ETSI TR 104 032 V1.1.1 (2024-02)

**Securing Artificial Intelligence (SAI);
Traceability of AI Models**

Reference

DTR/SAI-004

Keywords

artificial intelligence, cyber security, digital right management, ML watermarking, trustworthy AI

Contents

Intellectual Property Rights

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 describes the role of traceability in the challenge of Securing AI and explores issues related to sharing and re-using models across tasks and industries. The scope includes threats, and their associated remediations where applicable, to ownership rights of AI creators as well as to verification of models origin. Mitigations can be non-AI-Specific (Digital Right Management applicable to AI) and AI-specific techniques (e.g. ML watermarking) from prevention and detection phases. They can be both model-agnostic and model enhancement techniques. The present document aligns terminology with existing ETSI ISG SAI documents and studies, and references/complements previously studied attacks and remediations (ETSI GR SAI 004 [i.2] and ETSI GR SAI 005 [i.3]). It also gathers industrial and academic feedback on traceability and ownership rights protection and model verification in the context of AI.

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] OpenAI's GPT-3 Language Model: A Technical Overview.

[i.2] ETSI GR SAI 004: "Securing Artificial Intelligence (SAI); Problem Statement".

[i.3] ETSI GR SAI 005: "Securing Artificial Intelligence (SAI); Mitigation Strategy Report".

[i.4] Artificial Intelligence Market Insights, 2020-2023.

[i.5] BankInfo Security®: "To Combat Rogue AI, Facebook Pitches 'Radioactive Data'".

[i.6] McKinney S. M. et al. (2020): "International evaluation of an AI system for breast cancer screening". Nature 577, pp. 89-94.

[i.7] Intellectual Property aspects of Machine Learning, NXP, 2022.

[i.8] Foss-Solbrekk K.: "Three routes to protecting AI systems and their algorithms under IP law: The good, the bad and the ugly". Journal of Intellectual Property Law & Practice 16.3 (2021), pp. 247-258.

[i.9] Hu X., Liang L., Li S., Deng L., Zuo P., Ji Y., Xie X., Ding Y., Liu C., Sherwood T. & Xie Y. 2020: "DeepSniffer: A DNN Model Extraction Framework Based on Learning Architectural Hints". In Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20). Association for Computing Machinery, New York, NY, USA, pp. 385-399.

[i.10] Xu Q., Arafin Md T. & Qu G. (2021): "Security of Neural Networks from Hardware Perspective: A Survey and Beyond". In Proceedings of the 26th Asia and South Pacific Design Automation Conference (ASPDAC '21). Association for Computing Machinery, New York, NY, USA, pp. 449-454.

[i.11] Chakraborty A., Mondal A. & Srivastava A. 2020: "Hardware-assisted intellectual property protection of deep learning models". In Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference (DAC '20). IEEE™ Press, Article 172, pp. 1-6.

[i.12] Goldstein B.F., Patil V.C., Ferreira V.C., Nery A.S., França F.M.G. & Kundu S.: "Preventing DNN Model IP Theft via Hardware Obfuscation". In IEEE™ Journal on Emerging and Selected Topics in Circuits and Systems, vol. 11, no. 2, pp. 267-277, June 2021.

[i.13] Li J., He Z., Rakin A., Fan D. & Chakrabarti C. (2021): "NeurObfuscator: A Full-stack Obfuscation Tool to Mitigate Neural Architecture Stealing", pp. 248-258. 10.1109/HOST49136.2021.9702279.

[i.14] Mahya Morid A., Alrahis L., Colucci A., Sinanoglu O. & Shafique M. (2022): "NeuroUnlock: Unlocking the Architecture of Obfuscated Deep Neural Networks".

[i.15] Tramer F., Zhang F. J., Reiter M. & Ristenpart T. (2016): "Stealing machine learning models via prediction apis. In 25th USENIX Security Symposium (USENIX Security 16), pp. 601-618.

[i.16] Kálmán S., Al-Afandi, J. & Horváth A. (2019): "MimosaNet: An Unrobust Neural Network Preventing Model Stealing".

[i.17] Uchida Y., Nagai Y., Sakazawa S. & Satoh S. I. (2017): "Embedding watermarks into deep neural networks". In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, pp. 269-277.

[i.18] Adi Y., Baum C., Cisse M., Pinkas B. & Keshet J. (2018): "Turning your weakness into a strength: Watermarking deep neural networks by backdooring". In 27th USENIX Security Symposium (USENIX Security 18), pp. 1615-1631.

[i.19] Zhang J., Gu Z., Jang J., Wu H., Stoecklin M. P., Huang H. & Molloy I. (2018): "Protecting intellectual property of deep neural networks with watermarking". In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pp. 159-172.

[i.20] Fan L., Ng K. W. & Chan C. S. (2019): "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attack. Advances in Neural Information Processing Systems (NIPS)".

[i.21] Chen H., Rouhani B. D., Fu C. Z. & Koushanfar F. (2019): "Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models". In Proceedings of the 2019 on International Conference on Multimedia Retrieval, pp. 105-113.

[i.22] Wang T. and Florian K. (2021) RIGA: "Covert and Robust White-Box Watermarking of Deep Neural Networks". Proceedings of the Web Conference 2021.

[i.23] Li Z., Hu C., Zhang Y. & Guo S. (2019): "How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN". In Proceedings of the 35th Annual Computer Security Applications Conference, pp. 126-137.

[i.24] Guo J. & Potkonjak M. (2018): "Watermarking deep neural networks for embedded systems". In 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1-8.

[i.25] Szyller S., Atli B., Marchal S. & Asokan N. (2019): "DAWN: Dynamic Adversarial Watermarking of Neural Networks".

[i.26] Jia H., Choquette-Choo C. A. & Papernot N. (2020): "Entangled Watermarks as a Defense against Model Extraction".

[i.27] Le Merrer E., Perez P. & Tredan G. (2019): "Adversarial frontier stitching for remote neural network watermarking. Neural Computing and Applications", pp. 1-12.

[i.28] Lukas Nils, Yuxuan Zhang and Florian Kerschbaum. (2019): "Deep neural network fingerprinting by conferrable adversarial examples".

[i.29] Cao Xiaoyu, Jinyuan Jia and Neil Zhenqiang Gong. "IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary". Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security.

[i.30] Namba R., & Sakuma J. (2019): "Robust watermarking of neural network with exponential weighting". In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, pp. 228-240.

[i.31] Chen Huili, Bita Darvish Rouhani and Farinaz Koushanfar (2019): "BlackMarks: Blackbox Multibit Watermarking for Deep Neural Networks".

[i.32] Sablayrolles A., Douze M., Schmid C. & Jegou H.: "Radioactive data: tracing through training". In Proceedings of the 37th International Conference on Machine Learning, PMLR 119:8326-8335, 2020.

[i.33] Yang Z., Dang H. & Chang E. C. (2019): "Effectiveness of Distillation Attack and Countermeasure on Neural Network Watermarking".

[i.34] Pan S. J. & Qiang Y. (2009): "A survey on transfer learning". IEEE Transactions on knowledge and data engineering, pp. 1345-1359.

[i.35] Hinton G., Oriol V. & Jeff D. (2015): "Distilling the knowledge in a neural network. NIPS Deep Learning and Representation Learning".

[i.36] Papernot N., Song S., Mironov I., Raghunathan A., Talwar K. & Erlingsson Ú. (2018): "Scalable private learning with pate". arXiv:1802.08908.

[i.37] Chen X., Wang W., Bender C., Ding Y., Jia R., Li B. & Song D. (2019): "REFIT: a Unified Watermark Removal Framework for Deep Learning Systems with Limited Data".

[i.38] Chen T., Goodfellow I. & Shlens J. (2016). Net2net: "Accelerating learning via knowledge transfer". In Proceedings of ICLR.

[i.39] Wei T., Wang C., Rui Y. & Chen C. W. (2016): "Network morphism. In International Conference on Machine Learning", pp. 564-572.

[i.40] Wang B., Yao Y., Shan S., Li H., Viswanath B., Zheng H., & Zhao B. (2019): "Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks". pp.707-723. 10.1109/SP.2019.00031.

[i.41] Aiken W., Kim H. & Woo S.S. (2020): "Neural Network Laundering: Removing Black-Box Backdoor Watermarks from Deep Neural Networks".

[i.42] Fredrikson M., Jha S. & Ristenpart T. (2015): "Model inversion attacks that exploit confidence information and basic countermeasures". In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1322-1333.

[i.43] Kapusta K., Thouvenot V., Bettan O., Beguinet H. & Senet H. (2021): "A Protocol for Secure Verification of Watermarks Embedded into Machine Learning Models". In Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '21).

