AIJack
Security and Privacy Risk Simulator for Machine Learning

Hideaki Takahashi (Koukyosyumei)
The University of Tokyo
takahashi-hideaki567@g.ecc.u-tokyo.ac.jp

December 30, 2023

Abstract

This paper introduces **AIJack**, an open-source library designed to assess security and privacy risks associated with the training and deployment of machine learning models. Amid the growing interest in big data and AI, advancements in machine learning research and business are accelerating. However, recent studies reveal potential threats, such as the theft of training data and the manipulation of models by malicious attackers. Therefore, a comprehensive understanding of machine learning's security and privacy vulnerabilities is crucial for the safe integration of machine learning into real-world products. AIJack aims to address this need by providing a library with various attack and defense methods through a unified API. The library is publicly available on GitHub (https://github.com/Koukyosyumei/AIJack).

1 Introduction

Machine learning (ML) has become a foundational component of diverse applications, spanning image recognition to natural language processing. As these technologies proliferate, the need to comprehend and address security risks associated with ML models becomes imperative.

While ML models continuously enhance accuracy, attackers can significantly diminish it by introducing artificially created data. Evasion Attacks [1, 2], using specialized algorithms, can induce malfunctions in models. Consider a road sign classification model; by adding imperceptible noise to a "speed limit 30km" sign, it may misclassify it as "speed limit 60km." Another attack, Poisoning Attack [3], injects contaminated data during training, lowering model accuracy.

To counter such attacks, various strategies have been proposed. Certified robustness [4, 5], a technique to formally guarantee that adversarial examples cannot lead to undesirable predictions, has gained attention. Another approach is debugging machine learning models [6, 7, 8], which aims to identify inputs causing unexpected behaviors.

ML also poses privacy challenges. Collecting large amounts of data for training can infringe on privacy, leading to data breaches. Model Inversion Attacks [9, 10] reconstruct training data from pre-trained models, threatening sensitive information. Similar attacks, like Membership Inference Attacks [11], aim to determine if a data point is part of the model's training data.

Privacy protection techniques, including differential privacy [12], k-anonymity [13], homomorphic encryption [14], and distributed learning [15, 16], have been proposed. Differential privacy prevents individual data inference, while homomorphic encryption enables arithmetic operations on encrypted data. Distributed methods like Federated Learning (FL) [15] and Split Learning [16] facilitate collaborative learning among data owners.

It is crucial to assess the security and privacy risks of ML models and evaluate countermeasure effectiveness. To simplify such simulations, we propose the open-source software, **AIJack**, offering various attack and defense methods. AIJack enables experimenting with various combinations of attacks and defenses with simple code. Built on PyTorch [17] and scikit-learn [18], users can easily incorporate AIJack into existing code.

2 Package Design

Type	Subcategory	Method
Collaborative	Horizontal FL	FedAVG [15], FedProx [19], FedMD [20], FedGEMS [21], DSFL [22], MOON [23], FedEXP [24]
Collaborative	Vertical FL	SplitNN [16], SecureBoost [25]
Attack	Model Inversion	MI-FACE [9], DLG [10], iDLG [26], GS [27], CPL [28], GradInversion [29], GAN Attack [30]
Attack	Label Leakage	Norm Attack [31]
Attack	Poisoning	History Attack [32], Label Flip [32], MAPF [32], SVM Poisoning [3]
Attack	Backdoor	DBA [33], Model Replacement [34]
Attack	Free-Rider	Delta-Weight [35]
Attack	Evasion	Gradient-Descent Attack [1], FGSM [2], DIVA [36]
Attack	Membership Inference	Shadow Attack [11]
Defense	Homomorphic Encryption	Paillier [14]
Defense	Differential Privacy	DPSGD [12], AdaDPS [37], DPlis [38]
Defense	Anonymization	Mondrian [13]
Defense	Robust Training	PixelDP [4], Cost-Aware Robust Tree Ensemble [5]
Defense	Debugging	Model Assertions [6], Rain [7], Neuron Coverage [8]
Defense	Others	Soteria [39], FoolsGold [40], MID [41], Sparse Gradient [42]

Table 1: List of Supported Algorithms

AIJack is designed with the following principles:

- *All-around abilities for both attack and defense*: AIJack provides a flexible API for over 40 attack and defense algorithms (see Tab. 1 for the comprehensive lists). Users can experiment with various combinations of these methods.
- *PyTorch-friendly design*: AIJack supports many PyTorch models, allowing integration with minimal modifications to the original code.
- *Compatibility with scikit-learn*: AIJack supports many scikit-learn models, enabling easy integration of attacks and defenses.
- *Fast Implementation with C++ backend*: AIJack employs a C++ backend for components like Differential Privacy and Homomorphic Encryption, enhancing scalability.
- *MPI-Backend for Federated Learning*: AIJack supports MPI-backed Federated Learning for deployment in High-Performance Computing systems.
- *Extensible modular APIs*: AIJack comprises simple modular APIs, allowing easy extension with minimal effort.

