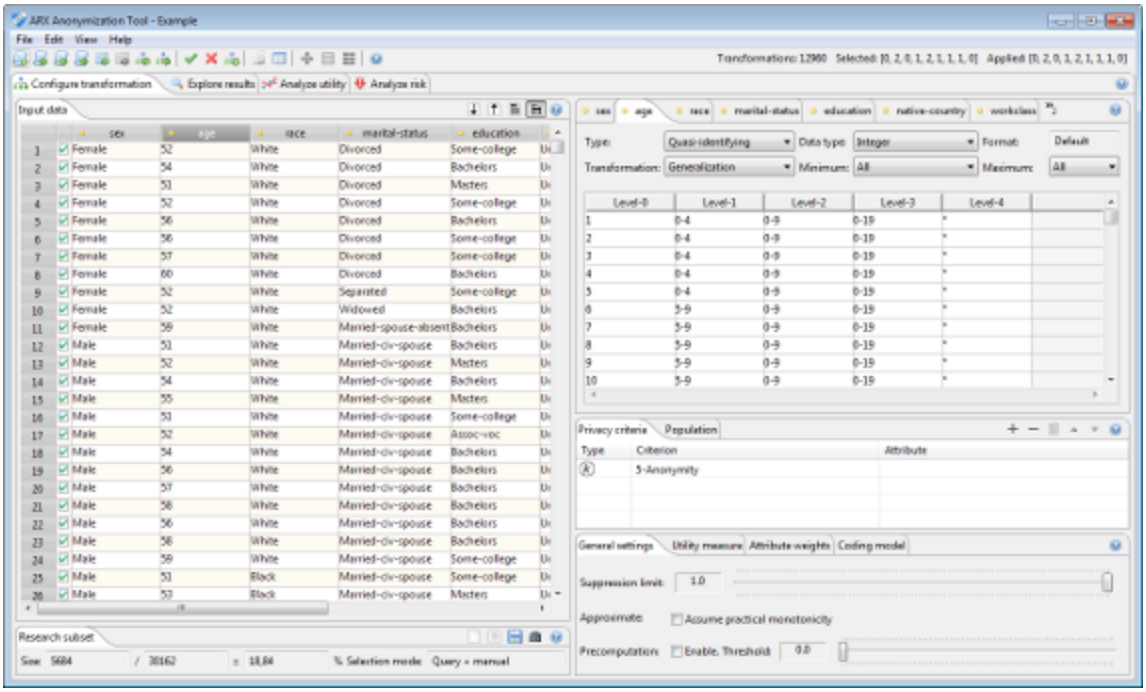


Figure 8: Example of ARX tool. Source: own screenshot

In conjunction, the toolkit also offers various transformation schemas (e.g. global or local transformation schemas, top-, and bottom-coding) and data quality models (e.g. cell-oriented, attribute-oriented).

The toolbox comes in a desktop application available on multiple platforms or API (Java). In addition, data analysis tools focus on utility analysis, or the usability of the anonymized data compared to the original dataset, an important aspect of data anonymization. Some statistical data is measured, including information loss, squared error, and classification performance (based on built-in logistic regression, random forest, and naive Bayes). Finally, risk analysis is built into the tool, including re-identification risk, population uniqueness analysis, etc.

3.5. Diffprivlib

Diffprivlib is an abbreviation of Differential Privacy Library, an open-sourced library created and maintained by IBM. This Python library is designed to support integrating differential privacy into machine learning workflow. Diffprivlib is a Python library and is built toward compatibility

with existing libraries such as Numpy, Scikit-learn, and PyTorch. Some of the key features of Diffprivlib include:

- Differential privacy: built-in methods such as Laplace noise mechanism and Gaussian noise
- Differential privacy integrated aggregation: queries such as sum, count, mean, etc. can be done with protection
- Machine learning models: the library includes some differential privacy protected machine learning algorithms: naive Bayes, random forest, k-mean, etc.
- Utility metrics: Similar to ARX, the library support multiple utility measurements for privacy-utility tradeoff

3.6. Summary

We will summarize the aforementioned tools for adversarial artificial intelligence and highlight their main features in the below table for comparison:

