ETSI TR 104 222 V1.2.1 (2024-07)

TECHNICAL REPORT

Securing Artificial Intelligence; Mitigation Strategy Report

Reference

RTR/SAI-009

Keywords

artificial intelligence, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Important notice

The present document can be downloaded from:
https://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

If you find a security vulnerability in the present document, please report it through our Coordinated Vulnerability Disclosure Program:
https://www.etsi.org/standards/coordinated-vulnerability-disclosure

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.
No recommendation as to products and services or vendors is made or should be implied.
No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.
In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Securing Artificial Intelligence (SAI).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document summarizes and analyses existing and potential mitigation against threats for AI-based systems as discussed in ETSI GR SAI 004 [i.1]. The goal is to have a technical survey for mitigating against threats introduced by adopting AI into systems. The technical survey shed light on available methods of securing AI-based systems by mitigating against known or potential security threats. It also addresses security capabilities, challenges, and limitations when adopting mitigation for AI-based systems in certain potential use cases.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GR SAI 004: "Securing Artificial Intelligence (SAI); Problem Statement".

[i.2] Doyen Sahoo, Quang Pham, Jing Lu and Steven C. H. Hoi: "Online Deep Learning: Learning Deep Neural Networks on the Fly", International Joint Conferences on Artificial Intelligence Organization, 2018.

[i.3] Battista Biggio and Fabio Roli: "Wild patterns: Ten years after the rise of adversarial machine learning", Pattern Recognition, 2018.

[i.4] Qiang Liu, Pan Li, Wentao Zhao, Wei Cai, Shui Yu and Victor C. M. Leung: "A Survey on Security Threats and Defensive Techniques of Machine Learning: A Data Driven View". IEEE Access 2018.

[i.5] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha and Michael P. Wellman: "SoK: Security and Privacy in Machine Learning". IEEE European Symposium on Security and Privacy (EuroS&P) 2018.

[i.6] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang and Anil K. Jain: "Adversarial Attacks and Defenses in Images, Graphs and Text: A Review". International Journal of Automation and Computing volume 17, pages151-178(2020).

[i.7] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay and Debdeep Mukhopadhyay: "Adversarial Attacks and Defences: A Survey", arXiv preprint arXiv:1810.00069v1.

[i.8] Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu and Jinwen He: "Towards Privacy and Security of Deep Learning Systems: A Survey". IEEE Transactions on Software Engineering 2020.

[i.9] NIST IR 8269-(Draft): "A Taxonomy and Terminology of Adversarial Machine Learning".

[i.10] Christian Berghoff1, Matthias Neu1 and Arndt von Twickel: "Vulnerabilities of Connectionist AI Applications: Evaluation and Defence", Frontiers in Big Data volume 3, 2020.

[i.11] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles A. Sutton, J. Doug Tygar and Kai Xia: "Exploiting Machine Learning to Subvert Your Spam Filter", Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET) 2008.

[i.12] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru and Bo Li: "Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning", IEEE Symposium on Security and Privacy 2018: 19-35.

[i.13] Nathalie Baracaldo, Bryant Chen, Heiko Ludwig and Jaehoon Amir Safavi: "Mitigating Poisoning Attacks on Machine Learning Models: A Data Provenance Based Approach", AISec@CCS 2017: 103-110.

[i.14] Sanghyun Hong, Varun Chandrasekaran, Yigitcan Kaya, Tudor Dumitras and Nicolas Papernot: "On the Effectiveness of Mitigating Data Poisoning Attacks with Gradient Shaping", arXiv: 2002.11497v2.

[i.15] Nitika Khurana, Sudip Mittal, Aritran Piplai and Anupam Joshi: "Preventing Poisoning Attacks On AI Based Threat Intelligence Systems", IEEE International Workshop on Machine Learning for Signal Processing (MLSP) 2019: 1-6.

[i.16] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto and Fabio Roli: "Bagging Classifiers for Fighting Poisoning Attacks in Adversarial Classification Tasks", International Workshop on Multiple Classifier Systems (MCS) 2011: 350-359.

[i.17] Yao Cheng, Cheng-Kang Chu, Hsiao-Ying Lin, Marius Lombard-Platet and David Naccache: "Keyed Non-parametric Hypothesis Tests", International Conference on Network and System Security (NSS) 2019: 632-645.

[i.18] Tran, Brandon, Jerry Li and Aleksander Madry: "Spectral signatures in backdoor attacks", In Advances in Neural Information Processing Systems, pp. 8000-8010. 2018.

[i.19] Chen, Bryant, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy and Biplav Srivastava: "Detecting backdoor attacks on deep neural networks by activation clustering", Artificial Intelligence Safety Workshop @ AAAI, 2019.

[i.20] Yuntao Liu, Yang Xie and Ankur Srivastava: "Neural Trojans", 2017 IEEE International Conference on Computer Design (ICCD), Boston, MA, 2017, pp. 45-48, doi: 10.1109/ICCD.2017.16.

[i.21] Bingyin Zhao and Yangjie Lao: "Resilience of Pruned Neural Network against poisoning attack", International Conferecne Conference on Malicious and Unwanted Software (MALWARE) 2018, page 78-83.

[i.22] Liu, Kang, Brendan Dolan-Gavitt and Siddharth Garg: "Fine-pruning: Defending against backdooring attacks on deep neural networks", In International Symposium on Research in Attacks, Intrusions and Defenses, pp. 273-294. Springer, Cham, 2018.

[i.23] Wang, Bolun, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng and Ben Y. Zhao: "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks", In 2019 IEEE Symposium on Security and Privacy (SP), pp. 707-723.

[i.24] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du and Dawn Song. 2019: "Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in AI systems", arXiv preprint arXiv:1908.01763v2 (2019).

[i.25] Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith C.Ranasinghe and Surya Nepal: "STRIP: A Defence Against Trojan Attacks on Deep Neural Networks", 2019 Annual Computer Security Applications Conference (ACSAC '19).

[i.26] Chou Edward, Florian Tramèr, Giancarlo Pellegrino: "sentiNet: Detecting Localized Universal Attack Against Deep Learning Systems", The 3rd Deep Learning and Security Workshop (2020).

[i.27] Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan and Sudipta Chattopadhyay: "Model Agnostic Defence against Backdoor Attacks in Machine Learning", arXiv preprint arXiv:1908.02203v2 (2019).

[i.28] Bao Gia Doan, Ehsan Abbasnejad and Damith Ranasinghe: "Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems", The 36th Annual Computer Security Applications Conference (ACSAC 2020).

[i.29] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh and Dhruv Batra: "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", International Conference on Computer Vision (ICCV'17).

[i.30] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter and Bo Li: "Detecting AI Trojans Using Meta Neural Analysis", IEEE S&P 2021.

[i.31] Huili Chen, Cheng Fu, Jishen Zhao and Farinaz Koushanfar: "DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks", IJCAI 2019: 4658-4664.

[i.32] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash and Heiko Hoffmann: "Universal Litmus Patterns: Revealing Backdoor Attacks in CNNs", CVPR 2020.

[i.33] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru and Fabio Roli: "Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks", USENIX Security Symposium 2019.

[i.34] Yinpeng Dong, Tianyu Pang, Hang Su and Jun Zhu: "Evading Defenses to Transferable Adversarial Examples by Translation-Invariant Attacks", CVPR 2019.

[i.35] Carlini and Wagner: "Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods", the 10th ACM Workshop on Artificial Intelligence and Security 2017.

[i.36] Anish Athalye, Nicholas Carlini, David Wagner: "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples", ICML 2018.

[i.37] Florian Tramer, Nicholas Carlini, Wieland Brendel and Aleksander Madry: "On Adaptive Attacks to Adversarial Example Defenses", NIPS 2020.

[i.38] Florian Tramèr, Jens Behrmann, Nicholas Carlini, Nicolas Papernot, Jörn-Henrik Jacobsen: "Fundamental Tradeoffs between Invariance and Sensitivity to Adversarial Perturbations", ICML 2020.

[i.39] Gintare Karolina Dziugaite, Zoubin Ghahramani and Daniel M. Roy: "A study of the effect of JPG compression on adversarial images", International Society for Bayesian Analysis (ISBA 2016) World Meeting.

[i.40] Uri Shaham, James Garritano, Yutaro Yamada, Ethan Weinberger, Alex Cloninger, Xiuyuan Cheng, Kelly Stanton and Yuval Kluger: "Defending against Adversarial Images using Basis Functions Transformation", ArXiv 2018.

[i.41] Richard Shin and Dawn Song: "JPEG-resistant adversarial images", in Proc. Mach. Learn. Comput. Secur. Workshop, 2017.

[i.42] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E. Kounavis and Duen Horng Chau: "Shield: Fast, Practical Defense and Vaccination for Deep Learning using JPEG Compression", KDD 2018.

[i.43] Chuan Guo, Mayank Rana, Moustapha Cissé and Laurens van der Maaten: "Countering Adversarial Images using Input Transformations", ICLR 2018.

[i.44] Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang and Radha Poovendran: "Blocking Transferability of Adversarial Examples in Black-Box Learning Systems", ArXiv 2017.

[i.45] Ian J. Goodfellow, Jonathon Shlens and Christian Szegedy: "Explaining and Harnessing Adversarial Examples", ICLR 2015.

[i.46] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu: "Towards Deep Learning Models Resistant to Adversarial Attacks", ICLR 2018.

[i.47] Madry et al. (2019). RobustML.

[i.48] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin and Nicolas Usunier: "Parseval Networks: Improving robustness to adversarial examples", ICML 2017.

[i.49] Ross and Doshi-Velez: "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients", AAAI 2018.

[i.50] Cem Anil, James Lucas and Roger Grosse: "Sorting out Lipschitz function approximation", ICML 2019.

[i.51] Jeremy Cohen, Elan Rosenfeld and Zico Kolter: "Certified Adversarial Robustness via Randomized Smoothing", ICML 2019.

[i.52] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha and Ananthram Swami: "Distillation as a defense to adversarial perturbations against deep neural networks", 2016 IEEE Symposium S&P.

[i.53] Nicolas Carlini and David Wagner: "Towards evaluating the robustness of neural networks", 2017 IEEE symposium on Security and Privacy (S&P).

[i.54] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu and Luca Daniel: "CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks", AAAI 2019.

[i.55] Ching-Yun Ko, Zhaoyang Lyu, Tsui-Wei Weng, Luca Daniel, Ngai Wong and Dahua Lin: "POPQORN: Quantifying Robustness of Recurrent Neural Networks", ICML 2019.

[i.56] Aleksandar Bojchevski and Stephan Günnemann:"Certifiable Robustness to Graph Perturbations", NIPS 2019.

[i.57] G. Katz, C. Barrett, D. L. Dill, K. Julian and M. J. Kochenderfer: "Reluplex: An efficient SMT solver for verifying deep neural networks", in International Conference on Computer Aided Verification. Springer, 2017, pp. 97-117.