Code 1 and Code 2 show the example codes implementing Evasion Attack and Poisoning Attack against SVM trained with scikit-learn, respectively.

```
from aijack.attack import (
    Evasion_attack_sklearn
)

attacker = Evasion_attack_sklearn(
    clf, initial_data_positive
)
malicious_data, log = attacker.attack(
    initial_data_negative
)
```

Code 1: Evasion Attack against SVM

```
from aijack.attack import (
    Poison_attack_sklearn
)

attacker = Poison_attack_sklearn(
    clf, X_train, y_train
)
malicious_data, log = attacker.attack(
    initial_data, 1, X_valid, y_valid
)
```

Code 2: Poisoning Attack against SVM

Example codes for Federated Learning can be found in Code 3 to Code 6. Code 3 implements FedAVG with PyTorch, the standard Federated Learning protocol, Code 4 implements a Gradient Inversion Attack against Federated Learning (a type of Model Inversion Attack), Code 5 applies Paillier Encryption, one of the most popular Homomorphic Encryption methods, to mitigate inversion attacks, and Code 6 migrates Code 3 to MPI-backend.

```python
from aijack.collaborative import (
  FedAVGClient,
  FedAVGServer,
  FedAVGAPI
)









clients = [FedAVGClient(model_1),
           FedAVGClient(model_2)]
server = FedAVGServer(clients, model_3)

api = FedAVGAPI(
    server, clients,
    criterion, optimizers, dataloaders,
)
api.run()
```

Code 3: Standard Federated Learning (FL)

```python
from aijack.collaborative import (
  FedAVGClient,
  FedAVGServer,
  FedAVGAPI
)
from aijack.attack import (
  GradientInversionAttackServerManager
)

mg = GradientInversionAttackServerManager(
    input_shape,
)
MaliciousServer = mg.attach(FedAVGServer)

clients = [FedAVGClient(model_1),
           FedAVGClient(model_2)]
server = MaliciousServer(clients, model_3)

api = FedAVGAPI(
    server, clients,
    criterion, optimizers, dataloaders,
)
api.run()
```

Code 4: Gradient Inversion Attack against FL

```python
from aijack.collaborative import (
  FedAVGClient,
  FedAVGServer,
  FedAVGAPI
)
from aijack.defense import (
  PaillierGradientClientManager,
  PaillierKeyGenerator
)

keygenerator = PaillierKeyGenerator(512)
pk, sk = keygenerator.generate_keypair()

mg = PaillierGradientClientManager(pk, sk
    )
PaillierClient = mg.attach(FedAVGClient)

clients = [PaillierClient(model_1),
           PaillierClient(model_2)]
server = FedAVGServer(clients, model_3)

api = FedAVGAPI(
    server,
    clients,
    criterion,
    optimizers,
    dataloaders,
)
api.run()
```

Code 5: FL with Paillier Encryption

```python
from aijack.collaborative import (
  FedAVGClient,
  FedAVGServer,
  MPIFedAVGAPI,
  MPIFedAVGClientManager,
  MPIFedAVGServerManager
)

mcm = MPIFedAVGClientManager()
msm = MPIFedAVGServerManager()
MPIClient = mcm.attach(FedAVGClient)
MPIGServer = msm.attach(FedAVGServer)

if myid == 0:
    client_ids = list(range(1, size))
    server = MPIServer(comm, client_ids, ◦
     model)
    api = MPIFedAVGAPI(
        comm, server, True, criterion
    )
else:
    client = MPIClient(comm, model,
    user_id=myid)
    api = MPIFedAVGAPI(
        comm, client, False, criterion,
        optimizer, dataloader
    )

api.run()
```

Code 6: FL with MPI backend

Code7 illustrates the standard workload involved in training a neural network using Differential Private Stochastic Gradient Descent (DPSGD). Our API's design draws inspiration from Opacus [43], a renowned framework dedicated to deep learning with differential privacy. Additionally, our AIJack facilitates a more intuitive implementation, closely aligning with the algorithm initially proposed in[12].