	Name	Type	Features	Link
1	**Adversarial Robustness Toolbox**	Open-source	- Active development & clear roadmap - Include implementations of state-of-the-art methods - Implement adversarial attacks, defenses, evaluation, estimators, metrics, preprocessing - Clear documentation	adversarial-robustness-toolbox
2	**RobustML**	Open-source	- Active development - Community-driven - Implement adversarial defenses (and some attack mentioned for those	https://github.com/robust-ml/robustml

			defenses) and analysis - Focus primarily on white-box attacks, the highest attacking capability - Clear guideline on how to a defense can be included in RobustML API	
3	**Cleverhans**	Open-source	- Active development - Open to community contribution - Implementation of state-of-the-art adversarial attacks and defense - Clear guideline on how to contribute - Clear documentation and blog post	Github: https://github.com/cleverhans-lab/cleverhans Blog: http://www.cleverhans.io/
4	**ARX - Data Anonymization Tool**	Open-source	- Active development - Open to community contribution - Implement various data anonymization techniques, transformation schemas - Built-in utility analysis post-transformation - Built-in risk analysis toolset - Clear guideline on how to contribute	Paper implementations: https://arx.deidentifier.org/publications/ Main page: https://arx.deidentifier.org/
5	**Diffprivlib**	Open-source	- Active development - Focus on differential privacy technology	https://github.com/IBM/differential-privacy-library

			- Built-in aggregation methods, machine learning, utility metrics	/tree/main

4. Practical case studies

In this chapter, one real-life case study will be provided and summarized, based on the paper "'*Real Attackers Don't Compute Gradient': Bridging The Gap Between Adversarial ML Research And Practice"* by Appruzzese et al. [11]. This paper emphasizes the difference between scholarly adversarial scenarios studied by academics and compares it to real-life case studies.

This case study mentioned Facebook, a popular social network, handles spam detection on its platform. Spam here refers to a social post that is designed to trick people into clicking on a link, taking some other action, or spreading explicitly banned content (e.g. spreading gun violence). The theoretic scenario that most papers will assume is a single ML spamming detection system with unknown architecture (blackbox) deployed for spam detection. However, the real-life system is far from that assumption: it consists of multi-layer filters with different functionalities, and ML is only utilized at the last step of the equation. Below is a description of the spam filtering system deployed at Facebook:

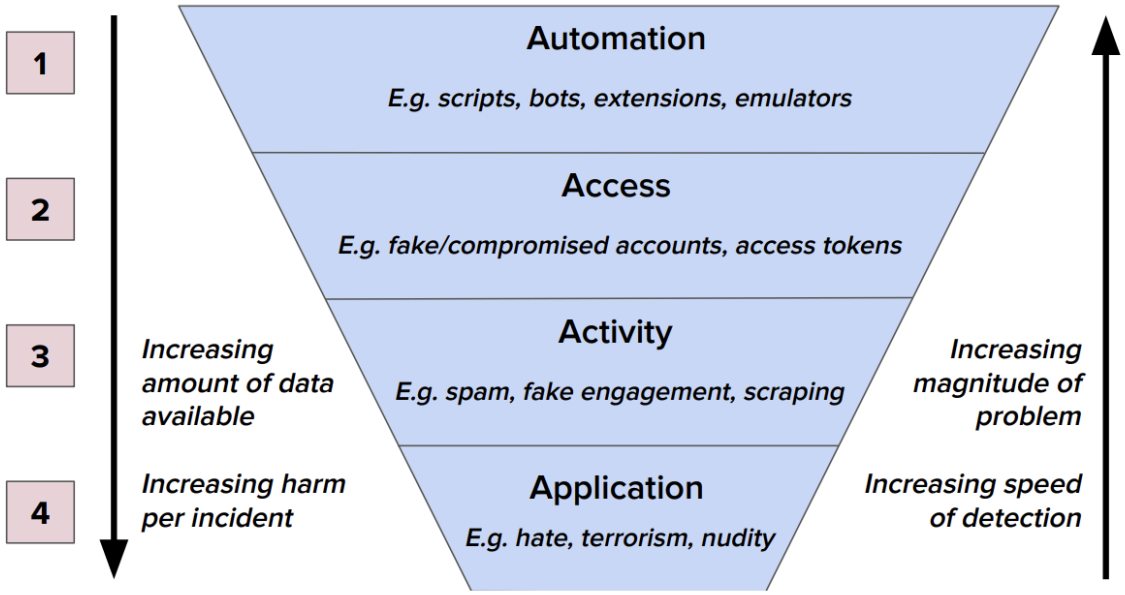


Figure 8: Example of ARX tool. Source: own screenshot

We start with the *Automation* layer, where the system will classify if our post is carried out by an automated action or not. Facebook achieves this by observing the reads and writes frequencies, as well as the sign-in and sign-out ratio. This eliminates some attempts of attackers planning to do exploratory attacks, which require constant retrieval of output by trying different inputs. The *Access* layer is the second layer of the Facebook spam filtering system. It is responsible for analyzing write requests and determining whether the corresponding post was carried out by someone who is authorized, such as a logged-in user or an advertising account. Through this second layer, bad accounts can be detected and stopped before advancing further. The third layer is called *Activity*, which analyzes a person's activity during the usage. This can include their interactions and social graph activities. Any spam post passing the third layer should be carefully crafted accounts, which will seamlessly be active as a normal account, posting meticulously written spam that cannot be detected, in which case, it is reasonable to bypass the first three layers. Finally, the *Application* layer is where machine learning comes in: specially trained models can be used to detect hate, terrorism, and nudity, preventing the spread of misinformation and violence.