[i.58] X. Huang, M. Kwiatkowska, S. Wang and M. Wu: "Safety verification of deep neural networks", in International Conference on Computer Aided Verification. Springer, 2017, pp. 3-29.

[i.59] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori and A. Criminisi: "Measuring neural net robustness with constraints", in Advances in neural information processing systems, 2016, pp. 2613- 2621.

[i.60] V. Tjeng, K. Y. Xiaoand R. Tedrake: "Evaluating robustness of neural networks with mixed integer programming", in International Conference on Learning Representations (ICLR), 2019.

[i.61] S. Wang, K. Pei, J. Whitehouse, J. Yang and S. Jana: "Efficient formal safety analysis of neural networks", in Advances in Neural Information Processing Systems, 2018, pp. 6367-6377.

[i.62] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri and Martin Vechev: "AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation", IEEE S&P 2018.

[i.63] G. Singh, T. Gehr, M. Püschel and M. Vechev: "An Abstract Domain for Certifying Neural Networks", Proceedings of the ACM on Programming Languages, vol. 3, no. POPL, p. 41, 2019.

[i.64] Yizhak Yisrael Elboher, Justin Gottschlich and Guy Katz: "An Abstraction-Based Framework for Neural Network Verification", International Conference on Computer Aided Verification 2020.

[i.65] Guy Katz, Derek A. Huang Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer and Clark Barrett: "The Marabou Framework for Verification and Analysis of Deep Neural Networks", International Conference on Computer Aided Verification 2019.

[i.66] ERAN: Neural Network Verification Framework.

[i.67] Lily Weng, Pin-Yu Chen, Lam Nguyen, Mark Squillante, Akhilan Boopathy, Ivan Oseledets and Luca Daniel: "PROVEN: Verifying Robustness of Neural Networks with a Probabilistic Approach", ICML 2019.

[i.68] Linyi Li, Xiangyu Qi, Tao Xie, and Bo Li: "SoK: Certified Robustness for Deep Neural Networks", ArXiv, 2020.

[i.69] Shixin Tian, Guolei Yang and Ying Cai: "Detecting Adversarial Examples Through Image Transformation", AAAI 2018.

[i.70] Weilin Xu, David Evans and Yanjun Qi: "Feature squeezing: Detecting adversarial examples in Deep Neural Networks", Network and Distributed Systems Security Symposium 2018.

[i.71] Bo Huang, Yi Wang and Wei Wang: "Model-Agnostic Adversarial Detection by Random Perturbations", International Joint Conference on Artificial Intelligence (IJCAI-19).

[i.72] Xin Li and Fuxin Li: "Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics", ICCV 2017.

[i.73] Kevin Roth, Yannic Kilcher and Thomas Hofmann: "The Odds are Odd: A Statistical Test for Detecting Adversarial Example", ICML 2019.

[i.74] Dongyu Meng and Hao Chen: "MagNet: A Two Pronged Defense against adversarial examples", ACM Conference on Computer and Communications Security (CCS) 2017.

[i.75] Nicholas Carlini, David Wagner: "MagNet and "Efficient Defenses Against Adversarial Attacks" are Not Robust to Adversarial Examples", arXiv 2017.

[i.76] Faiq Khalid, Hassan Ali, Hammad Tariq, Muhammad Abdullah Hanif, Semeen Rehman, Rehan Ahmed and Muhammad Shafique: "QuSecNets: Quantization-based Defense Mechanism for Securing Deep Neural Network against Adversarial Attacks", IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS) 2019.

[i.77] Sanjay Kariyappa and Moinuddin K. Qureshi: "Improving adversarial robustness of ensembles with diversity training", ArXiv 2019.

[i.78] Thilo Strauss, Markus Hanselmann, Andrej Junginger, Holger Ulmer: "Ensemble Methods as a Defense to Adversarial Perturbations Against Deep Neural Networks", Canadian Conference on Artificial Intelligence 2020.

[i.79] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang and Ian Molloy: "Protecting Intellectual Property of Deep Neural Networks with Watermarking", ASIACCS'18.

[i.80] Wang, Tianhao and Florian Kerschbaum: "Attacks on digital watermarks for deep neural networks", in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2622-2626. IEEE, 2019.

[i.81] Takemura, Tatsuya, Naoto Yanai and Toru Fujiwara: "Model Extraction Attacks against Recurrent Neural Networks", arXiv preprint arXiv:2002.00123 (2020).

[i.82] Manish Kesarwani, Bhaskar Mukhoty, Vijay Arya and Sameep Mehta: "Model Extraction Warning in MLaaS Paradigm", ACSAC 2018.

[i.83] Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang and Jie Shi: "BDPL: A Boundary Differentially Private Layer Against Machine Learning Model Extraction Attacks", ESORICS 2019.

[i.84] Mika Juuti, Sebastian Szyller, Samuel Marchal and N. Asokan: "PRADA: Protecting Against DNN Model Stealing Attacks", 2019 IEEE European Symposium on Security and Privacy (EuroS&P).

[i.85] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter and Thomas Ristenpart: "Stealing Machine Learning Models via Prediction APIs", Usenix Security 2016.

[i.86] Liu, Yuntao, Dana Dachman-Soled and Ankur Srivastava: "Mitigating reverse engineering attacks on deep neural networks", in 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 657-662.

[i.87] Lukas, Nils, Yuxuan Zhang and Florian Kerschbaum: "Deep Neural Network Fingerprinting by Conferrable Adversarial Examples", arXiv preprint arXiv:1912.00888v3, (2020).

[i.88] Cao, Xiaoyu, Jinyuan Jia and Neil Zhenqiang Gong: "IPGuard: Protecting the Intellectual Property of Deep Neural Networks via Fingerprinting the Classification Boundary", ACM ASIA Conference on Computer and Communications Security (ASIACCS), 2021.

[i.89] Nicolas Papernot, Martin Abadi, Ulfar Erlingsson, Ian Goodfellow, Kunal Talwar: "Semi-supervised knowledge transfer for deep learning from private training data", ICLR 2017.

[i.90] Max Friedrich, Arne Köhn, Gregor Wiedemann and Chris Biemann: "Adversarial learning of privacy preserving test representations for de-identification of medical records", Proceeding of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.

[i.91] Taihong Xiao, Yi-Hsuan Tsai, Kihyuk Sohn, Manmohan Chandraker and Ming-Hsuan Yang: "Adversarial Learning of Privacy-Preserving and Task-Oriented Representations", AAAI 2020.

[i.92] Cynthia Dwork and Aaron Roth: "The algorithmic foundations of differential privacy", Foundations and Trends in Theoretical Computer Science, 2014.

[i.93] Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar and Li Zhang: "Deep learning with differential privacy", Proceedings of the 2016 ACM CCS, 2016.

[i.94] Milad Nasr, Reza Shokri and Amir Houmansadr: "Machine learning with membership privacy using adversarial regularization", Proceedings of the 2018 ACM CCS, 2018.

[i.95] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang and Neil Zhenqiang Gong: "MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples", ACM CCS 2019.

[i.96] Ziqi Yang, Bin Shao, Bohan Xuan, Ee-Chien Chang and Fan Zhang: "Defending Model Inversion and Membership Inference Attacks via Prediction Purification", ArXiv, arXiv:2005.03915v2, 2020.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

adversarial examples: carefully crafted samples which mislead a model to give an incorrect prediction

conferrable adversarial examples: subclass of transferable adversarial examples that exclusively transfer with a target label from a source model to its surrogates

distributional shift: distribution of input data changes over time

inference attack: attacks launched from deployment stage

model-agnostic mitigation: mitigations which do not modify the addressed machine learning model

model enhancement mitigation: mitigations which modify the addressed machine learning model

training attack: attacks launched from development stage

transferable adversarial examples: adversarial examples which are crafted for one model but also fool a different model with a high probability

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AE	Adversarial Example
AI	Artificial Intelligence
API	Application Interface
BDP	Boundary Differential Privacy
BIM	Basic Iterative Method
CIFAR	Canadian Institute For Advanced Research
CNN	Convolutional Neural Network
CW	Carlini & Wagner (attacks)
DNN	Deep Neural Network
DP-SGD	Differential-Privacy Stochastic Gradient Descent
FGSM	Fast Gradient Sign Method
GNN	Graph Neural Network
IP	Intellectual Property
JPEG	Joint Photographic Experts Group
JSMA	Jacobian-based Saliency Map
KNHT	Keyed Non-parametric Hypothesis Tests
L-BFGS	Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (algorithm)
MNIST	Modified National Institute of Standards and Technology
MNTD	Meta Neural Trojan Detection
PATE	Private Aggregation of Teacher Ensemble
PCA	Principal Component Analysis
PGD	Project Gradient Descent
PRADA	Protecting Against DNN Model Stealing Attacks
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RONI	Reject On Negative Impact
SAI	Securing Artificial Intelligence
SAT	Satisfiability
SGD	Stochastic Gradient Descent
SMT	Satisfiability Modulo Theories
STRIP	STRong Intentional Perturbation
TRIM	Trimmed-based algorithm
ULP	Universal Litmus Pattern

4 Overview

4.1 Machine learning models workflow

Artificial intelligence has been driven by the rapid progress in deep learning and the wide applications of deep learning, such as image classification, object detection, speech recognition and language translation. Therefore, the present document focuses on deep learning and explores existing mitigations countermeasuring attacks on deep learning.

A machine learning model workflow is represented in Figure 1. The model life-cycle includes both development and deployment stages. The *training dataset* is the subset of domain data samples used to train the model, and it can be obtained from one or multiple data sources, represented in Figure 1 as *data supply chain*. A pretrained model can be used as input to create the target model. At development stage, via the training dataset, the model is trained. The trained model is then tested. Pursuant to ETSI GR SAI 004 [i.1], the testing step will include functional test and adversarial test. At deployment stage, the trained and tested model is deployed, i.e. becomes the *model in operation*. Given *inference input*, the *model in operation* delivers an *output*. In Figure 1, the dotted lines from the *model in operation* back to the *model under development* capture the model *updates* in online learning scenarios [i.2]. *Updates* can be pairs of inference input and user feedback, served as new training data to refine the model. *Updates* can also be locally-computed model parameter refinements. These multiple dotted lines between the *model under development* and the *model in operation* capture the federated learning scenarios, where a global model is distributed among several entities and entities provide model updates to refine the global model.

Attack types described in ETSI GR SAI 004 [i.1] are classified into two main categories, i.e. *training attacks* which occur during development stage and *inference attacks* which occur during deployment stage. Training attacks include poisoning attacks and backdoor attacks while inference attacks include evasion attacks and reverse engineering attacks. Model stealing attacks, which violate model confidentiality, and data extraction attacks, which exploit models' vulnerabilities, are two subtypes of reverse engineering attacks. Some state-of-the-art adversarial machine learning papers [i.3], [i.4], [i.5], [i.6], [i.7], [i.8] and reports [i.9], [i.10] provide more details of existing attacks against deep learning systems. Existing mitigations for poisoning attacks, backdoor attacks, evasion attacks, model stealing attacks and data extraction attacks are summarized and analysed in the present document.

