Physics solutions for privacy leaks in machine learning

Alejandro Pozas-Kerstjens
Institute of Mathematical Sciences
(CSIC-UCM-UC3M-UAM)
28049 Madrid, Spain
physics@alexpozas.com

Senaida Hernández-Santana
Departamento de Matemática Aplicada a la Ingeniería Industrial
Universidad Politécnica de Madrid
28006 Madrid, Spain

José Ramón Pareja Monturiol
Institute of Mathematical Sciences
(CSIC-UCM-UC3M-UAM)
28049 Madrid, Spain

Marco Castrillón López
Departamento de Álgebra, Geometría y Topología
Universidad Complutense de Madrid
28040 Madrid, Spain

Giannicola Scarpa
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos
Universidad Politécnica de Madrid
28031 Madrid, Spain

Carlos E. González-Guillén
Departamento de Matemática Aplicada a la Ingeniería Industrial
Universidad Politécnica de Madrid
28006 Madrid, Spain

David Pérez-García
Departamento de Análisis Matemático y Matemática Aplicada
Universidad Complutense de Madrid
28040 Madrid, Spain

Abstract

We show that tensor networks, widely used for providing efficient representations of quantum many-body systems and which have recently been proposed as machine learning architectures, have especially prospective properties for privacy-preserving machine learning. First, we describe a new privacy vulnerability in feedforward neural networks, illustrating it in synthetic and real-world datasets. Then, we develop well-defined conditions to guarantee robustness to such vulnerability, and we rigorously prove that these conditions are satisfied by tensor networks. We supplement the analytical findings with practical examples where matrix product states are trained on datasets of medical records, showing large reductions on the probability of an attacker extracting information about the training dataset from the model's parameters when compared to feedforward neural networks.

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

1 Introduction

Vast amounts of data are routinely processed in machine learning pipelines, every time covering more aspects of our interactions with the world. When the models processing the data are made public, is the safety of the data used for training it guaranteed? This is a question of utmost importance when processing sensitive data such as medical records, but also for businesses whose competitive advantage lies in data quality.

It is known that, in general, deep learning models leak information about the data used for training it. For instance, it is possible to extract the presence or absence of a particular datapoint in the training dataset, even when only having access to input-output queries to the model [1]. Moreover, these models can be, after training, repurposed for performing privacy-violating tasks without the need of the original data [2]. Importantly, the known techniques for avoiding these privacy leaks are based on the addition of noise via differential privacy mechanisms [3] or constraining the optimization space [2], thus impacting the model's performance. In this regard, the ideal model would only retain from the training dataset the information that is essential to perform well. As a consequence, access to the model's parameters in one such models would not be more informative about the data used for training than having a description of it by means of recording the outputs generated for different inputs. This problem is similar in spirit to the protection of software against Man-At-The-End attacks [4].

In this work we connect the notion of equivalence between black-box and white-box access to a trained model with the concept of gauge symmetry, which is commonplace in many areas of physics, and we demonstrate theoretically and illustrate experimentally that tensor networks (which are motivated by the study of quantum many-body systems [5]) are a promising source of privacy-preserving machine learning architectures. Moreover, we rigorously prove that in specific tensor network architectures it is possible and easy to find alternative parametrizations of a model that are as informative as a black-box access to it. These architectures, known as matrix product states (MPS) [6], are promising learning models [7, 8] that now compete [9–11] or even surpass [12] traditional architectures in certain situations. The erasure of information irrelevant for the task at hand is achieved with no impact on the model's performance, in contrast with solutions based on differential privacy.

2 A new privacy vulnerability in feedforward neural networks

Training neural networks is routinely performed via gradient-based optimization. Also, most model classes contain at some point concatenations of parametrized affine transformations, of the form $\boldsymbol{y}^{(l)} = W^{(l)} \cdot \boldsymbol{z}^{(l-1)} + \boldsymbol{b}^{(l)}$, and fixed nonlinearities, $\boldsymbol{z}^{(l)} = \phi_l(\boldsymbol{y}^{(l)})$. These parameters, after training, contain information about the training dataset that, ideally, a model should not reveal.