[i.44] Kallas K. & Furon T. (2022): "ROSE: A RObust and SEcure DNN Watermarking". 2022 IEEE^{TM} International Workshop on Information Forensics and Security (WIFS), Shanghai, China, 2022, pp. 1-6.

[i.45] Moreau L., Clifford B., Freire J., Futrelle J., Gil Y., Groth P., Kwasnikowska N., Miles S., Missier P., Myers J., Plale B., Simmhan Y., Stephan E. & Van den Bussche J. (2010): "The open provenance model core specification (v1.1)". Future Generation Computer Systems, July 2010.

[i.46] W3C Working Group Note 30 April 2013: "PROV-Overview - An Overview of the PROV Family of Documents".

[i.47] ETSI GR SAI 002: "Securing Artificial Intelligence (SAI); Data Supply Chain Security".

[i.48] Gama J., Žliobaitė I., Bifet A. et al.: "A survey on concept drift adaptation. ACM computing surveys (CSUR)", 2014, 46(4): pp. 1-37.

[i.49] Krawczyk B. & Cano A.: "Online ensemble learning with abstaining classifiers for drifting and noisy data streams. Applied Soft Computing", 2018, 68: pp. 677-692.

[i.50] Kuncheva L. I.: "Change detection in streaming multivariate data using likelihood detectors". IEEE transactions on knowledge and data engineering, 2011, 25(5): pp. 1175-1180.

[i.51] Bifet A. & Gavalda R. (2007): "Learning from time-changing data with adaptive windowing?" Proceedings of the 2007 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2007: pp. 443-448.

[i.52] Zhang S., Pan C., Song L. et al. (2021): "Label-Assisted Memory Autoencoder for Unsupervised Out-of-Distribution Detection. Joint European Conference on Machine Learning and Knowledge Discovery in Databases". Springer, Cham, 2021: pp. 795-810.

[i.53] Wu, X., Hu, Z., Pei K, et al. (2021): "Methods for deep learning model failure detection and model adaption: A survey". IEEE^{TM} International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE^{TM}, 2021: 218-223.

[i.54] Mikołajczyk A. & Grochowski M. (2018): "Data augmentation for improving deep learning in image classification problem. International interdisciplinary PhD workshop (IIPhDW)". IEEE, pp. 117-122, 2018.

[i.55] Huang C., Hu Z., Huang X et al. (2021): "Statistical certification of acceptable robustness for neural networks". International Conference on Artificial Neural Networks. Springer, Cham, 2021: pp. 79-90.

[i.56] Shen Z., Cui P., Zhang T. et al. (2020): "Stable learning via sample reweighting". Proceedings of the AAAI Conference on Artificial Intelligence, vol 34(04), pp. 5692-5699.

[i.57] Jo J., Verma V., Bengio Y. (2018): "Modularity matters: Learning invariant relational reasoning tasks".

[i.58] Hummer W., Muthusamy V., Rausch T., Dube P., El Maghraoui K., Murthi A. & Oum P. (2019): "ModelOps: Cloud-Based Lifecycle Management for Reliable and Trusted AI". IEEE^{TM} International Conference on Cloud Engineering (IC2E), pp. 113-120.

[i.59] Vartak M., Subramanyam H., Lee W-E, Viswanathan S., Husnoo S., Madden S. & Zaharia M. (2016): "MODELDB: A System for Machine Learning Model Management. Workshop on Human-In-the-Loop Data Analytics (HILDA)", June 26 2016, San Francisco, CA, USA.

[i.60] Gundersen O. E. (2021): "The fundamental principles of reproducibility. Philosophical Transactions of the Royal Society A", 379(2197), 20200210.

[i.61] Pineau J., Vincent-Lamarre P., Sinha K., Larivière V., Beygelzimer A., d'Alché-Buc F., Fox E. and Larochelle H., 2021: "Improving reproducibility in machine learning research: a report from the NeurIPS 2019 reproducibility program". Journal of Machine Learning Research, 22.

[i.62] Chen B., Wen M., Shi Y., Lin D., Rajbahadur G. K. & Jiang Z. M. (2022, May): "Towards training reproducible deep learning models". In Proceedings of the 44th International Conference on Software Engineering (pp. 2202-2214).

[i.63] Arcuri A. and Briand L.: "A practical guide for using statistical tests to assess randomized algorithms in software engineering". In Proceedings of the 33rd International Conference on Software Engineering, New York, NY, USA, May 2011, pp. 1-10.

[i.64] Pham H.V., Qian S., Wang J., Lutellier T., Rosenthal J., Tan L., Yu Y. & Nagappan, N., 2020, December: "Problems and opportunities in training deep learning software systems: An analysis of variance". In Proceedings of the 35th IEEE/ACM international conference on automated software engineering (pp. 771-783).

[i.65] Elsken T., Metzen J.H. & Hutter F. (2019): "Neural Architecture Search: A Survey".

[i.66] Dehmer M. & Pickl S. (2015): "Network Complexity Measures: an Information-theoretic Approach".

[i.67] M. Fan, W. Wei, X. Xie, Y. Liu, X. Guan and T. Liu. (2021): "Can We Trust Your Explanations? Sanity Checks for Interpreters in Android Malware Analysis". In IEEE^{TM} Transactions on Information Forensics and Security, vol. 16, pp. 838-853.

[i.68] Yan S., Tao G., Liu X. et al. (2020): "Correlations between deep neural network model coverage criteria and model quality". In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, New York, NY, USA: Association for Computing Machinery, pp. 775-787.

[i.69] Lyu Y., Rajbahadur G. K., Lin D., Chen B. & Jiang Z. M. (2021): "Towards a consistent interpretation of AIOps models". ACM Transactions on Software Engineering and Methodology (TOSEM), 31(1), 1-38.

[i.70] Tulio Ribeiro M., Singh S. & Guestrin C.: "Why should I trust you?: Explaining the predictions of any classifier". Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM (2016).

[i.71] Tulio Ribeiro M., Singh S. & Guestrin C.:"Anchors: High-Precision Model-Agnostic Explanations" AAAI Conference on Artificial Intelligence (AAAI), 2018.

[i.72] Lundberg, S. M. & Su-In, L. (2017): "A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems".

[i.73] Rajbahadur G. K., Wang S., Oliva G., Kamei Y, Hassan A. E. (2021): "The impact of feature importance methods on the interpretation of defect classifiers". IEEE^{TM} Transactions on Software Engineering, pp. 10.1109/TSE.2021.3056941.

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void.

3.2 Symbols

Void.

3.3 Abbreviations

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

AI	Artificial Intelligence
AIA	Artificial Intelligence Agent
AIH	Artificial Intelligence Host
API	Application Programming Interface
AUC	Area Under the Curve
DevOps	Development Operations
DNN	Deep Neural Network(s)
DRM	Digital Right Management
FL	Federated Learning
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
IP	Intellectual Property
IPR	Intellectual Property Rights
ML	Machine Learning
MLaaS	Machine Learning as a Service
MLOps	Machine Learning Operations
NDA	Non Disclosure Agreement
OPM	Open Provenance Model
OS	Operating System
PKI	Public Key Infrastructure
TEE	Trusted Execution Environment

4 Introduction

4.1 The importance of traceability

During recent years, Machine Learning, and especially Deep Learning, put a new light on all kind of industries that solve complex problems by providing noteworthy capabilities in domains such as voice recognition, image comprehension, natural language processing, autonomous vehicles, and many more. However, its adoption is slowed down by the difficulty of training of an appropriate model that would fit the particular use case. To train a good ML model, it is indeed necessary to gather massive amount of relevant and ideally non-synthetic data, which can be a real burden as data can be expensive to produce, classified or protected by privacy regulations. Moreover, a sophisticated training requires also high computational resources, which represents a non-negligible cost. Finally, many engineers and researchers may be enlisted to design, implement and optimize an advanced ML model. All of the above-mentioned issues may discourage non AI-specialized actors from training its own models from scratch.

Two new challenges arise in such a context. First, new business models are being developed, aiming at monetizing data, pre-trained models, MLaaS services, or the training procedure. This fuels the development of web scrappers and model extraction attacks, as both data and models became attractive targets for attackers. Instead of spending time on acquiring a dataset or training a new model, attackers may now try to steal those already created by their competitors. It becomes thus crucial to ensure protection of ownership rights of the exposed models and data (besides, data theft may violate privacy regulations). Second, as training of a model is a fastidious and costly process, its reproducibility becomes a challenge on its own.

NOTE: The market for AI services is estimated to exceed 5,5 trillion dollars by 2027 [i.4].

Motivated by the growing complexity of modern AI and the development of new markets for ML models, the present document addresses two aspects of ML traceability: traceability that aims at providing protection against model misuse and traceability that improves AI trustworthiness. It focuses on techniques designed for Machine Learning and Deep Learning models, as those were the most studied in the context of AI traceability.