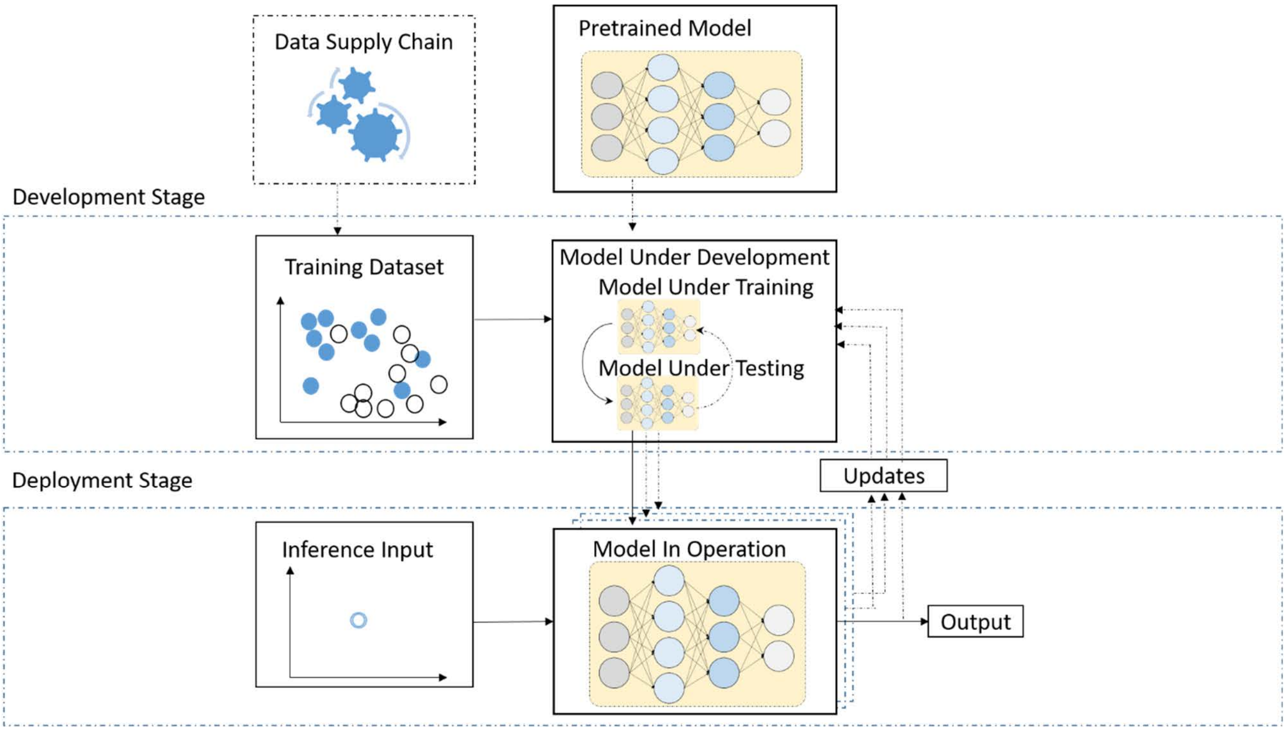

Figure 1: Machine learning model workflow

4.2 Mitigation strategy framework

Mitigations are summarized as approaches in a framework where a feasible strategy can be built against attacks under specific assumptions. In addition to being categorized by which attack they address, mitigations are further classified by whether the addressed model is modified when the mitigation is applied.

NOTE 1: The addressed model refers to the addressed machine learning model. The two mitigation categories are named as *model enhancement mitigations* and *model-agnostic mitigations*, where model enhancement mitigations modify the addressed model whereas model-agnostic mitigations do not. Model modification here emphasizes on model internal change, such as modifying parameters, reducing neurons, and changing optimizers and loss functions.

EXAMPLE 1: Adversarial training updates model parameters and thus is a model enhancement mitigation.

EXAMPLE 2: Ensemble techniques where a model is combined with other models as a whole are model-agnostic.

The intuition of this classification is to indicate whether the *model in operation* is modified when certain mitigations are applied. In some cases, updating deployed models is not possible, and model-enhancement mitigations are the only mitigations possible.

In most cases, mitigations in development stage usually modify the addressed model, while mitigations in deployment stage do not. However, in some application scenarios, the distinction is blurred. For instance, in online learning scenarios, pairs of inference input and user feedback are taken as new training data to refine the model. Then model-enhancement mitigations are possible for the deployed model.

Mitigation approaches are summarized in Table 1. They can serve as measures to prevent, detect, respond to, resolve or control for attacks.

NOTE 2: It is an active research area that how mitigation approaches against different attacks interfere each other.

Table 1: Mitigation Strategy Framework

Attack Types		Model Enhancement Mitigation Approaches	Model-agnostic Mitigation Approaches
Training	Poisoning attack	Clause 5.2.2	Clause 5.2.3
		• Enhance data quality • Data sanitization • Block poisoning	• Output restoration
	Backdoor attack	Clause 5.3.2	Clause 5.3.3
		• Enhance data quality • Data sanitization • Trigger detection • Model restoration	• Trigger detection • Trigger deactivation • Backdoor detection
Inference	Evasion attack	Clause 6.2.2	Clause 6.2.3
		• Data preprocessing • Model hardening • Robustness evaluation	• AE detection • Input restoration • Output restoration
	Model stealing	Clause 6.3.2	Clause 6.3.3
		• IP management	• Limit the number of queries • Stealing detection • Output obfuscation • Fingerprinting
	Data extraction	Clause 6.4.2	Clause 6.4.3
		• Embed data privacy • Training with privacy	• Limit the number of queries • Obfuscated confidence scores

5 Mitigations against training attacks

5.1 Introduction

Mitigations against training attacks are summarized and analyzed in clause 5, i.e. mitigations to protect the machine learning model from poisoning attacks and backdoor attacks.

Poisoning attacks are those where attackers tamper with the learning process by injecting adversarial data samples or modifying some data samples in the training dataset such that the resulting trained model has degraded accuracy. The poisoned dataset can contain mislabeled or confused data samples. Backdoor attacks aim at embedding malicious functionalities into the model. Attackers firstly embed a backdoor (also called Trojan) into the addressed model during the training phase. At inference time, the resulting model behaves as normal for clean inputs. However, when attackers feed an input containing a special pattern, so-called trigger, to the model, the model misbehaves, i.e. outputs attacker-intended results. As mentioned in ETSI GR SAI 004 [i.1], the attacker-intended results can be a targeted-and-incorrect output (integrity violation), or an output revealing (part of) the training dataset (confidentiality violation).

Poisoning attacks and backdoor attacks are different in terms of attack purposes and technical approaches. Poisoning attacks compromise the model functionality while backdoor attacks aim at embedding backdoors and triggering them later on. Backdoor attacks can use poisoning as part of the attack, but there are alternative means.

Existing mitigations against poisoning attacks and backdoor attacks are summarized and analyzed in clauses 5.2 and 5.3. To analyze existing mitigations, three major metrics are taken, i.e. model accuracy, mitigation overhead and model robustness.

5.2 Mitigating poisoning attacks

5.2.1 Overview

Poisoning attacks aim to degrade model performance on a broad set of inputs and are generally considered attacks on availability as described in ETSI GR SAI 004 [i.1]. This classification is based on the fact that poisoning attacks have been observed in inherently adversarial settings (such as spam filters), where they usually try to increase misclassification to the point of making the system barely usable, thus hampering availability of its proper functionality.

Such attacks are particularly relevant in case the distribution of input data changes over time (distributional shift) and frequent model updates are used for keeping up with this change. In this case, procuring trustworthy data sets for training takes a lot of effort. Unlike situations without significant distributional shift, this is not a one-time, but a constant effort.

A common feature of environments that are susceptible to poisoning attacks is the intrinsically adversarial setting. Those environments face a quickly changing attack landscape, and are themselves attractive targets for attackers.

EXAMPLE 1: Spam filters, malware scanners, firewalls and intrusion detection systems.

If the input data distribution is stable and models are not frequently updated, poisoning attacks are easy to detect during the testing phase. This assumes that the test data set is sufficiently large and representative for the data distribution. In particular, for detecting more targeted poisoning attacks that only decrease performance on some subset of inputs, more targeted and fine-grained testing can be performed.

EXAMPLE 2: For classifiers, measuring model performance not only over the entire data set, but also per class.

Existing approaches against poisoning attacks are summarized in Figure 2.

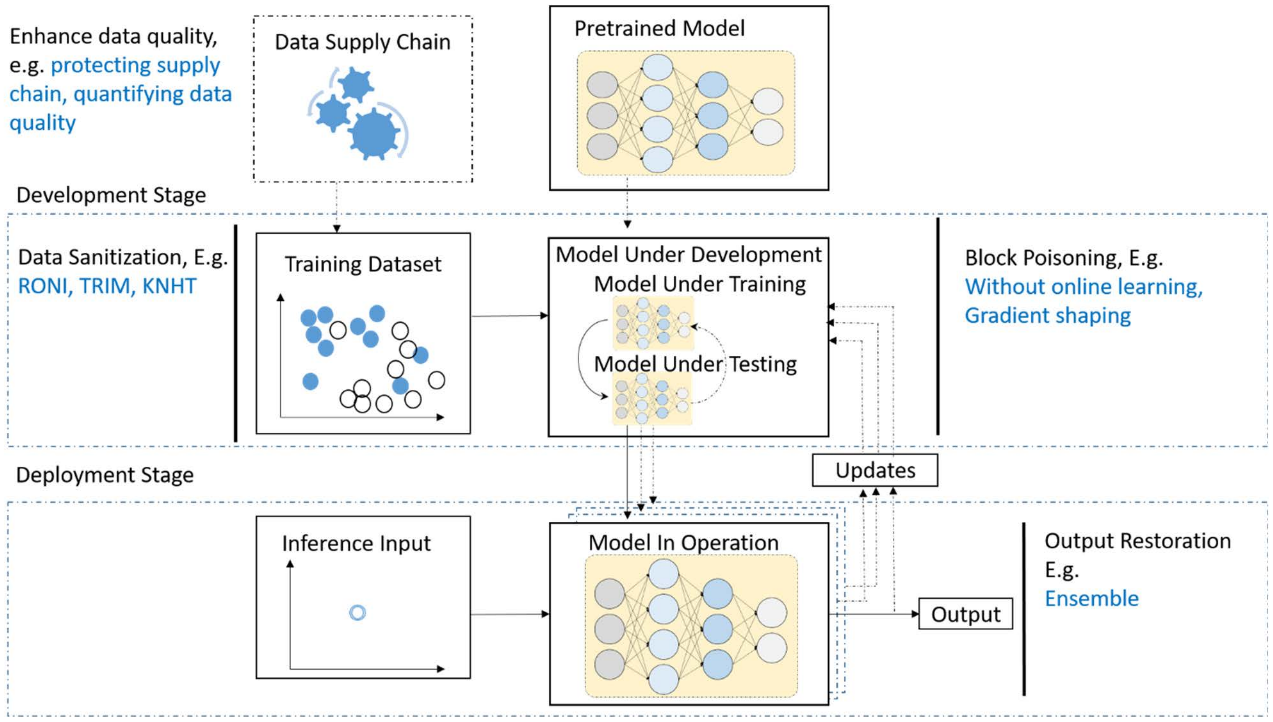
Enhance data quality, e.g. protecting supply chain, quantifying data quality
Data Supply Chain
Pretrained Model
Development Stage
Data Sanitization, E.g. RONI, TRIM, KNHT
Training Dataset
Model Under Development
Model Under Training
Model Under Testing
Block Poisoning, E.g. Without online learning, Gradient shaping
Deployment Stage
Updates
Inference Input
Model In Operation
Output
Output Restoration E.g. Ensemble

Figure 2: Approaches of mitigation techniques against poisoning attacks

5.2.2 Model enhancement mitigations against poisoning attacks

Enhance data quality is a model enhancement approach in data supply chain:

- One mitigation is to protect the data supply chain against manipulations to thwart attacks which tamper with or degrade the quality of training data. The risk for incorrect or manipulated data is much lower if data is sampled from a controlled environment.