For simplicity, let us consider a hypothetical situation where all points in the training dataset have one binary feature which takes always the same value, say 1, and the model is a concatenation of affine transformations and nonlinearities. Since all the points in the dataset have the same value of the binary feature, this is of no use for the target task, and thus an ideal final model shall not depend on it. As we show in Fig. 1, during training, the biases and weights in the model are driven to neutralize the effect of this irrelevant variable on the final prediction. The corresponding weights and biases are directed towards taking opposite signs if the binary feature takes the value 1, and towards taking same signs if it takes the value -1. An attacker that has access to the model can thus easily recover the nature of the irrelevant feature just by looking at the parity of some of the parameters.

In more realistic scenarios it is more common to have features which have some imbalance between the different values it can take. We demonstrate in Fig. 2b that in such situation this type of vulnerability is still present. There, we show an illustration with deep neural networks trained on real-world data of medical records derived from the `global.health` database [13]. The task in which the neural networks are trained is the prediction of the outcome of an infection given demographics, symptoms, and the parity of the date when the case was recorded (this is the feature irrelevant to the task). Then, the attacks follow the spirit of shadow training [1, 14]. Further details on the dataset, the prediction task, and the attack can be found in appendices E and F. Notably, for the case of neural networks, simple logistic regression attacks perform very well, highlighting the importance of the vulnerability.

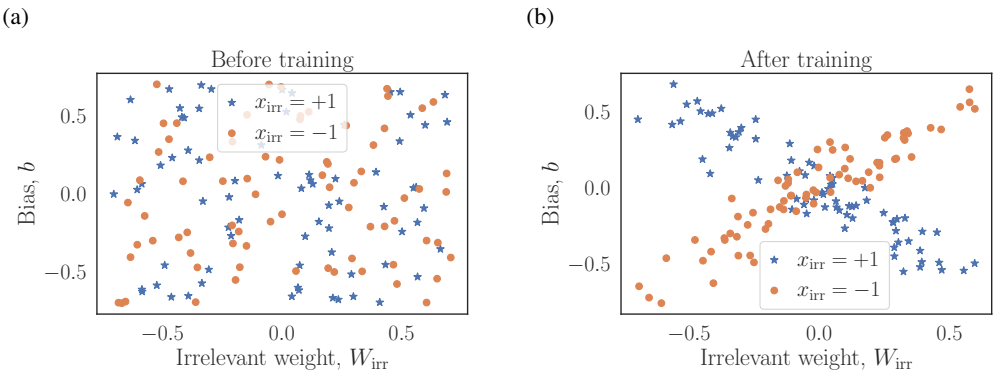


Figure 1: Illustration of the proposed vulnerability. Each point represents a simple neural network model, $f_\theta(\boldsymbol{x}) = \phi\left(W_{\text{rel}}x_{\text{rel}} + W_{\text{irr}}x_{\text{irr}} + b\right)$, that is trained to learn the function $y(\boldsymbol{x}) = \text{sign}(x_{\text{rel}})$. Each model is trained on a different dataset where $x_{\text{rel}} \sim \mathcal{N}(0, 1)$, and x_{irr} is $+1$ for all datapoints used to train the models depicted by the blue stars and -1 for the orange circles. The plots show the values of W_{irr} and b (a) before and (b) after training on a different, random dataset for each model. In this architecture, the gradients of any loss function $\mathcal{L}$ satisfy $\partial_{W_{\text{irr}}}\mathcal{L} = x_{\text{irr}}\partial_b\mathcal{L}$, so they will drive the parameters to distinguishable regimes, which can be identified after training as demonstrated in (b).

3 Privacy from reparametrization invariance

Continuing with the example, a way to erase the information about irrelevant features that does not lead to privacy leaks is setting to zero the corresponding weights, so the influence of those features is not propagated through the network. In a hypothetical situation where one fixed such parameters and trained the rest, the result would be an alternative collection of weights and biases leading to a model, ideally equally performing, yet not containing any information about the training dataset other than that needed for making the prediction. Thus, the ability to characterize the sets of parameters that lead to the same model opens the door to choosing parameters which contain no more information about the training dataset than what can be inferred from recording outputs and corresponding inputs.