4.2 Traceability for ownership rights protection and fight against AI misuse

The ETSI Problem Statement [i.2] mentions reverse engineering and AI misuse as potential threats to AI systems. Indeed, an attacker unable to produce its own ML model may try to extract an existing one from the competitor's system in order to use it without paying or resell it. A similar situation may happen if an authorized user of a model decides to make an unauthorized usage of a model. While the ETSI Mitigation Strategy Threats [i.3] provides with an overview of some defence mechanisms against model extraction, the present document goes deeper into the problem. It presents both classical Digital Right Management (DRM) and novel traceability techniques, such as ML watermarking, that can be used to prevent or detect theft or misuse at any time of the ML lifecycle.

EXAMPLE 1: In 2020, a large social network operator introduced "radioactive data" to fight against datasets thefts. It was a reaction to the rise of big data startups, which are scrapping publicly available photographs from social networks [i.5].

EXAMPLE 2: Language models, such as the recently released "Chat GPT", can be used as universal writer's assistant that can help completing scholar assignments. Watermarking their outputs may help detect a misuse of the tool.

4.3 Traceability for trustworthy AI

Beside the challenge of protecting AI from theft or misuse, the present document tackles also the problem of traceability that aims at guaranteeing an easier reproduction of results. Such traceability can be understood as the monitoring of the whole lifecycle of a model, which includes not only the tracing of a model but also of its data and metadata as well as the details of the training process. It aims at accelerating the AI lifecycle and fostering collaboration and model reuse. In the context of secure AI, it should help detect attacks and enable easier integration of security mechanisms in the model development lifecycle.

EXAMPLE: In a paper from Nature [i.6], AI was shown to outperform radiologists in specific settings on the task of breast cancer screening. At the same time, it was pointed out that the absence of sufficiently documented methods and computer code underlying the study effectively undermined the scientific value of the research.

5 Traceability for ownership rights protection

5.1 Classical Digital Right Management

5.1.0 Introduction

This clause describes how classical DRM means - such as patents, trade secrets, and copyrights- can apply to AI and tries to identify where are their limitations. It is based on information contained in [i.7].

5.1.1 Classical DRM mechanisms explained

Intellectual Property Rights (IPR) are legal rights that protect non-tangible business assets against various types of misuse by third parties. Such misuse can be stopped by a legal injunction issued by a court, often combined with claims of financial damages and/or seizure of infringing products. However, each type of IPR has its own particular requirements and limitations. Below, copyrights, patents, database rights and trade secrets are presented.

NOTE: IP rights protection may vary depending on the country. For instance, many EU Member States do not even see trade secrets as part of the IP domain. Moreover, there is no consensus over the definition of an algorithm and the way it should be protected, e.g. algorithms may be protected in Italy but not in Germany [i.8].

Copyright is the most well-known type of IPR. A copyright is the right to forbid copying and dissemination of a protected work. Traditionally this right has been used much in the creative arts, e.g. for music, books and photographs. However, copyright applies just as much to business works such as software, manuals, whitepapers, company videos and so on.

Patents are the heavy lifters of the IPR world. When an innovation is protected by a patent, a patent owner may prevent others from making, using or selling any device incorporating that innovation. Unlike a copyright, a patent protects any independent re-creation. The patent holder can demand royalties or simply put an end to someone's commercial use of his innovation. The major drawback is the application process, which involves costly fees and a multi-year examination process with uncertain outcome. A complication with software is the case law on "software patents," which may be perceived negatively in some countries. Thus, it may be difficult to enforce a patent on an innovation that heavily draws on software or automation in all jurisdictions.

A relative newcomer in the IPR world is the database right. Introduced in Europe in the late 1990s, the database right protects a collection of information against copying and reuse. The main requirement to qualify for a database right is that substantial investment was made in the creation or maintenance of the data in the database. As with copyright, no formal registration or application is required. Examples of protected databases include online dictionaries, labelled image collections and source data for cartographical maps. In all cases, the data should be organized for search and browsing. Outside of the European Union, however, the database right is not recognized, which further complicates IPR. The U.S. has a long-standing legal tradition that collections of data are not protectable by IPR; only creative works can be protected under copyright.

The status of trade secrets in the IPR world differs around the world, but in general, misappropriation of well-protected information is actionable by law. In this instance, the owner of the information would be required to show how it applied adequate security measures against unauthorized access. A would-be trade secret thief could then counter by proving that the information was already available in the public domain.

Typically, companies guard their secrets by signing Non-Disclosure Agreements (NDAs) with customers or other third parties. Strict contractual obligations then prohibit copying or reuse, in some jurisdictions strengthened by contractual fines or other legal measures. NDA provisions may also be present in other agreements. However, someone who learns the confidential data from a legitimate purchase of a product is not bound by such provisions, even when using special techniques such as reverse engineering. This limits the strength of trade secret law.

5.1.2 Protection of the training set

Creating a good training set for a particular ML application can be a time-consuming and expensive effort and, although in a typical setting an infringer has no direct access to this training set, the ability to copy the set is easy if the person has access. This is where IP law comes in.

A training set would be protected with a database right, if the owner of the training set has its principal place of business in the European Union. However, such right would only be enforceable against an infringer in the same jurisdiction. Whether copyright can be claimed on an ML training set is a more difficult question. A training set is not created to be a piece of art. The typical intention is to ensure the data fits the use case. Creating a well-fitting set of data on a topic may not necessarily not be a creative activity under copyright law. One potential copyright claim are the data classification descriptors.

EXAMPLE 1: If categories are chosen through a creative process - "beautiful/ugly", "strong/weak", "big/small" - then the training set could be said to be protected by copyright through this creative labelling. A classification based on factual elements - "cat/dog", "traffic light/ streetlight/parking sign" - does not necessarily impart creativity and therefore possibly may not allow for copyright protection.

In some applications, training sets are generated by simulation or other artificial means. Arguably, these training sets could be copyright protected as the choice of how to simulate or generate could be seen as a creative choice. However, to date, this has not been challenged in court.

EXAMPLE 2: AI, and more particularly GANs, can be used to generated non-private training datasets from private data. Such datasets will preserve the characteristics of the original datasets without leaking sensitive information.

Companies will often consider their training sets to be carefully guarded secrets. Since a training set is not required to be shared for the ML model to be used, it seems straightforward. The best approaches are to both guard the training set from illicit copying and apply strong contractual restrictions to parties that have the training set.

5.1.3 Protection of the training parameters

The training set and model are only a part of the value of a good ML system. The parameters that steer the training algorithm may also have value: choosing the right training parameters takes time and effort from highly trained engineers.

For the set of training parameters that create the ML system, copyright protection is a sensible approach. If a data scientist determining these parameters uses creative efforts to select the right training parameters, the resulting set of parameters would likely be protected by copyright. But if the training parameters were found through exhaustive search (e.g. evaluating a number of options proposed in the literature) or algorithmic process, copyright may not be available. The same would apply to the model that is produced using those training parameters and a given training set.

A database right is least likely on the parameter set because one criterion for database rights is that it applies to a collection of individual elements that are systematically or methodically arranged. A parameter set is unlikely to fit that criterion.

5.1.4 Protection of the architecture

The architecture of the system is the underlying foundation for the ML system. Its design is a key aspect of the proper functioning of the system. After training, the architecture can be put in practice.

A system like this has two aspects: the graph defining the architecture and the software implementing it. The graph is protected under the same conditions as given for the protection of the model parameters. Patents would theoretically be available for innovative hardware aspects of it, but however most innovation in this area is purely software. The software that implements training and/or inference would typically be protected with copyright, as it is principally software designed using creative efforts.

5.1.5 Protection of the ML system

In theory, a computer system programmed with a well-chosen parameter set and trained on a specific training set could fall within the realm of patentable subject matter. However, current case law in Europe and the United States would require the system to be designed to perform real-world tasks such as steering a car or recognizing images from the real world. To date, it would be speculative to conclude a patent is obtainable on an ML system that operates in a more abstract manner, e.g. recognition and/or classification without a specific use case in the real world. The software of the ML system could be protected by copyright just like any other software.

A database right for the ML system is theoretically arguable: in a way the dataset is made searchable through the model and the software executing that model. However, this has never been decided in court or outlined in legal literature.

5.1.6 Protection of the model against copying

When an ML system is available without contractual or usage restrictions to the public, a unique way to copy its functionality becomes available. Essentially, the copyist has a dataset of unclassified items and submits each item to the ML system. Each answer is carefully recorded as the classification of the copyist's dataset. The obtained labelled dataset can then be used to train a model of similar quality. It has been shown that this works effectively, even if the dataset contains non-problem domain data and if the architecture and model parameters of the target and clone do not match. Under copyright or database law, it is unclear if this act is legal or not. The dataset from the original ML system is not copied; only its output is used, and then only to label a different dataset.