NOTE: In the adversarial settings, protecting data supply chain takes substantial effort or can even be infeasible.

- A general method in pre-processing data as a means to minimize risks of data manipulations was proposed in [i.15]. It tries to estimate the quality of candidate training data and only uses data of sufficient quality for training the model. The authors consider the use-case of threat intelligence systems and use semi-supervised learning to judge the trustworthiness of data. The limitation of this approach consists in that it mostly considers increasing data quality with respect to benign problems (such as unintentional wrong labelling) and does not address targeted and optimized attacks.

Data sanitisation is a technique which aims to detect individual data points that have a negative impact on the model's performance and to exclude them from the final training dataset. Many variants of this approach have been explored:

- Reject on negative impact (RONI), [i.11]): RONI was the first proposal and identifies outliers by training the model with and without each point and comparing the performance. Due to frequently retraining the addressed model, RONI's run-time overhead is significant, and its performance is also worse than later proposals.

- TRIM [i.12] iteratively estimates the model parameters and trains on the subset of best-fitting input points at the same time, until convergence is reached. Its results are much better than those of RONI. However, TRIM is devised for linear regression only and thus not applicable to (deep) neural networks.

- Provenance-based [i.13]: another extension of RONI uses (presumably correct) meta data about data provenance and first clusters data accordingly. Due to the additional information used, this approach achieves better results than RONI and is more efficient by a certain factor (essentially, the average cluster size), since the model does not need to be retrained for each individual point but only the cluster centroids. The intuition of [i.13] is that for data points of common provenance the probability of being poisoned is strongly correlated. Applying this technique assumes the existence and correctness of information on data provenance.

- Keyed Non-parametric Hypothesis Tests (KNHT) [i.17] assumes a set of clean training data which describe intended data distribution and inspect newly-collected ones. The rationale is to compare the two set of data and if their similarity is insufficient, the newly collected ones are further inspected. KNHT does the data distribution comparison after mapping the two data distributions into another space via a set of functions with secret keys.

With the focus on the addressed model, **blocking poisoning** can be performed by blocking online learning and also during model training process. One example of blocking poisoning during model training process is as follows:

- Gradient shaping [i.14] starts from the new insight that poisoning (and backdoor) attacks exhibit larger gradients with differing orientations for poisoned data as compared to clean data. Hence, gradient shaping aims to prevent these properties in the gradients from occurring during model training (e.g. by cropping large values). The gradient shaping technique does not use additional information about the training dataset, and its run-time overhead is negligible. The results presented in [i.14] look quite promising for several attacks, but like other approaches, the strategy cannot thwart strong state-of-the-art attacks.

5.2.3 Model-agnostic mitigations against poisoning attacks

At the deployment stage, different models can also be combined as a final model to **restore output**. One example method is bagging ensembles [i.16], where the data sets used for training are assumed not identical, since the attack itself is carried out through the training dataset. In the most threatened applications, this condition is again hard to satisfy, since the availability of a sufficient amount of trustworthy data is the fundamental problem.

5.3 Mitigating backdoor attacks

5.3.1 Overview

Backdoor attacks involve two main steps: backdoor embedding and triggering. Thus, existing mitigations against backdoor attacks focus on detecting and removing backdoors and triggers based on different knowledge requirements, i.e. training datasets, models (full knowledge, partial knowledge or zero knowledge), and inference input. In development stage, backdoors and triggers can be detected and even removed by modifying the addressed model. In deployment stage, backdoor detection aims at revealing potential weaknesses of the addressed model and trigger detection goal is to prevent backdoors being triggered. In Figure 3, some model enhancement and model-agnostic mitigation approaches are shown with examples.

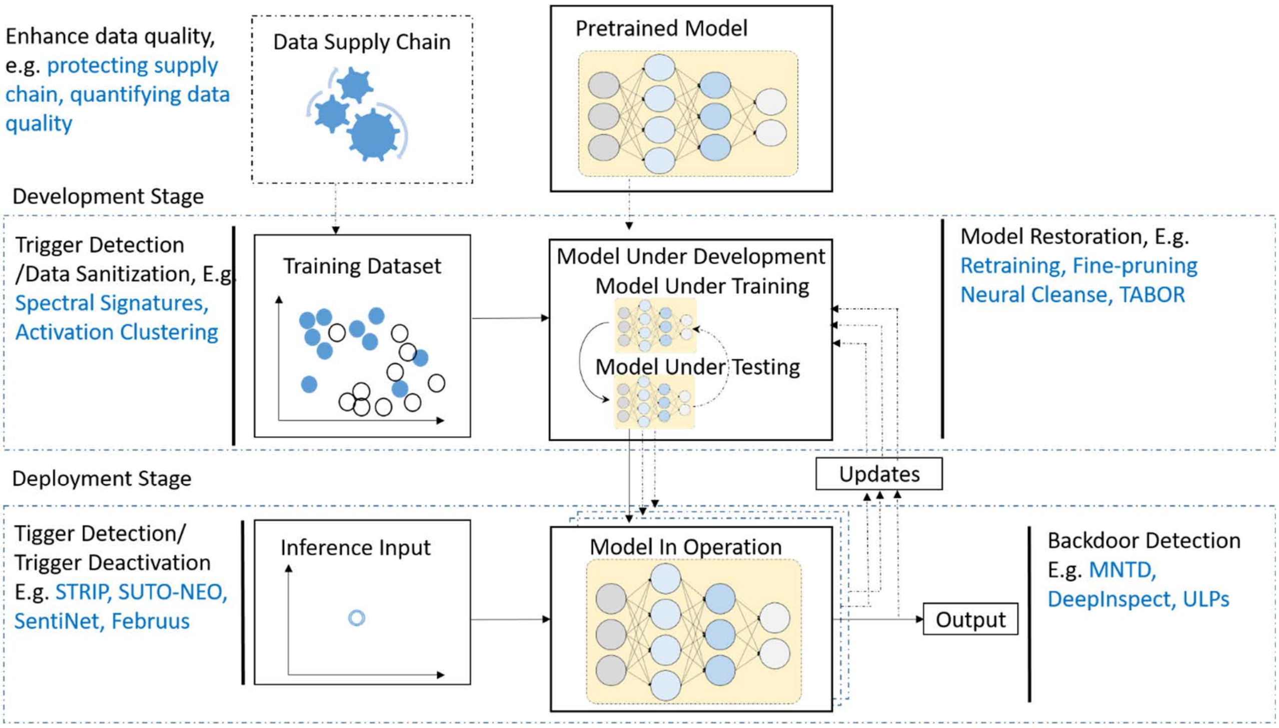
Enhance data quality, e.g. protecting supply chain, quantifying data quality
Data Supply Chain
Pretrained Model
Development Stage
Trigger Detection /Data Sanitization, E.g. Spectral Signatures, Activation Clustering
Training Dataset
Model Under Development
Model Under Training
Model Under Testing
Model Restoration, E.g. Retraining, Fine-pruning Neural Cleanse, TABOR
Deployment Stage
Updates
Tigger Detection/ Trigger Deactivation E.g. STRIP, SUTO-NEO, SentiNet, Februus
Inference Input
Model In Operation
Output
Backdoor Detection E.g. MNTD, DeepInspect, ULPs

Figure 3: Approaches of mitigation techniques against backdoor attacks

5.3.2 Model enhancement mitigations against backdoor attacks

Model enhancement methods aim at removing triggers and backdoors from training dataset and the addressed model at development stage.

With the **focus on training datasets**, enhancing data quality, data sanitization and trigger detection are three approaches. Poisoning attacks are a mean of implementing backdoor attacks; hence, enhancing data quality and data sanitization techniques against poisoning attacks are also effective to stop backdoor attacks.

Trigger detection from the training dataset is another approach to discover triggers and then remove them. Detection methods are based on some statistics difference observed between input with triggers and from benign inputs. However, those observations can be unique for some specific datasets. Here are some existing trigger detection methods with regard to training datasets:

- Spectral signature [i.18]: it is observed that spectral signatures can be used to identify and remove poisoned inputs, where spectral signatures are the spectrum of the covariance of a feature representation learned by the neural network.

- Activation clustering [i.19]: triggers can be detected from the neuron activation in the final hidden layer of a network. It is observed that input with triggers have different patterns of neuron activation from benign input. Hence, by using unsupervised clustering on neuron activation in the final hidden layer, input with triggers can be identified. The addressed model can then be retrained excluding triggers.

With the **focus on the addressed model**, **model restoration** is one approach. One potential way to do so is to detect and remove backdoors from the model. But model restoration can also be done without backdoor detection. Here are some existing model restoration methods:

- Retraining [i.20]: retraining the model with a small subset of clean training data is explored for correcting backdoor-embedded neural networks.
- Fine-pruning [i.22]: pruning a neural network eliminates some less important neurons [i.21]. It is a means of raising the success bar of backdoor attacks at some cost of accuracy. Pruning technique is effective when attackers are unaware of pruning protection. Fine-pruning retrains the model after pruning to address pruning-aware backdoor attacks. By applying fine-pruning, a backdoor-embedded neural network can be restored.
- Neural-Cleanse [i.23]: small input perturbations are reverse engineered from the model. Those perturbations trigger backdoor behavior in the model and then a backdoored class can be identified. The addressed model can then be patched by retraining with purified data. This process does backdoor un-learning. Neural cleanse method relies on an assumption that the trigger for the backdoor-infected class is smaller than the median size of the reverse engineered trigger for all classes. This assumption fails when triggers have various sizes.
- TABOR [i.24]: the backdoor detection task is transformed to a non-convex optimization problem with a newly-designed objective function to narrow down search space. Hence, input with triggers can be efficiently reverse-engineered. After triggers are found, by retraining the model with purified data, the model can be restored.