We can draw from the case study above that the focus of researchers and real-life scenarios of adversarial attacks may differ from each other. Balancing between the two can become difficult: on one hand, we cannot be naive and make the assumption that such an intricate attack will not happen and should not be considered; on the other, we still need to develop a more realistic research direction for the scholars towards the future that is suitable for real-life.

5. Future research directions

In this chapter, we continue the idea from the previous real-life case study to expand upon future research directions. As we have seen, researchers must reconsider their focus with regard to theory and real-life scenarios. We cannot completely ignore the effectiveness of advanced adversarial attacks on Machine learning systems, nor can we deny that the current

state of research has deviated from the real world. Therefore, there should be a sub-field that specifically targets applied adversarial AI, where we expand upon these ideas:

- A system can be built on multiple different security layers, which may or may not incorporate Machine Learning
- Detection of hand-crafted adversarial data that cannot be easily assessed (edge cases)

Finding a balance between those two ideas can be challenging, and it opens to many questions such as when do we say stop on the adversarial attack strength, how much non-ML approach could prevent these attacks from happening, thereby we can narrow the target these ML models have to focus on.

6. Conclusion

In this report, we give a comprehensive summary of adversarial machine learning, including the taxonomy, and some examples of state-of-the-art attacks and defense. and include a short list of five open-source tools for adversarial attacks and/or defense. We also mention a real-life case study of Facebook spamming detection system, and follow up with the future directions.

7. References

[1] Battista Biggio, Blaine Nelson, & Pavel Laskov. (2013). Poisoning Attacks against Support Vector Machines.

[2] Tianyu Gu, Brendan Dolan-Gavitt, & Siddharth Garg. (2019). BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain.

[3] Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, & Tom Goldstein. (2022). Sleeper Agent: Scalable Hidden Trigger Backdoors for Neural Networks Trained from Scratch.

[4] Shih-Han Chan, Yinpeng Dong, Jun Zhu, Xiaolu Zhang, & Jun Zhou. (2022). BadDet: Backdoor Attacks on Object Detection.

[5] Nicholas Carlini, & David Wagner. (2017). Towards Evaluating the Robustness of Neural Networks

[6] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, & Matthias Hein. (2020). Square Attack: a query-efficient black-box adversarial attack via random search

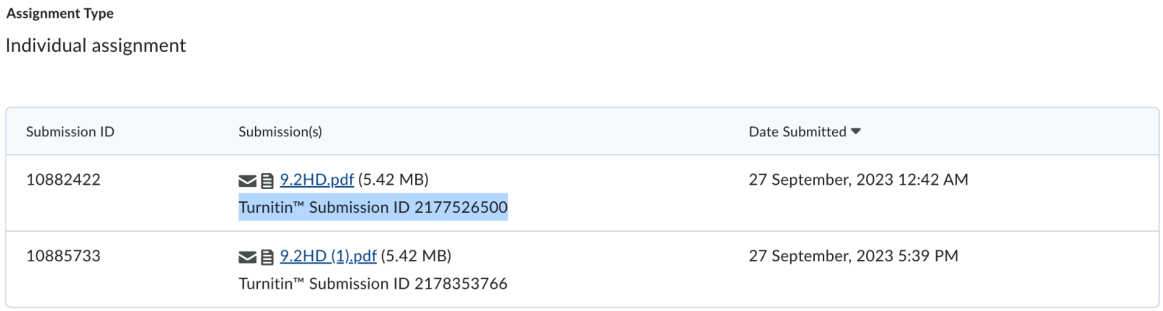
[7] Adversarial Robustness Toolbox author (n.d.), Adversarial Robustness Toolbox. Available at: https://github.com/Trusted-AI/adversarial-robustness-toolbox

[8] RobustML author (n.d.), FAQ. Available at: https://www.robust-ml.org/faq/

[10] Wang, X., Li, J., Kuang, X., Tan, Y. A., & Li, J. (2019). The security of machine learning in an adversarial setting: A survey. Journal of Parallel and Distributed Computing, 130, 12-23.

[11] Giovanni Apruzzese, Hyrum S. Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, & Kevin A. Roundy. (2022). "Real Attackers Don't Compute Gradients": Bridging the Gap Between Adversarial ML Research and Practice.

Screenshot of Turnitin submission



Assignment Type

Individual assignment

Submission ID	Submission(s)	Date Submitted ▾
10882422	9.2HD.pdf (5.42 MB) Turnitin™ Submission ID 2177526500	27 September, 2023 12:42 AM
10885733	9.2HD (1).pdf (5.42 MB) Turnitin™ Submission ID 2178353766	27 September, 2023 5:39 PM