If the dataset classification is creative in itself, the copyist may infringe that copyright by reusing the labels. This could even apply if only the labels are copied and reused to classify a completely independent dataset. However, this has never been tested in court.

5.1.7 Comparison of classical DRM mechanisms in the ML context

Interest in IP rights is increasing to protect investments, from copyrights on training sets to patents on classification systems. Current IP law and practice is evolving, and case law is sparse. It is uncertain how legal protection for ML-system and ML-driven products will mature.

However, some general indications are already available and presented in Table 1. The contents of this table may be subject to change with evolving statutory and case law in various jurisdictions.

In short, trade secret laws are currently the most common way to approach AI IPR protection, as protection under copyrights or patents encounters difficulties. Extending patents to incorporate algorithms, algorithmic models and their bespoke datasets was identified as the most promising solution [i.8], as it would improve the transparency of the systems (and therefore, improve the understanding of AI-based decisions that can be crucial for some critical applications), but also would grant a better insight into which systems actually exist.

Table 1: Comparison of DRM mechanisms in the context of ML

	Intellectual Property Right (IPR)			
	Patent	**Copyright**	**Database right**	**Trade secret**
Protects	Technological innovation	Creative expression	Substantial investment in creation of the collection	Information is kept secret
Jurisdiction	Worldwide	Worldwide	Apply as long as the owner and infringer are in the EU	Worldwide
Protected item				
Architecture	Possible, but see software below	Unlikely for underlying graph, unless creativity in choices	No	Yes
Training set and Test set	Unlikely, but see software below	No, except creative labels or creatively handpicked dataset	Yes	Yes
Training parameters	Unlikely, but see software below	Unlikely, unless creativity choices	No	Yes
Model	Unlikely	Unlikely unless creativity in watermark, labels, parameters or architecture choices	Unlikely	Yes
Software implementing the ML functionality	Yes, as part of trained system with the model and only when aimed at real world tasks	Yes, but the functionality that it implements is excluded from protection	No	Yes

5.2 AI-specific prevention of model misuse

5.2.1 Full-knowledge exposure strategies

Model distribution and deployment present an attack surface. During transmission from the model provider to an edge device, the model can be vulnerable to classical man-in-the-middle attacks. At the edge device, models can be vulnerable to AI-specific attacks targeting model theft, such as hyperparameter extraction via side-channel attacks or model parameter theft via bus monitoring attacks. Thus, security mechanisms should be applied in order to protect the distribution of the model against model theft, as well as to prevent unauthorized use of the model after deployment.

Efforts on preventing neural networks architecture stealing during deployment have focused on enhancing hardware to eliminate information leakage, ex. by encrypting memory addresses, obfuscating GPU memory accesses, or introducing noise via fake memory traffic [i.9] and [i.10]. However, encrypting memory leads to a performance overhead and noisy memory traffic may not be sufficient to thwart side-channel attacks.

ML-specific hardware/software obfuscation can be leveraged as a lightweight level of protection against extraction attacks and unauthorized model usage. It can be applied during the model training [i.11] or as a postprocessing treatment [i.12]. Here are some examples of recent obfuscation techniques:

- A model owner can use a key-dependent backpropagation algorithm to train a DNN architecture which obfuscates the model's learned weight space. In more details, such training locks random neurons within the defined key. An authorized user will use a trusted hardware with the key embedded on-chip to run the obfuscated model [i.11]. The accuracy of the obfuscated model will significantly decrease for an authorized usage.

- Already trained models can be obfuscated by swapping convolutional filters based on a secret key [i.12]. Different strategies can be used for the swapping. They will balance between obfuscation level, output class distribution, and accuracy. Preserving output class distribution hides from the attacker the fact that the model is locked.

- The obfuscation can be performed during the scripting stage of a DNN, affecting the number of computations, latency, and the number of memory accesses, and hence modifying the execution trace [i.13] and [i.14].

NOTE 1: Model distribution may be also secured using traditional cryptographic mechanisms (ex. hardware root of trust combined with public-key infrastructure).

NOTE 2: Securing deployment may require AI-specific techniques. Indeed, advanced cryptography - such as Homomorphic Encryption or Secure Multi-Party Computation - may be used to encrypt a deployed model during use. However, this solution comes at the cost of performance decrease. TEE are a promising alternative, although they are not all supporting GPU usage and they can still be vulnerable to side-channel attacks.

5.2.2 Zero-knowledge exposure strategies

When exposing a model in a zero-knowledge setting, typical for the Machine Learning as a Service, the main preoccupation of the model owner in the context of ownership protection is to protect the model against extraction attacks, such as [i.15], that aim at creating a copy of the deployed model by using its predictions to train a new model. Protection against extraction attacks can be achieved by limiting the number of queries to the model or detecting extraction patterns in the queries addressed to the model. Another solution is to train the exposed model to be extremely sensitive to weight changes and therefore be resistant to stealing attacks [i.16].

In addition to theft prevention mechanisms, active watermarking can be implemented at the level of the model interface (see clause 5.3.3.1). This defence mechanism consists in marking the extracted model on the fly: answers to a small subset of the queries are modified on purpose in order to mark the behaviour of the stolen model and enable its identification in the future.

5.3 ML watermarking

5.3.0 Introduction to ML watermarking

Model watermarking is the main technique used for AI-specific detection of AI origin. It aims at enabling traceability by embedding a watermark into the model before sharing it. A model watermark is understood as an unusual change in the model (such as a change in its look [i.17] or behaviour [i.18] and [i.19]), inserted on purpose in the model by its creator or the legitimate owner, during or after the model training. Usually, multiple watermarks are embedded into a model, which are kept secret until the model verification. Model verification consists in model owner demonstrating their knowledge of one or more watermarks.

5.3.1 Verification setting and type

5.3.1.0 Introduction

Two categories of model watermarking can be distinguished, according to their verification settings. *Full-knowledge watermarks* require the access to the full model in order to be detectable. They are usually embedded into model's weights/architecture and thus verifiable only if model internals are accessible. *Zero-knowledge watermarks* can be verified even if only the model's API is exposed: they are changes in the model behaviour that can be verified by analysing model input-output pairs. *White-box* and *black-box* terminology could be also applied to describe these two watermark categories.

Watermarks can be verified in a *static* or *dynamic* way. The first verification type, occurring only in the full-knowledge setting, analyses the model look. The second verification type, analyses the model behaviour.

NOTE: Most often, full-knowledge watermarks are also static. However, exceptions are possible. A scheme presented in [i.20] identifies a model by analysing its performance with and without some special layers that perform the role of the watermarks. Although requiring a full-knowledge setting, this scheme would be categorized as rather dynamic.