5.3.3 Model-agnostic mitigations against backdoor attacks

One model-agnostic method is to increase the quality of training data and to thwart attacks by protecting the data supply chain against manipulations. When data is sampled from a well-controlled environment, the training data quality is higher. Hence, the risks for incorrect or manipulated data are lower. Model-agnostic methods can also be developed for deployment stage. By analyzing given inference data samples and the addressed models, triggers and backdoors can be detected or deactivated at run-time at deployment stage.

NOTE: Most existing methods require a number of queries to the model to tell if one input contains a trigger.

With the **focus on inference data samples**, trigger detection and trigger deactivation are two approaches. After triggers are detected, they can be deactivated. But trigger deactivation can be done without trigger detection.

When a trigger is detected from a given inference data sample, the data sample can be treated separately. For instance, the data sample can be discarded or purified by noising the trigger or removing the trigger. Some existing methods of **trigger detection** are summarized below:

- STRIP [i.25]: a runtime trigger detection scheme is designed based on an assumption that an input with trigger is insensitive to perturbations. By applying a set of perturbations, a given inference input is then represented as a set of data samples. The classification entropy introduced by the set of data samples is a quantitative measure on how likely the given inference input contains a trigger. Inputs with low classification entropy even when applied strong perturbations are unlikely to be benign. The assumption is not universal. But this method does not require to know model parameters and can be performed at run-time.
- SUTO-NEO [i.27]: it is a trigger detection and prediction restoration scheme for image classification task. With an assumption that only one trigger exists and the trigger location is fixed on the image, an input image can be analyzed to see if a trigger exists. Given an input image, the potential trigger location can be searched by comparing classification results of the intact image and the image with a dominant color block at a random position. If the results diverse, the position likely has a trigger. To further confirm about it, the image block covered by the color block is implanted to other known-results training images. If those known-result training images with the patched image block have different prediction results from before, the position and the trigger are confirmed. To restore the classification result, the image with a trigger is patched with a block of the dominate color and then the corresponding predication can be restored.
- SentiNet [i.26]: by combining techniques of model interpretability and object detection, SentiNet is a mean of trigger detection at run-time against localized and universal backdoor attacks on image classifiers. It can detect more than one trigger with various sizes.

Trigger deactivation intends to stop backdoors being triggered. One way is to detect and remove triggers, but there are means of trigger deactivation without knowing the triggers. One existing method is described below:

- Februus [i.28]: researchers proposed a trigger deactivation method by input restoration technique against input-agnostic triggers. They analyzed which region of the input has the most impact on the classification result via visual explanation tool GradCAM [i.29] and then deactivate potential triggers. The most influential region is replaced by a neutralized color box and then the masked input is restored by generative adversarial networks. This method needs access to not only the model but also the training dataset.

With the **focus on the addressed deep learning model**, one approach is to detect whether the model contains a backdoor. Here are some existing **backdoor detection** methods:

- Trigger reverse engineering: as described earlier, neural cleanse method [i.23] and TABOR [i.24] both consist of two steps. The first step is to reverse engineer triggers from the addressed model and the second step is to retrain the model by purified data Although at deployment stage, the retraining step cannot be performed, the knowledge of triggers from the first step helps on detecting backdoored models.

- Meta Neural Trojan Detection (MNTD) [i.30]: a meta neural analysis framework is proposed for classifying benign models and backdoored models. A meta classifier can be built to classify models by representing a model as a sequence of predictions from fine-tuned queries. This method needs to build shadow models from clean datasets and self-generated backdoored datasets. But the method only requires query access to the addressed model with zero knowledge of model parameters.

- DeepInspect [i.31]: it detects backdoor without knowledge of model parameters and clean or training datasets. The method consists of three steps: model inversion to get a surrogate training dataset, trigger generation by conditional generative adversarial networks, and anomaly detection based on statistical hypothesis testing. DeepInspect detects backdoor in models with less prior knowledge at a cost of slightly lower detection rate.

- Universal Litmus Patterns (ULPs) [i.32]: similar to MNTD, Universal Litmus Patterns fine-tunes a set of optimized input images for determining whether a model contains backdoors. Although ULPs cannot deactivate triggers or backdoors, it provides an efficient and effective backdoor detection technique against single trigger-based backdoor attacks.

6 Mitigations against inference attacks

6.1 Introduction

Mitigations against inference attacks are summarized and analyzed in clause 6, i.e. evasion attacks, model stealing attacks, and data extraction attacks.

Evasion attacks refer to those attacks where adversaries carefully manipulate input samples to evade a deployed model at inference time.

EXAMPLE: For image classifiers, an attacker introduces crafted noise in a cat image such that the manipulated cat image is classified as a dog image by the deployed model.

Attack techniques vary according to the knowledge level of attackers on model parameters. Attackers with full knowledge are stronger than attackers with zero knowledge.

Model stealing attacks are attacks where attackers without knowledge of the addressed model try to steal model capabilities. There are two major attack approaches. One is to stealing model parameters by various means, such as side channel attacks or simply reading out model parameters from the underlying device. Another one is to generating a substitute model by exploiting the leaked information from a set of model input and output. In some occasions, model stealing attacks are about intellectual property issues. In other occasions, model stealing attacks are the first attack step for getting a surrogate model to launch other attacks afterwards, such as evasion attacks.

Data extraction attacks are attacks where attackers with ability of querying the addressed model attempt to infer the training dataset. These attacks violate data confidentiality, and are especially harmful for sensitive data and private data, such as facial recognition data, medical records and financial transactions. Data extraction attacks include *membership inference attacks* and *model inversion attacks*. Membership inference attacks are those where given a data sample, attackers can distinguish whether this data sample is in the training dataset of the addressed model. In model inversion attacks, given a model prediction, attackers find an input sample which is predicted by the addressed model to the given prediction.

Existing mitigations against evasion attacks, model stealing attacks, and data extraction attacks are summarized and analyzed in clauses 6.2, 6.3 and 6.4, respectively.

6.2 Mitigating evasion attacks

6.2.1 Overview

Evasion attacks involve two major steps: manipulating an input and feed the manipulated input to the addressed model. There are various known evasion attacks finding systematic ways to craft manipulated input samples. Those carefully crafted samples are called *adversarial examples* (AE).

Existing mitigations against evasion attacks focus on raising the bar of finding adversarial examples in development stage and detecting whether an input is an adversarial example in deployment stage. Some model enhancement mitigations and model-agnostic mitigations are shown with their approaches in Figure 4.

Transferability is a feature of adversarial examples. Some adversarial examples crafted for one model can fool a different model with a high probability. They are called *transferable adversarial examples*. The work in [i.33] demonstrates the transferability of adversarial examples among several machine learning models. A later work in [i.34] generates adversarial examples with better transferability.

The area of mitigations against evasion attacks has been rapidly evolving with new mitigations being proposed and broken at high speed. On the one hand, most mitigations empirically target very specific attack types, but are easily broken once the attacker can adapt to known mitigations, as has been repeatedly demonstrated [i.35], [i.36] and [i.37]. Such mitigations, even if they can be broken by adaptive attacks, can still be sufficient for certain use cases, since adaptive attacks take more efforts on the part of the attacker and are hence less likely to occur in practice. On the other hand, some mitigations are derived from theoretic analysis and formal reasoning in a certain threat model and hence remain effective against a wide range of attacks, including certain adaptive attacks. However, if the threat model does not capture the practical constraints of attacks sufficiently well, attackers can circumvent the threat model easily, and in this case these mitigations can be just as vulnerable as empirical ones. It is, for instance, well-known that the perturbation metrics used in threat models do not reflect human perception very well. In addition, there is a trade-off between the robustness and invariance to perturbations a model can have, which can likewise be exploited [i.38].

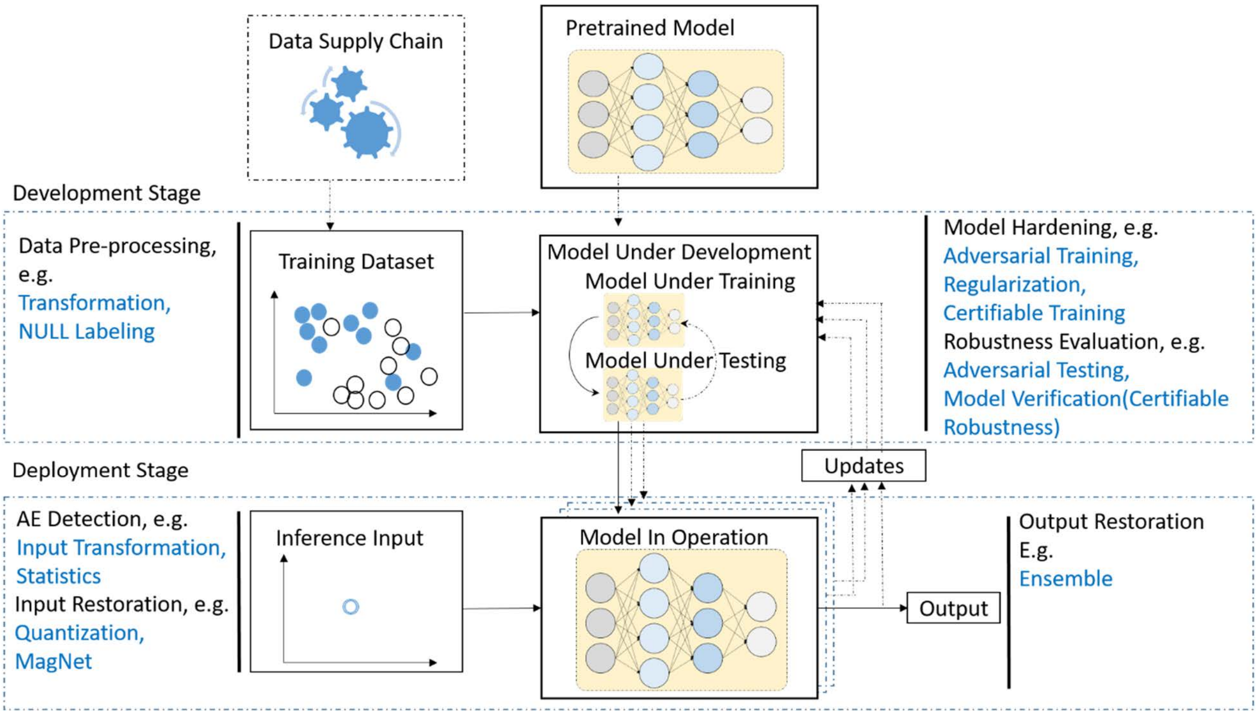

Figure 4: Approaches of mitigation techniques against evasion attacks

6.2.2 Model enhancement mitigations against evasion attacks

Model enhancement mitigations aim at introducing hardship of finding manipulation means. Different mitigation approaches are presented with existing mitigation examples where known limitations are also conveyed.