Additional examples are available in our documentation (https://koukyosyumei.github.io/AIJack/).

```python
from aijack.defense import GeneralMomentAccountant, DPSGDManager

accountant = GeneralMomentAccountant(noise_type="Gaussian", backend="cpp")
privacy_manager = DPSGDManager(accountant, optim.SGD)

accountant.reset_step_info()
dpoptimizer_cls, lot_loader, batch_loader = privacy_manager.privatize(
    noise_multiplier=sigma
)
optimizer = dpoptimizer_cls(net.parameters(), lr=lr)

for epoch in range(num_iterations):
    for X_lot, y_lot in lot_loader(optimizer):
        for X_batch, y_batch in batch_loader(TensorDataset(X_lot, y_lot)):
            optimizer.zero_grad()
            loss = criterion(net(X_batch), y_batch)
            loss.backward()
            optimizer.step()
```

Code 7: Deep Learning with Differential Privacy

Acknowledgement

We acknowledge every contributor for their support of this project. The comprehensive list of contributors can be found in our GitHub repository.

References

[1] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*, pages 387–402. Springer, 2013.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[3] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[4] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE symposium on security and privacy (SP)*, pages 656–672. IEEE, 2019.

[5] Yizheng Chen, Shiqi Wang, Weifan Jiang, Asaf Cidon, and Suman Jana. {Cost-Aware} robust tree ensembles for security applications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2291–2308, 2021.

[6] Daniel Kang, Deepti Raghavan, Peter D. Bailis, and Matei A. Zaharia. Model assertions for debugging machine learning. 2018.

[7] Weiyuan Wu, Lampros Flokas, Eugene Wu, and Jiannan Wang. Complaint-driven training data debugging for query 2.0. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1317–1334, 2020.

[8] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 1–18, New York, NY, USA, 2017. Association for Computing Machinery.

[9] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.

[10] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

[11] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[12] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[13] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 25–25, 2006.

[14] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[15] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[16] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[19] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[20] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[21] Sijie Cheng, Jingwen Wu, Yanghua Xiao, and Yang Liu. Fedgems: Federated learning of larger server models via selective knowledge fusion. *arXiv preprint arXiv:2110.11027*, 2021.

[22] Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Transactions on Mobile Computing*, 22(1):191–205, 2021.

[23] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.

[24] Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. Fedexp: Speeding up federated averaging via extrapolation. *arXiv preprint arXiv:2301.09604*, 2023.

[25] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.

[26] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[27] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[28] Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. A framework for evaluating gradient leakage attacks in federated learning. *arXiv preprint arXiv:2004.10397*, 2020.

[29] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16337–16346, 2021.

[30] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 603–618, 2017.

[31] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning. *arXiv preprint arXiv:2102.08504*, 2021.

[32] Xiaoyu Cao and Neil Zhenqiang Gong. Mpaf: Model poisoning attacks to federated learning based on fake clients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3396–3404, 2022.

[33] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020.

[34] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948. PMLR, 26–28 Aug 2020.

[35] Jierui Lin, Min Du, and Jian Liu. Free-riders in federated learning: Attacks and defenses. *arXiv preprint arXiv:1911.12560*, 2019.

[36] Wei Hao, Aahil Awatramani, Jiayang Hu, Chengzhi Mao, Pin-Chun Chen, Eyal Cidon, Asaf Cidon, and Junfeng Yang. A tale of two models: Constructing evasive attacks on edge models. *Proceedings of Machine Learning and Systems*, 4:414–429, 2022.

[37] Tian Li, Manzil Zaheer, Sashank Reddi, and Virginia Smith. Private adaptive optimization with side information. In *International Conference on Machine Learning*, pages 13086–13105. PMLR, 2022.

[38] Wenxiao Wang, Tianhao Wang, Lun Wang, Nanqing Luo, Pan Zhou, Dawn Song, and Ruoxi Jia. Dplis: Boosting utility of differentially private deep learning via randomized smoothing. *arXiv preprint arXiv:2103.01496*, 2021.

[39] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. Provable defense against privacy leakage in federated learning from representation perspective. *arXiv preprint arXiv:2012.06043*, 2020.

[40] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[41] Tianhao Wang, Yuheng Zhang, and Ruoxi Jia. Improving robustness to model inversion attacks via mutual information regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11666–11673, 2021.

[42] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 440–445, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[43] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*, 2021.