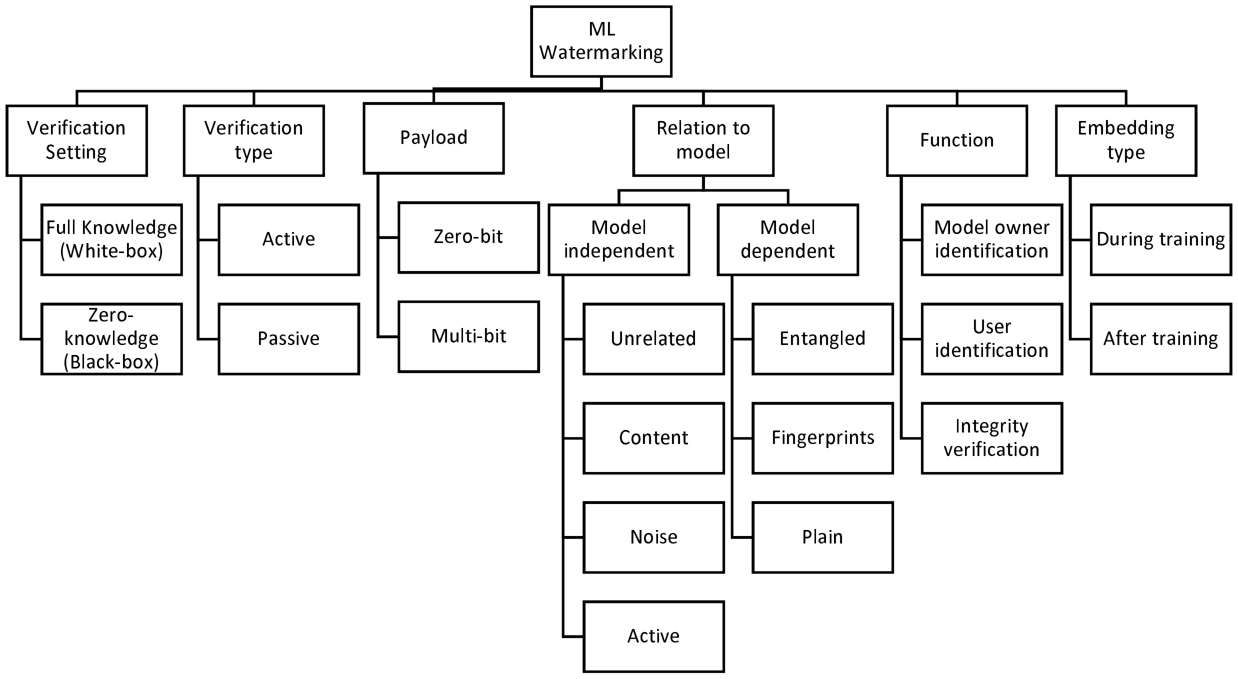

Figure 1: Taxonomy for ML watermarking

5.3.1.1 Full-knowledge watermarking

Full-knowledge watermarking (sometimes denoted as "white-box watermarking") modifies model parameters or architecture in a way that the inserted change can be related to the origin of the model. Modification of parameters can consist, for instance, in embedding of a message (e.g. of a binary vector) using a secret key into the parameters of the convolutional layers during training [i.17], [i.21] and [i.22]. The secret key has usually the form of a secret pseudo-random vector or matrix. To embed the watermark, the original cost function is modified by adding an additional term depending on this binary vector and a secret verification key. Ownership verification is performed by analysing the look of the model, e.g. its parameters distribution, and requires the use of the right secret key. Generative Adversarial Network (GAN) can be used during watermarks embedding to improve both accuracy of the marked network and its covertness (similarity between the weight distributions of a watermarked and a non-watermarked model) [i.22]. The generator corresponds then to the watermarked model training and the discriminator to the watermark detector.

A different type of full-knowledge watermarks does not modify parameters but adds whole specially crafted passport layers after the convolution layers of the model [i.20]. The inference performance of a pre-trained DNN model will either remain intact given the presence of these passports or be significantly deteriorated due to the modified or forged passports. Therefore, the ownership verification is performed by analysing the performance of the network with and without the passports. This type of watermarking aims at solving the problem of ambiguity attacks, in which an attacker inserts their own counterfeit watermarks inside of an already marked model.

5.3.1.2 Zero-knowledge watermarking

The idea behind zero-knowledge watermarking (sometimes denoted as "black-box watermarking") is to insert multiple legitimate backdoors into a model in order to enable its further identification [i.18] and [i.19]. During the training phase (or during re-training) a special *trigger set* composed of *key inputs* and their corresponding labels is included into the training dataset. Data in the trigger set is labelled in an incorrect way (from the point of view of the classification tasks) and thus it introduces a deviation to the model behaviour that will be later used to identify its ownership.

The zero-knowledge watermark detection is done by triggering the model using the key inputs and observing the corresponding outputs. An unmarked model should classify the key inputs according to its main classification task, while for a marked model the unusual behaviour should be observed, which will enable model identification.

Different types of zero-knowledge watermarking techniques were presented and tested for image classification tasks. They can be divided with regard to their relation to the model into *model dependent* and *model independent* techniques. In model independent techniques, the watermark insertion does not require a special modification of the training procedure as only the training dataset is modified.

5.3.2 Payload

Watermarking techniques can be also classified with regard to their capacity to convey bits of information. *Zero-bit* watermarking techniques just provide information whether a given model contains or not watermarks, while *multi-bit* watermarking techniques convey multiple bits of information that will be extracted from the model during the verification process.

5.3.3 Relation to model

5.3.3.1 Model independent

Following model independent zero-knowledge watermarking techniques can be distinguished:

- **Unrelated [i.18] and [i.19]**. Irrelevant data samples are used as key inputs to watermark the model. Labels for the key inputs are randomly selected from existing classes [i.18] or one label is randomly selected from existing classes and attributed to all key inputs [i.19]. If labels are randomly selected (or according to a user-defined process), then the technique will have multi-bit capacity. If only one label is selected, then the technique will have only zero-bit capacity. First advantage of the unrelated type of watermarking is that it should not have a significant impact on the model accuracy as the samples from the trigger set lie far away from the normal training dataset distribution. Second, if the key inputs are uncorrelated, then revealing one of the key inputs will not leak any information about other key inputs to the attackers. However, the decorrelation between key inputs and the rest of the dataset can be also a disadvantage, making the technique particularly vulnerable to watermark removal techniques. Moreover, inside a verification query 'unrelated' key inputs may be easily distinguishable from normal inputs.

- **Content [i.19]**. To form key inputs, a meaningful content (like logo or text but also specially crafted mark) is embedded into selected data samples from one source class of the original training dataset. One randomly selected label, that is different from the true data samples, is then attributed to the key inputs (and therefore the techniques has a zero-bit capacity). A variation on the 'content' technique, trains the model on both normal samples and samples modified with an almost imperceptible message mark (that can be generated for example from a logo [i.23]), in order to learn the model distinguish between normal data samples and key inputs. The label of the key inputs is predefined or attributed in function of the owner's signature [i.24], in order to make a link between the owner's identity and the watermark. As labels can be different for different key inputs, this variation has multi-bit capacity.

- **Noise [i.19]**. To form key inputs, a certain amount of noise is embedded into selected data samples from the one source class of the original training dataset. As for content watermarking, a randomly selected label, different from the original labels, is attributed to the key inputs and the techniques has zero-bit capacity.

- **Active ("on-the-fly") [i.25]**. The watermarking is performed at the level of the prediction API during the model extraction by an adversary. A small (e.g. < 0,5 %) subset of responses to the queries is modified to return incorrect labels, leading to the watermarking of the surrogate model.

5.3.3.2 Model dependent

In model dependent techniques, the watermarking is tied to the model classification task. This category of watermarking gathers mainly techniques leveraging adversarial examples or using classification boundaries to identify a model.

Some examples of model dependent watermarking:

- **Entangled watermarks [i.26]**. Watermarks are generated from any two similar classes that the model does not misclassify. The model owner chooses two classes, the source and target, which have high average cosine similarity. Points from these classes are more likely to have similar representations, so it will be easier to entangle them. Then, a predefined trigger is added to a fraction of the source class points to turn them into watermarks. The trigger is chosen to impact minimally the integrity of the classification for the given source and target classes. As the entangled watermarks are tied to the task manifold, they have an impact on the accuracy of the model (up to 20 % of accuracy loss). As all the key inputs have the same label, the technique has a zero-bit capacity.

- **Fingerprints [i.21]**. Adversarial perturbations of training samples are leveraged to produce watermarks situated close to the model's decision frontier [i.27]. The classifier is fine-tuned so samples that are not watermarks but also lie around the class frontier are well classified. As entangled watermarks, the techniques has zero-bit capacity. Ensemble methods can be used to make it resistant against distillation attacks [i.28]. In order to fully preserve accuracy, techniques that identify a model solely by analysing its classification boundary may be used [i.29].

- **Plain [i.30]**. Key inputs are unmodified data samples with modified labels and the technique relies on overfitting. This watermarking technique implies a special 'exponential weighting' training procedure that will reinforce the model resistance against modifications (as pruning or re-training the model could easily erase the overfitting of the model to key samples). More specifically, during the training process, the parameters of the neural network model that significantly contribute to predictions are identified and their weight value is increased exponentially, so that model modification cannot change the prediction behaviour, including predictions for key samples. Depending on the label attribution strategy, the technique can have zero-bit or multi-bit capacity. A multi-bit variation to the technique includes a pre-processing step that clusters all class labels into two groups [i.31]. The samples from one cluster are transformed using a targeted adversarial attack in order to make the model predict them as any class from the other cluster.

5.3.4 Requirements for efficient ML watermarking

ML watermarking is still a developing research track. Existing techniques tend to address one problem at time rather than provide a complete solution for ownership rights violation detection. However, some preliminary requirements for efficient watermarking could be already identified:

- **Fidelity**. Accuracy of the target neural network should not be significantly degraded as a result of watermark embedding.

- **Reliability**. Watermark extraction should yield minimal false negatives; a watermark should be effectively detected (using pertinent keys if needed).

- **Robustness**. Embedded watermark should be resilient against various model modifications, such as ex. pruning, fine-tuning, or watermark overwriting. The list of potential transformations leading to watermark erasure can be long and contain ex. backdoor removal techniques.

- **Integrity**. Watermark extraction should yield minimal false alarms (also known as false positives); the watermarked model should be uniquely identified (using the pertinent keys if needed).

- **Non-repudiation and clear owner identification**. The model owner cannot successfully dispute its ownership and the validity of the watermark. This requirement implicates that there is a clear relationship between the owner identity and the watermarks, ex. the watermarks are stored in a cryptographic vault related with the owner or they are created using the owner's unique key. Similarly, an attacker should not be able to perform an ambiguity attack on already existing watermarks. The link between the legitimate owner and the ML watermarks could be ensured using cryptographic techniques and/or dedicated ML watermarks verification protocols.