With **the focus on the training dataset**, **data preprocessing** is one approach. Data preprocessing techniques make crafting adversarial examples harder at the cost of computation overhead while keeping model accuracy. Some existing methods are:

- Transformation: to narrow down the attack surface of manipulating inputs, some data transformation techniques are applied. Adversarial perturbations are expected to be diminished through data transformation. For images, researchers took data compression as defense, for example, using Joint Photographic Experts Group (JPEG) for input data transformation [i.39]. Later on more data transformation techniques are proposed [i.40] as defense, including low-pass filtering, JPEG compression, Principal Components Analysis (PCA), soft-thresholding, patchwise PCA, low resolution wavelet approximation, against gradient-based attacks and concluded that JPEG compression tends to outperform others in most experimental cases. However, it has been [i.41] shown how to generate adversarial examples which survive JPEG compression. The compression defense [i.42] is advanced by combining JPEG compression and random compression level and showed that the new technique is effective against Carlini-Wagner's (CW) attack and DeepFool attack. Researchers [i.43] also studied data transformation as defense, including cropping-rescaling, bit-depth reduction, JPEG compression, total variance minimization, and image quilting on ImageNet and show that total variance minimization and image quilting are effective.

- NULL labelling [i.44]: for classifiers, instead of denoising adversarial inputs, NULL labelling technique lets the addressed model learn how likely an input is adversarial by adding a NULL label to the label space. As a result, adversarial examples are classified as NULL with a high probability while the model accuracy remains.

With **the focus on the addressed model**, one approach is **model hardening**. Model hardening techniques enhance model robustness at the cost of training overhead while keeping model accuracy remain. Model hardening techniques against particular attacks are in the arms race. Overtime, some of them can be defeated by adaptive attackers and hence have limited effectiveness. Some model hardening techniques are presented in the following with their strengths and limitations:

- Adversarial training is one of the most prominent techniques explored for mitigating evasion attacks. Adversarial training consists in training the model on a training data set augmented with correctly labelled adversarial examples, thus making sure the model will not be fooled by adversarial examples. The adversarial examples used for training can be computed using a variety of methods for mounting evasion attacks, such as the Fast Gradient Sign Method (FGSM) [i.45]. Some existing work [i.46] has shown that using the Projected Gradient Descent (PGD) method creates models with quite good robustness against evasion attacks, including adaptive attacks. This is probably due to the fact that PGD can be seen as a generalization of other attack methods, and can hence find the strongest adversarial examples for a given threat model with high probability. The comparatively good effectiveness of adversarial training using PGD compared to other defense methods is also confirmed by [i.47].
  Still, adversarial training only guarantees a certain level of robustness against evasion attacks, which is not sufficient for some applications. Adversarial training also increases the run-time for training by a certain factor that depends on the level of robustness provided.

- Regularization builds on the idea that preventing small input perturbations from changing model output can defend against adversarial examples. In mathematical terms this translates to bounding the Lipschitz constant [i.48] of the function implemented by the neural network or its gradient [i.49] by adapting the training procedure. In fact, there is a strong relation between these two approaches, since every differentiable function with bounded gradient is Lipschitz continuous (although the implied constant can be large, and the function in question is not everywhere differentiable).
  Regularization can certifiably improve robustness, but faces several limitations [i.50]. In many cases, the bounds on the Lipschitz constant or gradient, respectively, are not strictly enforced, but only encouraged by penalty terms. If the bounds are indeed enforced, this impacts the generalization performance of the network and thus its overall accuracy on complex tasks.

- Certifiable training: in the research line of robustness verification, certifiable training techniques attempt to train neural networks to improve the lower bound of robustness. Existing methods include designing specific regularizations, such as Lipschitz regularization, and randomized smoothing. Randomized smoothing technique [i.51] is derived from probabilistic robustness verification. For classifiers, a model is firstly trained by the training dataset augmented with Gaussian noises, where the output is a probability distribution over all classes. The final model is trained via the augmented dataset with labels indicating the class with the highest probability. Via randomized smoothing techniques, the derived model has probabilistic verifiable robustness.

- Gradient masking: to defend against gradient-based attacks, gradient masking is one approach. For example, shattered gradients introduce non-differentiable operations such that attackers cannot get correct gradients for crafting adversarial examples. Another example is stochastic gradients which introduce randomness on gradients. However, the results of [i.35] and [i.36] pointed out that gradient masking approach can be overcame effectively. To recover the gradient signals from shattered gradients, backward pass differentiable approximation techniques are proposed to approximate gradients [i.36]. To mitigate the uncertainty of stochastic gradients, expectation over transformation techniques are proposed to estimate gradients [i.35]. Hence, gradient masking approach is much less effective against gradient-based attacks.

- Distillation network [i.52]: it is also called defensive distillation. Distillation network is proposed as robust training technique for classifiers against adversarial examples. To train the distilled network, a teacher model is firstly trained on a training dataset. The distilled network is then trained on the same input and the corresponding output probabilities of the teacher network. The output probabilities are also called soft labels while the original labels of the training dataset are hard labels. Distillation network has been shown less effective against adaptive attackers [i.53].

In addition to model hardening techniques, **evaluating model robustness** is an essential primitive to quantify how robust a model is. Evaluation itself does not modify the addressed model. Coupling robustness evaluation and model hardening enables a loop of model robustness enhancement. Here are some existing methods of evaluating model robustness:

- Adversarial testing is closely related to adversarial training described earlier. Adversarial testing consists of testing the robustness of a trained model with respect to adversarial examples. These attacks can be created using a variety of methods, in particular projected gradient descent. Adversarial testing allows quantifying the robustness of a model to the considered attacks. If the testing results show that the robustness of the model is low, the model can be hardened more thoroughly, using adversarial training.

- Model verification for certifiable robustness: Certifiable robustness is a conceptual theoretic description capturing model robustness against evasion attacks. An adversarial example is a crafted input close enough to an intact input where they are somewhat indistinguishable to human, yet the addressed model gives them different output. Figure 5 gives a conceptual illustration, where x is an input, x' is an adversarial example, y and y' are model output for x and x' respectively. The distance between x and x' is evaluated by certain metrics, such as L_p distance. The addressed model is the function mapping input space to output space. Certifiable robustness attempts to certify a lower bound of the perturbation range δ where no adversarial example exists. Existing examples include certifiable robustness for CNN [i.54], RNN [i.55] and GNN [i.56].

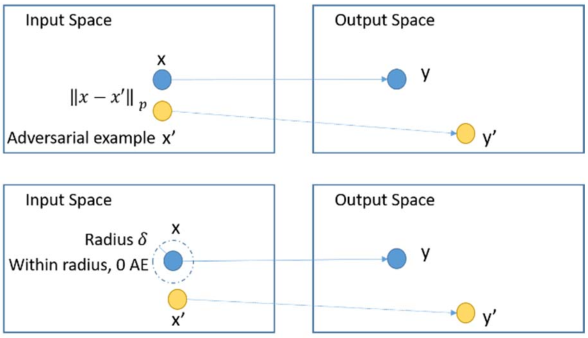

Figure 5: Adversarial examples and certifiable robustness

Model verification methods can be used to verify model robustness against certain attacks, such as Lp attacks where adversarial perturbations are limited within a range defined by Lp norm. Deterministic verification approaches are guaranteed to identify non-robustness when the given input is non-robust against the pre-described attacks. Probabilistic verification approaches identify non-robustness with certain probability. The verification method is called complete if the verification method claiming non-robustness imply that the addressed model is non-robust. Otherwise, the verification method is incomplete:

- For deterministic complete verification methods, Satisfiability (SAT) solvers are one example. Given neural network constraints and property constrains, SAT solver either output a counter-example or imply the absence of counter-example. Satisfiability Modulo Theories (SMT), which is a variant of SAT, can verify robustness of deep neural networks when property constraints represent model robustness [i.57] and[i.58]. However, SAT-based verification can only verify small-size neural networks. Scalability is still an active research area of model verification.

- For deterministic incomplete verification methods, linear programming based verification is an example. Linear programming-based verification technique represents neural network constraints as linear constraints and property constraints as objective functions, where constraints are all linear. Then linear programming solvers either find a counter-example or imply the absence of counter-example. Again, linear programming solvers serve as robustness verification when the objective is to determine the smallest noise that satisfies linear constraints of the network but results in misclassification [i.59] and [i.60]. When the addressed model is ReLU-based, ReLU is represented as two linear functions with the cost of larger problem size [i.61]. When the addressed model has non-linear activations, abstraction interpretation techniques [i.62], such as linear approximation [i.61] and [i.63], can be applied such that linear programming based verification can still work. It gains scalability. Via abstraction interpretation techniques, robustness of deep neural networks with practical sizes can be verifiable by incomplete verification. Abstraction refinement techniques [i.64] are proposed to incrementally refine the model abstraction to get a tighter verification result. Some reference tools for SMT-solvers [i.65] and abstraction interpretation based verification [i.66] are available.

- Existing probabilistic robustness verification methods are incomplete, yet they are scalable and efficient. PROVEN [i.67] is one example which provides probability certificates of neural network robustness. For some more information of robustness verification, here are some recommended references [i.68].

6.2.3 Model-agnostic mitigations against evasion attacks

Model-agnostic mitigations aim at detecting adversarial examples, restoring input samples or restoring model output.

With **the focus on inference data samples**, adversarial example detection and input restoration are two approaches. **Adversarial example detections** investigate input samples and tell if they are manipulated. Here are some existing methods:

- Input transformation: for image classifiers, image transformations, such as rotation and shifting, are proposed to detect adversarial examples by constructing a detector from the training dataset with image transformations [i.69]. Experimental results show the effectiveness by the detection rate on MNIST and CIFAR-10 against FGSM and CW evasion attacks. Another example of image transformation is feature squeezing [i.70], such as reducing the color bit depth of each pixel and spatial smoothing. One more example is adding random perturbation. An adversarial example is close to an input within a certain bounded range, but the addressed model gives different outputs. When a given input is indeed an adversarial example, adding some perturbation on the given input can push the output back to the original class with a non-negligible probability. Based on this rationale, researchers [i.71] proposed adding random perturbations on input and analysing the resulting output to detect adversarial examples. Their experimental results show the effectiveness on MNIST, CIFAR-10 and ImageNet against BIM, DeepFool and CW evasion attacks at the cost of generating a detector from the original training dataset.