- **Capacity**. Ideally, a watermarking scheme should enable the verification of the network multiple times. In the case where the watermarks are kept secret and revealed only during the verification, this implicates that multiple watermarks should be inserted into the model.

- **Efficiency**. Communication and computational overhead of watermark embedding, and extraction should be negligible.

Fulfilling all the requirements equally may be hard and a compromise between security and efficiency may be needed in practice (e.g. huge capacity in zero-knowledge watermarking will most probably impact fidelity, as embedding of a large number of watermarks will deteriorate the network performance).

5.4 Detection of training data misuse

AI-specific techniques could be also used to detect the theft of training datasets. The radioactive data approach [i.32] marks data and detects their illegitimate usage inside of ML training datasets. The technique differs from classical model watermarking as it aims at providing a proof of ownership of its training data and not the model itself. It consists in embedding of a special, almost invisible, 'radioactive' mark inside of a subset of images belonging to a dataset in order to detect later if the dataset was used to train a given model. The usage of 2 % or more of radioactive data in the dataset during the training will mark a model in a significant way. It is possible to verify if a model was trained on radioactive data in both full-knowledge and zero-knowledge setting. The impact of radioactive data on the accuracy of the model should be limited as long as the fraction of radioactive data is under 10 %.

5.5 Threats against AI-specific tracing mechanisms

5.5.0 Introduction

ML watermarking may be the target of different attacks. A summary of most common threats to ML watermarking and their countermeasures is presented in Table 2.

Table 2: Summary of most common attacks on ML watermarking

Attack type	Description	Attack examples	Setting	Defence mechanisms
Model extraction	Creating a copy of the victim model.	• Model extraction using predictions of the victim model available through an API • Model extraction using side-channels • Model extraction using model explanations	Full-knowledge/ Zero-knowledge	**Preventing extraction:** • Queries filtering • Network transformation disabling stealing **Theft detection:** • Entangled watermarks • Marking the extracted model through modified predictions
Watermark removal	Removing a watermark or weakening it strength.	• Fine-tuning, e.g. REFIT framework • Compression • Fine-pruning • Computation optimization • Network transformation • Backdoor detection and removal, e.g. neural cleanse, neural laundering • Collusion attack, e.g. averaging weights • Transfer learning, knowledge transfer, knowledge distillation	Full-knowledge	Robustification User dependent watermarks: • Entangled watermarks • Watermarks from adversarial examples

Attack type	Description	Attack examples	Setting	Defence mechanisms
Creating ownership confusion	Creating confusion by providing a counterfeit ownership proof.	• Inserting new watermarks in the model • Reusing existing watermarks that are not tied to the owner identity	Full-knowledge/Zero-knowledge	• Passport layers • Commitment schemes • Watermarks tied to the owner identity
Evading PI check	Avoiding verification of the stolen model by detecting verification attempts from the legitimate owner.	• Query analysis	Zero-knowledge	• Zero-knowledge watermarks with key samples undistinguishable from the rest of data samples • Verification protocol

5.5.1 Watermark removal

After appropriating a marked model, an attacker will try to disable the possibility of ownership verification by removing the owner's watermarks. Beside of such intentional attacks, watermark erasure may be an unintentional side-effect of processing applied by a legitimated user. In fact, it is a common practice that models undergo various transformations [i.33].

Some techniques that can lead to erasure of a watermark are listed below:

- **Transfer learning**. A model developed for a task is reused as the starting point for a model on a second task [i.34].
- **Knowledge transfer, knowledge distillation**. Predictions of one or more model are used to train a second, more generalized, model [i.35] and [i.36].
- **Fine-tuning**. A technique of transfer learning. A model that has already been trained for a given task is trained to perform a different task. It is often performed on pre-trained models to adapt them to a new use case with less effort than training a network from scratch. Fine-tuning is the main technique behind the REFIT framework for watermarks removal, which works against several types of back-box watermarks [i.37].
- **Compression or parameter pruning**. Parameters whose absolute values are very small are cut-off to zero. This technique is often used to deploy models in embedded systems or mobile devices [i.17].
- **Fine-pruning**. A combination of compression and fine-tuning. It improves over pruning by continuing to train the model after pruning the architecture.
- **Network transformation**. A well-trained neural network is morphed into a new one, so its network function is completely preserved for further training. Modification of the network structure may make full-knowledge watermarks undetectable [i.17], [i.38] and [i.39].
- **Backdoor detection and removal**. Most of zero-knowledge watermarks are legitimate backdoors embedded in the network on purpose by the owner. Therefore, techniques detecting and removing backdoors in deep neural networks, such as neural cleanse [i.40], can be applied to remove such watermarks. For instance, the neural laundering technique [i.41] has shown to be efficient against two zero-knowledge state-of-the art zero-knowledge watermarks ([i.19] and [i.18]).
- **Collusion attack**. A vendor may sell the same model but with different watermarks to multiple users. If the users collude, they can then produce an unmarked model [i.21].
- **Computation optimization**. A convolutional layer in a pre-trained model is decomposed into more convolutional layers with smaller filters, thus reducing the computation complexity. Computation optimization was mentioned in the literature as possibly having an impact on watermarks, but was not investigated yet in this context [i.33].

5.5.2 Ownership check evasion

An attacker may share the stolen model in the form of a cloud service, which will expose only model's API to the users. As full-knowledge verification of the model will not be possible, the legitimate owner will try to prove its ownership in a zero-knowledge setting, querying the cloud service with key inputs and looking for watermarked behaviour. To avoid ownership verification, the attacker may build a query detector that will inspect if a query is a clean one or a possible attempt to verify a zero-knowledge watermark. Once the detector decides the query is a possible verification query, the stolen model will return a random label from its output space.

5.5.3 Ambiguity attacks

An attacker may aim to cast doubt on the ownership verification by forging additional watermarks or reusing existing watermarks as their own. In the first case, an attacker will integrate new watermarks into an already existing model and therefore cast doubt about the legitimate ownership. They can also leverage adversarial examples to find abnormal model behaviour that could be interpreted as watermarks. In the second case, an attacker may use ex. a model inversion attack [i.42] to reconstruct the watermarks. The ambiguity attack supposes the existence of a third-party that is responsible for the identification of a model and therefore may have to deal with a situation where two independent parties provide two ownership proofs for the same model.

A way of preventing ambiguity attacks is to associate the watermarking technique with a watermarking registration/verification protocol. For instance, the owner may lock the information about their watermarks inside a cryptographic vault that will be open only during model verification [i.18] or they may undergo a verification aiming at proving that they have both a watermarked and a non-watermarked model [i.43]. Another solution would be to rely on a secret information that has to be provided by the owner to enable the watermarks verification [i.20]. In addition, ML zero-knowledge watermarking can be reinforced using cryptographic one-way hash functions (e.g. labels depend on the result of a one-way hash function taking key inputs as input) in order to provide a measurable ownership proof in terms of value and security [i.44].

NOTE: Providing a PKI-style infrastructure for ML watermarking verification may be very costly. It is probable, that as for multimedia watermarking, ML watermarking will stay an informal way of detecting ownership rights misuse. Ambiguity attacks are not relevant in such context.

6 Traceability for trustworthy AI

6.1 Classical provenance concepts in the context of AI

Provenance refers to any information describing the lineage of an information object. For example, for a given piece of data, its provenance information may indicate how/when the data is produced and by which party, etc. There are some existing standard data models for representing the provenance information such as Open Provenance Model (OPM) [i.45] and W3C PROV [i.46].

- OPM leverages the directed acyclic graph to describe the provenance of entities. One of the objectives of OPM is to enable provenance information exchange between different systems. In order to do that, a unified provenance model needs to be defined. For that purpose, OPM defined three types of nodes:

    1) An Artifact node is to represent an entity/object;

    2) A Process node is to represent the action performed by the artifact(s) and the execution of a process may also lead to one or more new artifact(s);

    3) An Agent node represents the associated entity that may enable, control, facilitate or affect a process. There are links built between different nodes, which are used to model their dependencies and relationships. By using OPM, it is easy to describe how an artifact is derived.

- W3C PROV model is composed of 12 description documents, of which PROV-DM (Data Model) is the core document. The PROV-DM model contains two important components: Concepts and Relationships. The concept includes three types of elements: Entity, Activity, and Agent. An entity refers to a thing that needs to be recorded for traceability purposes. An activity models the process of changing the state of an entity whereas an agent is an entity that takes certain responsibilities and affects an activity. In PROV-DM, seven core relationships are defined, including Used, WasAssociatedWith, WasAttributeTo, WasDerivedFrom, and WasGeneratedBy for describing traceability-related facts.

In the context of AI, provenance information (i.e. AI provenance information) can be used for different objectives:

- The first is to support AI training data creditability or trustworthiness, since AI models are often built or trained based on training data. The quality and the provenance of training data are essential to justify the creditability of an AI model. For example, if an AI model was trained based on a low-quality or erroneous data set, this AI model may not get used due to its low creditability.

- The second objective is to support reproducibility. For example, different research teams may want to re-produce the training process of an AI model in order to validate the model. Accordingly, the provenance information needs to be recorded to describe how the AI model was trained, which training data is used, and what hyper-parameters were selected.

- The third objective is to support AI explainability. For example, for the outputs (i.e. the predictions) of an AI model, the provenance information may be needed to describe why such a prediction is being made.

It is worth noting that data provenance information needs to be carefully managed especially under distributed system environment for the following reasons:

- First, to store data provenance information needs to be well balanced between the potential overhead and its benefits. For example, one of the approaches to support data provenance is to annotate a lot of metadata associated with the training data, which will definitely enable data transparency but also introduce massive data provenance information and resulted storage overhead.

- A limitation of the existing W3C PROV model is that it only specifies a standard data model for representing provenance information, but it does not address the issue related to how to maintain/store data provenance information and make this information trustworthy.

  EXAMPLE: A malicious entity may tamper with the provenance information about a training dataset, for instance, to modify the creator of the data set or modify the data pre-processing operation logs. As a result, such provenance information may become inaccurate or fake. A solution to this issue could be to store AI provenance information related to AI training data and AI processing to distributed ledgers. In this way, such AI provenance information will become immutable and can be trusted especially under distributed system environment.

6.2 AI-specific traceability needs

6.2.1 Traceability of data

Different aspects of security of the data supply chain were already addressed in ETSI GR SAI 002 [i.47]. This clause extends the latter report with an insight into the traceability aspect of data security.

The traceability of data can be helpful to realize data transparency. This is because the performance of an AI system in terms of AI model accuracy may largely depend on the training data quality. Therefore, the trustworthy AI relies on a trustworthy data set or data sources. To justify the trustworthiness of data, data traceability should be in place. Data provenance information may describe the following key aspects: who generated this data set, when this data set was generated, where this data set is hosted/stored, what are the characteristics (quality, features, etc.) of the data set, etc. It is often the case that raw data may not be ready to use by the AI system. Accordingly, certain data pre-processing may be needed in order to conduct data cleaning or other transformation operations. Therefore, how the data is being cleaned or pre-processed may also need to be traced.

More importantly, in a distributed AI scenario where the AI processing pipeline is distributed among different entities, more sophisticated AI data provenance management is needed. In this scenario, AI applications may initiate various AI processing tasks in an ad hoc way and those AI tasks may require AI processing to be executed by one or more AI Hosts (AIHs), where each AIH may host an AI Agent (AIA) for conducting AI-related processing (training or inferring) and/or some AI data to be used by AI processing. Federated Learning (FL) is a good example in which multiple FL participants (as AIHs) use their local data and computing resources to collaboratively train an AI model.

To generate data provenance information and data trace records is the first step. The next step is to guarantee the security of data trace records. For example, data trace records may need to be securely stored and cannot be altered; distributed ledgers with its immutability property can be leveraged to store data trace records. Also, the access to data trace records need to be secure and only open to authorized parties.

6.2.2 Traceability of the processing pipeline

AI processing pipeline also needs to support traceability. The first reason is that AI reproducibility needs to be realized. For example, a research team-1 may intend to reproduce the AI training process for an AI model X, which was generated by another team-2. The second reason is to support AI explainability. It is worth noting that AI processing can be either centralized or distributed, which may lead to the need for different traceability solutions:

- In a centralized AI processing case, all the processing is conducted by a centralized node/server. In other words, all the stages of the pipeline are executed by the same node. Various types of information can be recorded as provenance information. For example, the basic training environment information needs to be recorded such as software running environment, OS information, etc. Next, the input and output information of each stage can be recorded. During the model training stage, training-specific information is also essential for reproducing the training process, such as the hyperparameter selections/decisions, parameter values that are randomly set, etc.

- In distributed AI processing case, multiple nodes or parties collaboratively work together to complete an AI processing pipeline. In this scenario, more information needs to be recorded. For example, in addition to recording AI processing information about each node, other system-wise or network information needs to be recorded, such as the data flow and interactions among different nodes. In one example, node A is responsible for the data pre-processing stage, and the output of node A will be sent to node B that is responsible for the AI model training stage. In another example, even within the same AI training stage, multiple nodes may be involved (e.g. to split and distribute model training to multiple distributed nodes). Accordingly, how those nodes collaborate with each other may need to be recorded as provenance information of the AI processing pipeline. In addition, other system context information also needs to be recorded, such as the network conditions or connectivity qualities between different parties since this information may explain the performance of the AI processing pipeline. For example, an AI training delay may be not solely due to the insufficient computing power of different nodes, but also by caused by the poor network connectivity among them.

In distributed AI processing case, an important traceability-related issue is how to coordinate those multiple nodes (e.g. AIAs) so that they can efficiently and flexibly trace the AI processing pipeline. For example, traceability can be jointly designed and integrated with AI task deployment and/or AI model deployment:

- First, a traceability management node can be used to configure appropriate trace instructions to different AIAs; a trace instruction can specify what kinds of information should be logged by the selected AIAs during the execution of an AI task.

- Once an AIA obtains the AI task, it will install the AI task, prepare the needed AI data set, and/or connect to other AIAs if needed. The AIA may also configure its local AI tracing module based on the received AI trace instructions, in order to generate desired AI provenance information.

- Similarly, an AIA running an AI model may also be configured with trace instructions in order to record who are the consumers of this AI model for inference and/or detect whether any evasion attack exists.

- During the whole AI pipeline processing, all the created AI provenance information may be logged into a distributed ledgers so that the traceability information may not get malicious manipulation, which is essentially important to realize trustworthy AI.

6.2.3 Traceability of outputs

6.2.3.0 Introduction to outputs traceability

The outputs of an AI model may also require traceability, which has a number of potential benefits:

- The first benefit is to support AI explainability. For example, when an input is sent to an AI model, a prediction output is generated, along with useful information regarding why such a prediction is made. In many critical applications, such as medical AI, an explainable prediction is essentially important to justify its reasonability in order to convince doctors to adopt the prediction result during diagnosis.
- The second benefit is to record AI model inheritance. For example, in a transfer learning scenario, a trained AI model for application A may be treated as an initial AI model for training by another application B. In particular, it is known that intrinsic AI model defects may also be inherited by successors. Therefore, the traceability of AI model inheritance is very beneficial for AI model diagnosis.
- The third benefit is to record useful information for detecting any potential security attack, such as an AI evasion attack. By recording all the inputs to an AI model and its prediction results, it is possible to identify potential evasion attacks. For example, if two similar inputs (e.g. 99,99 % similarities) get two significantly different prediction results, it is possible that the AI model is experiencing an evasion attack.
- The fourth benefit is to detect the potential data set drift. In general, an AI model is trained based on certain training data. In practice, after the AI model is deployed, over time, the inputs of data for prediction may change, e.g. having a different data distribution compared to the original training data. The traceability information can be helpful to detect potential data drift so that the AI model calibration or re-training can be conducted in time.

6.2.3.1 Types of concept drift

Concept drift is a common phenomenon in the field of AI where data distribution changes. The data distribution refers to the joint probability distribution in the input data x and output data y, i.e. $p(X, y)$. Concept drift means that the joint probability distribution changes. The situation that causes the model posterior probability $p\,(y|X)$ to change is called *real concept drift*, and other types of concept drift are called *virtual concept drift* [i.48]. Real concept drift has higher impact on model performance than virtual concept drift:

- **Causes of concept drift.** Concept drift is usually caused by external scenarios. For example, the change from urban to rural environment in autonomous driving, and the change of user purchasing interest with the season. This is mostly the change of $p(X)$, which is consistent with 'data set drift' mentioned in the present clause. The other is the change of the decision boundary $p(y|X)$. For example, with the expansion of network bandwidth over time, the definition of abnormal burst traffic changes too.
- **Concept drift classification from the perspective of drift transition.** Concept drift can be divided into sudden drift, incremental drift, gradual drift, recurring drift, and blip drift [i.49]. Sudden drift refers to rapid and irreversible change. Both incremental and gradual drifts refer to the slowness of changes. Recurring drift is a temporary change that restores the previous state within a short period of time. A blip drift is a rare event that can be considered anomaly or an outlier.

Concept drift can be managed by detection or adaption. Concept drift detection can use a data-oriented approach, which treats the concept drift problem as an independent step before model optimization. Concept drift adaption can use a model-oriented approach, which tends to manage concept drift implicitly by model adaptation and model generalization.