- Statistics: to detect adversarial examples, some research seek for proper statistics. For example, for convolution neural networks, binary classifiers are constructed from the statistics of convolutional layer results from normal training dataset and adversarial examples [i.72]. Experimental results show the effectiveness on ILSVRC2012 (ImageNet-based) dataset with AlexNet and VGGNet network architectures against L-BFGS attacks at the cost of constructing a binary classifier. This method needs knowledge of training dataset. Another example of statistics method uses the expected perturbed log-odds over random perturbations to test if a given input *x* classified as class *y* is manipulated while the true class is *z* [i.73]. This method needs to access the results before the final layer at runtime. Experimental results show the effectiveness on CIFAR-10 and ImageNet against PGD attacks at the runtime cost of operating multiple inference for one input.

Input restoration techniques process input samples in order to reverse adversarial manipulation while preserving essential input features. Here are some existing methods:

- MagNet [i.74] is a framework of adversarial example detection and recovery. It integrates a set of detectors and a reformer. When an adversarial example has significant perturbation from the benign example, it can be detected by detectors. When the perturbation is less significant such that it cannot be detected by detectors, the reformer can restore the adversarial example to the benign example. One reformer example is to use autoencoder, which is an unsupervised learning algorithm. The autoencoder is trained by the training dataset without labels such that the output of autoencoder is close to the input. Using autoencoder as the reformer in MagNet does not change benign examples much but push adversarial examples to benign examples. Yet MagNet is less effective against the CW attacks [i.75] and has considerable computing overhead.

- Quantization methods [i.76] are proposed to diminish adversarial perturbations from input at inference time. The challenge is to find a proper quantization level such that perturbations are diminished while model accuracy remains. Trainable quantization method finds the suitable quantization level during training phase and applies it during inference phase. Experimental results show the effectiveness on MNIST and CIFAR-10 against FGSM, CW and JSMA attacks.

With **the focus on the addressed model**, **output restoration** is one approach. Ensemble techniques are one existing method to restore the model prediction:

- Ensemble: for classifiers, ensemble methods are proposed to mitigate adversarial perturbations. The assumption is that some adversarial examples misleading one classifier do not mislead other classifiers. Ensemble methods construct a set of classifiers to classify a new data input by a combination of their predictions. Researchers [i.77] proposed four ensemble constructions for model robustness, including random initial model parameters, similar but different network architectures, bagging, and adding Gaussian noise on training dataset. Their experimental results on MNIST and CIFAR-10 show that ensemble methods are effective against FGSM and BIM attacks. Another ensemble construction [i.78] is proposed by using different loss functions in the set of classifiers. Experimental results show that the ensemble method has better robustness against FGSM-based attacks at the cost of model accuracy on clean dataset. Some adversarial examples [i.33] can transfer among models, which are thus immune to ensemble methods.

6.3 Mitigating model stealing

6.3.1 Overview

There are two different types of model stealing attacks. The first one is to directly obtain model parameters by vulnerability of the system where the addressed model is located. Possible attack approaches include applying side channel attacks to extract model parameters and directly accessing model parameters in the memory of the deployed device. Another type of model stealing attack is to generate a substitute model with approximated capability by crafted queries to the victim model and corresponding model predictions.

Existing mitigations against model stealing attacks focus on securing model hyperparameters and parameters, detecting abnormal queries to the model in operation, obfuscating the inference output, and Intellectual Property (IP) management.

Protecting model hyperparameters and parameters against stealing is a traditional data confidentiality requirement. It can be achieved by adapting conventional data confidentiality techniques. Deploying secure hardware is one approach against side channel attacks and unauthorized memory access. Another approach against side channel attack is obfuscating side channel information [i.86]. Another approach against unauthorized access is applying cryptographic encryption schemes, that is, keeping model parameters in the encrypted form.

Existing mitigations against model stealing attacks via substitute models are summarized in Figure 6.

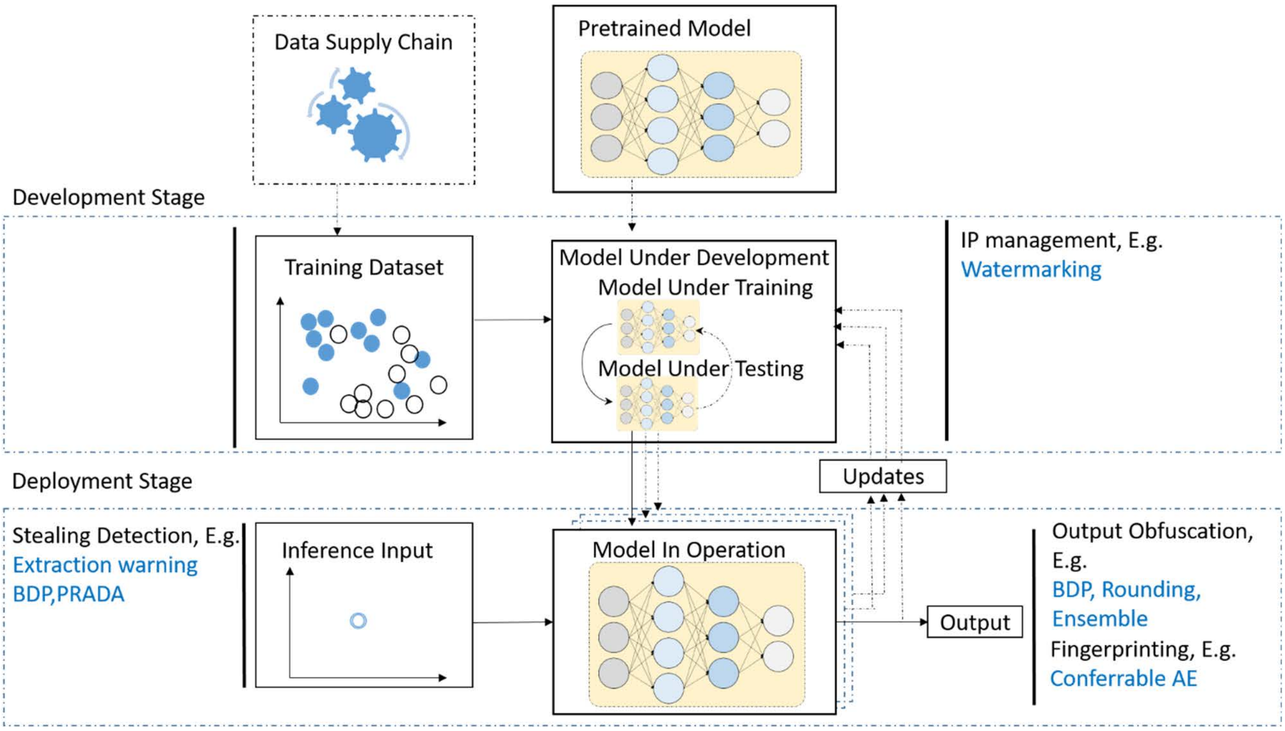

Figure 6: Approaches of mitigation techniques against model stealing attacks

6.3.2 Model enhancement mitigations against model stealing

Model enhancement mitigations against model stealing attacks have one approach consisting in **Intellectual Property (IP) protection**. To declare the ownership of the addressed model in case the model is extracted without authorization, special patterns or functionality can be embedded into the addressed model in the development stage:

- Watermarking [i.79]: to protect the copyright of DNN models, the technique embeds watermarks in the training dataset that enable to verify the ownership of deployed DNN services. The method creates the watermarks in the training dataset, then assigns pre-defined labels to the different watermarks, and finally trains the watermarks with pre-defined labels to DNNs. The DNNs automatically learn and memorize the patterns of embedded watermarks and pre-defined labels. By doing this, the protected model will produce the pre-defined prediction upon watermarked queries, and so will models obtained by the first type of model stealing attack. The watermarking technique turns backdoor attacks into a mitigation. In order to detect this model stealing attack, the detection method only requires normal usage of the suspicious model with zero knowledge of model parameters, i.e. only requires access to its API. The technique is usually applied over image data.

Different watermark types can be used, for example:

- Meaningful content embedded in original training data: Meaningful content such as images of text strings is added as watermarks to some of the images of the training dataset, so as direct reverse engineering to detect such watermarks is difficult. At ownership dispute, a watermarked query will be feed to the model of suspicion, and whether the model outputs the pre-defined label will be used as proof for verification.

- Independent training data with unrelated classes: Images from other classes which are irrelevant to the task of the protected DNN model are added as watermarks. In this way, while the embedded watermarks do not impact the original function of the model, the capability of recognizing the unrelated data is part of the functionality of the addressed model, which serves to verify the model ownership.

- Pre-specified noise: Specifically crafted noise can also be used as watermarks. The target model is trained to either memorize the specific noise created for watermarking or to generalize the noise so any noise that follows the same distribution of the pre-specified noise can be recognized.

Researchers have proved that watermarking is not always efficient to protect model IP [i.80] and watermarking does not prevent the model being extracted, which can go unnoticed if the adversary keeps the substitute model private [i.81].

6.3.3 Model-agnostic mitigations against model stealing

In addition to protecting model hyperparameters mentioned above, a generic model-agnostic mitigation is to **limit the number of queries** one can make with in a time period. Other existing model-agnostic mitigations against model stealing attacks majorly **detect suspicious queries** to the addressed model and then take different countermeasures, such as warning and **output obfuscation**. Here are some existing examples for reference:

- Extraction warning [i.82]: a cloud-based extraction monitor to inform model owners about the status of model extraction by both individual and colluding adversaries using a decision tree. The monitor observes the query response pairs of each adversary to the deployed decision tree model, and incrementally learns a local decision tree based on these tuples. The detection can be done at fixed time intervals or after the deployed model has answered certain number of queries. Two novel metrics are proposed to measure the model learning rate of adversaries. A first metric is based on entropy and measures the information gain of a decision tree with respect to a validation set provided by the model owner. The second metric is based on maintaining a compact model summary corresponding to each adversary with increasing number of queries. The compact model summary represents the boundaries of regions within the feature space that the adversary may have learnt for each class. The monitor assesses the coverage of summaries within their respective classes to compute the overall learning rate of an adversary. The objective is to detect if adversaries can, either individually or jointly, reconstruct a model that yields an accuracy beyond a given threshold from the queries obtained from the addressed model.

- Boundary differential privacy (ϵ-BDP) [i.83]: this method protects against model extraction attacks that use fine-tuned queries with differential property to infer the decision boundary of the addressed model. The proposed solution is to obfuscate the prediction responses near the decision boundary. Based on a perturbation algorithm called *boundary randomized response*, the ϵ -BDP method prevents adversaries from learning the decision boundary by a predefined precision regardless the amount of queries issued to the model prediction API. The technique is effective for both linear and non-linear models.

- Extraction query distribution identification [i.84]: Protecting Against DNN Model Stealing Attacks (PRADA) proposed a method that detects suspicious queries to the addressed DNN model by analysing the distribution of consecutive API queries and identifying whether the distribution deviates from benign behaviour, i.e. from a normal (Gaussian) distribution. PRADA collects stateful information of queries in addressed model prediction APIs and can detect all prior model extraction attacks with no false positives. This defense does not require any knowledge about the addressed model, nor about the training dataset, and it is agnostic to the distribution of the sample dataset used in the queries. However, the method can be circumvented by mimicking benign query distributions.