Concept detection methods include three strategies: probability distribution-based method, statistical features-based methods, and model-based feature methods:

1) Probability distributions of input data can be approximated by parametric methods (such as Gaussian distribution) and nonparametric methods (such as kernel density estimation). Probability distribution difference could be calculated by Kullback-Leibler divergence [i.50].

2) Statistical features include mean, median, variance and so on. Concept drift happens when statistical features change beyond a threshold [i.51].

3) Model-based feature method tries to extract high-dimensional features from data through deep learning or other methods. This method is more effective especially when data distribution is difficult to distinguish when applied statistical features [i.52].

Concept drift adaption methods include model adaptation and model generalization methods:

1) Model adaptation methods include continual learning, transfer learning, ensemble learning and meta-learning [i.53]. Continual learning aims to learn an adaptive model for new tasks (concepts) sequentially without forgetting old knowledge. Transfer learning is related to problems such as multitasking and domain adaption where concept drifts naturally exist. Ensemble learning can train multiple learners according to different conceptual scenarios. Meta-learning learns novel concepts through a small number of new samples.

2) For model generalization methods, data augmentation [i.54] and [i.55], causal inference learning [i.56] and modularity learning [i.57] can improve the model generalization ability in the concept drift scenario.

Concept drift detection has a number of evaluation metrics [i.48], including:

1) False alarms: The ratio of the number of false alarms in detection to the total number of detections.

2) Missing rate: The number of drifts that actually occurred but was not detected by the detection method.

3) Drift detection delay: The time interval between the detection point of concept drift and the actual concept drift point. Generally speaking, the time delay is used to measure the timeliness of the detection method for concept drift detection.

6.2.4 Model metadata and lifecycle management

In order to support full-fledged AI model traceability, it is necessary to record essential information across the whole lifecycle of an AI model, referred to as model trace records. For example, MLOps (DevOps for ML) becomes popular in the sense that not only "how to build models" should be managed and recorded, but "how to run/operate models" should also be managed. MLOps allows the developers to apply frequent small/incremental upgrades to the AI model on a daily basis. Model trace records need to be securely stored and only be accessible to authorized parties. Distributed ledgers can be leveraged to store model trace records to prevent any party from modifying them:

For example, [i.58] defined an MLOps framework, which extends the software lifecycle management to the AI domain and aims to enable customization/trust/traceability/flexibility of AI pipelines.

Vartak et al. [i.59] also focused on the model lifecycle management aspect. The major observation of this research is that it is very difficult for AI researchers/engineers to remember all the details of an AI training process and the obtained insights. Accordingly, this work proposed the so-called ModelDB, which is to provide an end-to-end AI model management solution. ModelDB may allow a user to track all the provenance information of an AI model by conducting visual exploration and analysis.

6.2.5 Reproducibility of the training process

6.2.5.1 Definition of reproducibility

Reproducibility is a fundamental concept in science, though the exact terminology and definition varies [i.60]. One general definition for reproducibility is that the results of a study or an experiment can be repeated identically by following the same methodology. In the domain of machine learning, the concept of reproducibility has been discussed in research and practice. Following the similar definitions in [i.64] and [i.61], a machine learning model's training process is *reproducible*, if under the same training setup (e.g. the same training dataset, training code, environment), the trained model produces the same results under the same evaluation criteria [i.62]. The evaluation criteria may be defined for a data sample (e.g. inference results) or over a data distribution (e.g. performance metrics).

NOTE: The concept is to be distinguished from *replicable*, which means under a different data sample (from the same distribution as the original data sample) combined with original code and analysis yields similar results.

Reproducibility of the model training process is helpful in the development of AI models in the following ways:

- It is critical for the training process to be reproducible for debugging and testing, so that the problematic behaviour can be consistently reproduced. When developers make changes to any part of the AI training process (let it be environment, data, or code), it would be unclear whether the difference between the newly trained model's performances is resulted by the intentional changes from the developers, or simply from randomness, if the training process is irreproducible.
- Reproducible training process is critical for auditing and claim verification. Auditors and customers would not be able to verify the performance and behaviour claims of a model based on one training process, if such process cannot be reproduced even when all assets are traced. In this regards, reproducibility complements traceability in achieving a more trustworthy AI model.

6.2.5.2 Evaluation and measurement of reproducibility

As defined in clause 6.2.5.1, the reproducibility of a model's training process has a prerequisite that the training setup is consistent:

- Most of the training setups are textual or numerical in nature (e.g. training code and environment configurations) and therefore straightforward to ensure consistency.
- For assets such as datasets that are not necessarily textual or numerical in nature, comparisons with hash functions (e.g. SHA256) can be used to ensure binary consistency.

Once the prerequisite is guaranteed, one can repeat the training process for two or more times to measure the reproducibility across multiple training processes. There are multiple different evaluation criteria that can be used, with different levels of difficulties to satisfy. Examples categorized based on level of difficulties to satisfy are described below:

- Criteria with lower level difficulties to satisfy:
    1) **Metric-level correctness.** Several different performance metrics can be used to assess the reproducibility of the correctness of models, including but not limited to: *accuracy*, *precision*, *recall*, *F1-score*, *AUC* (for classification task), *mae_loss* (for regression task). These metrics should be calculated across multiple runs (≥ 30) [i.63] to assess the variance of the distribution [i.64]. In addition, for classification tasks, class-level metrics can be evaluated in addition to overall metrics.
    2) **Model complexity.** In the field of AutoML, neural network architectures can be generated automatically using a search-based approach [i.65]. These automatically generated neural network architectures can also have reproducibility. To evaluate whether the generated neural network architecture is reproducible, model complexity metrics can be used. Common metrics for model complexity includes *model size*, *number of parameters*, and other network-related complexity metrics [i.66].
    3) **Resource consumption [i.67].** To evaluate model reproducibility by resource consumption, the resources consumed by the training and inference process should be clearly stated. Metrics of evaluating training process resource consumption include *training time* and *training epochs*. Pham et al. [i.64] discovered that the training time across different runs of the training process can vary up to 4 014,8 %. This result reflects the impact from randomness in the training process. In addition, the resources consumed by the inference process can also be evaluated using metrics such as *inference time*.
- Criteria with higher level difficulties to satisfy:
    1) **Instance-level correctness.** To evaluate the reproducibility of the instance-level correctness of a model, one can compare the inference results of individual data points in a given test set across different training runs. Metrics such as the ratio of data points with consistent inference results over the total number of data points can be calculated.
    2) **Model robustness [i.68].** Model robustness metrics such as misclassification ratio can be used to evaluate a model's ability to handle adversarial attacks. The consistency of such metrics under the same adversarial inputs across several training runs can be used to evaluate the reproducibility of model robustness.

3) **Model interpretation [i.67].** Many techniques have been proposed to provide interpretations of a model. At a global (model) level, interpretations can be derived using techniques such as permutation feature importance score [i.69]. At a local (instance) level, interpretations can be derived using techniques such as LIME [i.70], Anchor [i.71], and SHAP [i.72]. High reproducibility of model interpretation is an important aspect of high quality interpretations. For example, to evaluate the reproducibility of global interpretations, rankings of feature importance can be extracted from each run of training process, and then compared using metrics such as *Kendall's tau* (also known *as* Kendall rank correlation coefficient), *Kendall's w* (also known as Kendall's coefficient of concordance) or *top_n overlap score [i.73]*.

EXAMPLE 1: Assume that the training setup is determined (i.e. the training data, training code, software and hardware environments are set) for a given binary classification task. To evaluate the reproducibility of the training process, one can repeat the training twice and obtain two trained models under such identical training setup. The two trained models are then tested on the same test dataset. If the accuracies of the two models on the test dataset are identical, one can conclude that under the metric-level correctness evaluation criteria, the training process is reproducible.

EXAMPLE 2: Furthermore, if for every data point in the test dataset, the two models produce identical inference results, one can then conclude that under the instance-level correctness evaluation criteria, the training process is reproducible.

NOTE: Metrics listed above are often real numbers, and hence when compared, should take into consideration the precision of floating-point numbers stored in memory.

6 Conclusion

The present document summarizes existing and potential mitigation approaches against the misuse of the ownership rights, as well as presents existing challenges related to traceability in the context of ML. Classical Digital Right Management solutions are presented in the light of ML in the first part of the present document. Then, novel AI-specific techniques enabling models identification are presented along with the related threats and countermeasures. The second part of the present document is dedicated to the problem of traceability for trustworthy AI. It starts with the description of classical provenance concepts in the context of AI. It follows with a description of AI-specific traceability needs through the whole lifecycle of a model, which should be addressed in order to enable a good reproducibility of the results.

History

Document history		
V1.1.1	February 2024	Publication