- Rounding confidences [i.85]: with the aim to limit the information provided by model query APIs, this technique proposes to round confidence scores of the target model to some fixed precision. For decision trees, rounding confidence scores increase the chance of node identifier collisions and thus reduces attacker's success rate. The defense has also shown its effectiveness against extraction attacks in decision trees, but not in some other models.

- Ensemble [i.85]: ensemble methods such as aggregation of predictions by a number of individual models, can be used to defend against model reverse engineering attack. In this research paper, the author found that ensemble methods' resilience to extraction attacks is higher, since attackers will only be able to obtain relatively coarse approximations of the target function.

Another mitigation approach is to prove the model ownership via **fingerprinting** when there is a suspicious model:

- Fingerprinting based on conferrable adversarial examples [i.87]: this defense addresses DNN-based classifiers and it extracts a set of inputs from the source model so that only surrogate models agree with the source model on the classification of such inputs, whereas (benign) reference models relative to the source model do not. These inputs called *conferrable adversarial examples* are a subclass of transferable adversarial examples that exclusively transfer with a target label from a source model to its surrogates. While this fingerprinting scheme is robust to almost all derivation and extraction attacks, strong adaptive attacks can still remove the fingerprint. To implement this protection, the defender has the full knowledge of the source model parameters, query access to the target model deployed by the attacker and a limited number of queries to verify whether the target model is a surrogate.

- Fingerprinting the classification boundary [i.88]: this technique uses gradient descent method to find fingerprinting data points near the classification boundary of the addressed DNN-based classifier. These data points together with their predicted labels are used as the fingerprint for the addressed classifier. The model owner can identify that a suspect classifier is potentially pirated from the addressed model by querying its prediction API to get the labels of the fingerprinting data points. If most of the suspect classifier's predictions match those of the fingerprint for the owner's model, then the IP is compromised with a high probability. Authors claim that unlike watermarking, the technique does not incur in any accuracy loss for the addressed classifier, since it does not tamper its training or fine-tuning process at all.

6.4 Mitigating data extraction

6.4.1 Overview

Data extraction attacks include two different types of attacks, model inversion attacks and membership inference attacks. *Model inversion attacks* on deep neural networks attempt to find a potential model input given a model output and model querying permissions. *Membership inference attacks* attempt to distinguish whether a data sample is in the training dataset or not given model querying permissions. Attack approaches are extracting leaked information from a collection of model input and output pairs and corresponding confidence levels of model output.

Existing mitigations against data extraction attacks focus on embedding data privacy, training with privacy and limiting leaked information. They are summarized in Figure 7.

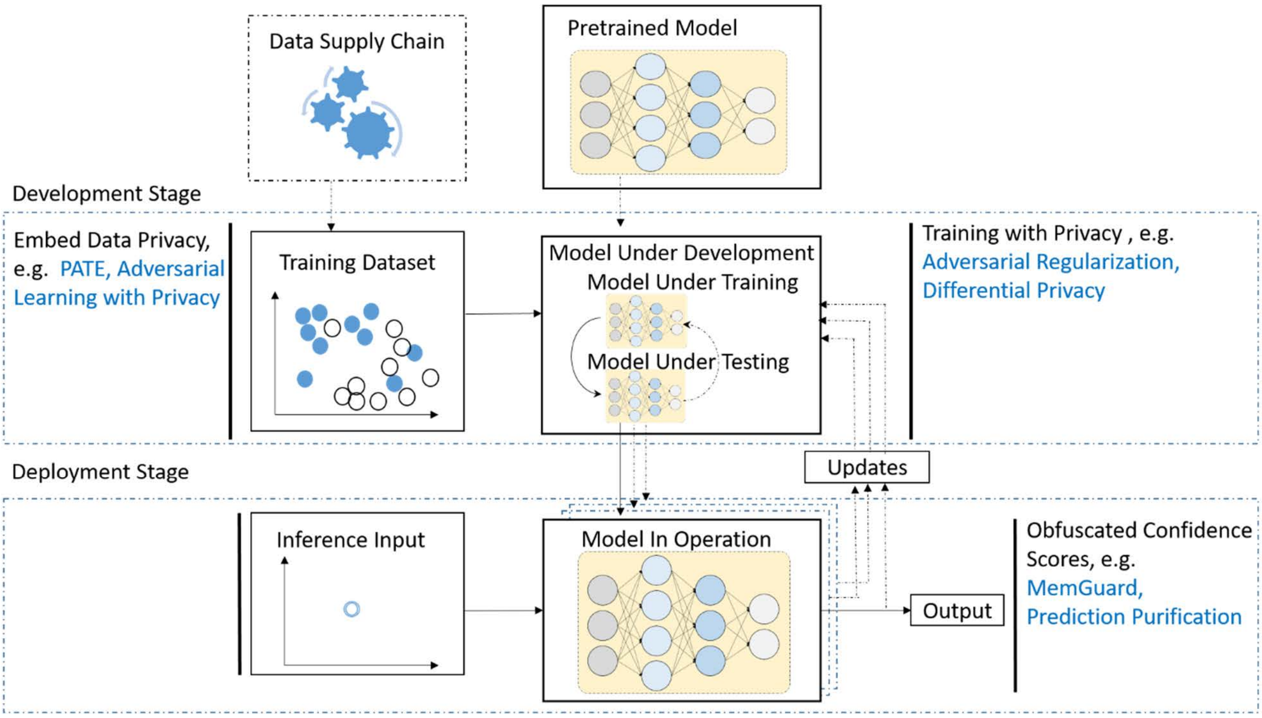
Data Supply Chain
Pretrained Model
Development Stage
Embed Data Privacy, e.g. PATE, Adversarial Learning with Privacy
Training Dataset
Model Under Development
Model Under Training
Model Under Testing
Training with Privacy , e.g. Adversarial Regularization, Differential Privacy
Deployment Stage
Updates
Inference Input
Model In Operation
Output
Obfuscated Confidence Scores, e.g. MemGuard, Prediction Purification

Figure 7: Approaches of mitigation techniques against data extraction attacks

6.4.2 Model enhancement mitigations against data extraction

To provide data privacy, two model enhancement approaches are to control the information provided by the training dataset for the training process and to train the addressed model with privacy guarantees.

With the focus on data, **embedding privacy** for training dataset is one approach. Here are some examples:

- Private Aggregation of Teacher Ensembles (PATE) [i.89] combines multiple models trained with disjoined datasets. These models are used as teachers for a student model, which learns to make prediction. The prediction is chosen by noisy voting among all of the teachers. These teacher models are not published and thus inaccessible except to the student model. PATE provides differential privacy for training data independent of the learning algorithm. Despite the provable robustness against membership inference attacks, PATE is hard to achieve with negligible utility loss.

- Adversarial learning with privacy: to provide data privacy, one approach is to train a data representation from the original training dataset for the model training at a later time. The trained data representation reveals less sensitive information of training data set while keeping model accuracy as much as possible. One example [i.90] is addressing medical records in text format. Medical records are converted to another representation with less identity information. The process is also called *de-identification*. The de-identification process is composed by three training stages, including training an initial representation, training a potential adversary, retraining the representation against such adversary. By using de-identified data representation, given two data representations, an adversary cannot distinguish if they are converted from the same input data. Another example [i.91] is addressing facial images. The representation generation of training data set is trained with two objectives of optimizing model accuracy and maximizing adversarial reconstruction loss. The adversarial reconstruction loss is defined by predictions of a discriminator and a perceptual distance. Hence, the obtained data representation of the training dataset reveals less information while keeping model accuracy.

With the focus on the addressed model, training with privacy is one approach. Here are some examples.

- DP-SGD [i.93]: differential privacy [i.92] is a framework for quantitatively measuring the privacy guarantees provided by an algorithm. The basic idea is to add randomness to algorithm's behaviours. Learning with differential privacy provides provable privacy guarantees and mitigating the risks of exposing sensitive information about training data. For a model trained with differential privacy, the behaviours of the trained model is less affected by any single training data. Hence, it is difficult to tell which data record belongs to the training dataset by observing the model's behaviour. By using the framework of differential privacy, Differentially Private Stochastic Gradient Descent (DP-SGD) technique is proposed to train a model with provable membership privacy. DP-SGD adds noises to the gradients used in Stochastic Gradient Descent (SGD), which is the core of almost all deep learning algorithms. The parameters of the trained model are iteratively updated using random noises and gradients. The privacy comes from the added noises. However, DP-SGD imposes a significant classification accuracy loss for protecting large models on high dimensional data.

- Adversarial regularization [i.94] mitigates membership inference attacks through adversarial training. The model's predictions on the members of its training dataset are indistinguishable from its predictions on the non-members of its training dataset from the same distribution. This requirement is captured as a min-max game optimization problem. The adversarial training algorithm is designed to train a model that minimizes both the prediction error and the maximum gain of the membership inference attack against it.

6.4.3 Model-agnostic mitigations against data extraction

To provide data privacy, one model-agnostic approach is to reveal less information from model prediction and to **restrict the number of queries** that can be made to the model. In some occasions, model prediction not only provides the final result but also other information, such as confidence scores for all possible predictions. **Obfuscating confidence scores** can reduce revealed information. Some examples are here:

- MemGuard [i.95] add noises to confidence scores to mitigate membership inference attacks. Assume attackers use a classifier to distinguish whether an input is in the training dataset. MemGuard crafts noises such that the manipulated confidence scores become adversarial examples to attacker's classifier. As a result, the attacker cannot successfully launch membership inference attack.

- Prediction purification [i.96] purifies the confidence score vectors predicted by the target classifier by reducing their dispersion. As a result, attackers get less information from purified confidence score vectors. Prediction purifications can be built to mitigate only membership inference attack, only model inversion attack and both attacks. Moreover, from experimental results, prediction purification against membership inference attack is effective against model inversion attack and vice versa. The generation of purifiers needs a subset of training dataset. The generated purifier is composed with the addressed classifier model to purify the confidence score vectors. Experimental results show prediction purification can reduce the membership inference accuracy and increase the model inversion error at the cost of little classification accuracy drop and little distortion to the confidence scores.

7 Conclusion

The present document summarizes existing and potential mitigation approaches against poisoning attacks, backdoor attacks, evasion attacks, model stealing attacks, and data extraction attacks for AI-based systems. Mitigation approaches are firstly summarized as model enhancement and model-agnostic, and then grouped by their rationales. Due to the rapid evolvement of attack technology for AI-based systems, existing mitigations can become less effective over time yet mitigation approaches and their rationales remain. In addition, most of the approaches presented stem from an academic setting and make certain assumptions which need to be taken into account when these approaches are applied in practice. The present document intends to serve as a securing AI technical survey reference for the planning, design, development, deployment, operation and maintenance of AI-based systems.

History

Document history		
V1.1.1	March 2021	Published as ETSI GR SAI 005
V1.2.1	July 2024	Publication