This invariance under reparametrizations is commonly known as gauge symmetry [15]. Gauge symmetry is a concept that is commonplace in many-body physics, nuclear and particle physics, or in general relativity. Our first result is the connection of the notion of gauge symmetry with that of equivalence between white-box and black-box access to a model, namely:

Observation 1. *Consider the set of white-box representations for some architecture, $\mathcal{W} \in \mathbb{C}^n$, and its associated set of black-box representations, $\mathcal{B} = \pi(\mathcal{W})$ for some function π that assigns every to white-box representation of a model its corresponding black-box oracle. If a right inverse $\alpha : \pi \circ \alpha = id_{\mathcal{B}\to\mathcal{B}}$ can be defined, then for every function $f : \mathcal{W} \to \mathbb{C}$*

$$f(\hat{\theta}) = \hat{f}[\pi(\hat{\theta})],$$

where $\hat{\theta} = \alpha \circ \pi(\theta) \in \mathcal{W}$ are the parameters denoting a canonical form *for the model described by θ and $\hat{f} = f \circ \alpha$ is the application of f in black boxes.*

The proof and further details on the notation used can be found in Appendix A. Observation 1 means that the evaluation of an attack (modeled in a generic way by a function f that maps the parameters of a model to a complex number representing the desired information) in a set of parameters that describe the canonical form of a model, $\alpha \circ \pi(\theta)$, coincides with the evaluation of the induced attack, $\hat{f} = f \circ \alpha$, in the associated black box, $\pi(\hat{\theta}) = \pi(\theta)$. Therefore, an attack cannot extract from a canonical form any information that is not present in the associated black box. Note that this result is not restricted neither to MPS, to general tensor networks, nor to neural networks.

4 Global canonical form in MPS

We now know that if a canonical form can be defined, then the information about the training dataset stored in the model's parameters is the same information that is stored in the corresponding black box.

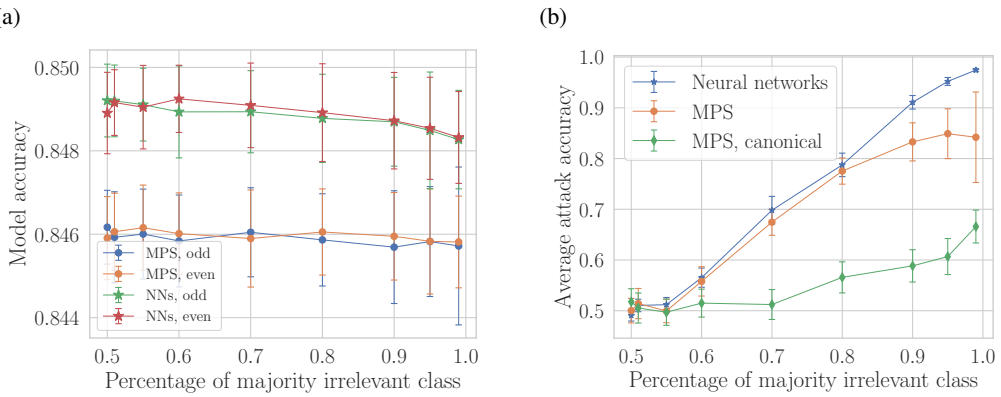


Figure 2: Comparison between deep neural networks and MPS when learning a model predicting the outcome of infections given demographics and symptoms in a real-world dataset [13]. Figures of merit are computed as a function of the percentage of dominant value in the irrelevant feature, namely the parity of the day of reporting. (a) depicts the average accuracy in the whole database from which the different training sets are generated. The fact that models trained on datasets biased towards different values of the irrelevant feature perform equally indicates that the feature is indeed irrelevant. (b) represents the accuracy of attacks predicting the dominant value of the irrelevant feature in the training dataset. Importantly, the MPS not expressed in canonical form are vulnerable similarly to the neural networks.

However, it could still be possible that accessing this information is easier when knowing the model's parameters. In this section we prove that this is not the case for MPS [6–8] (see Appendix B for its definition and relevant properties). The reason for this is that the map $\alpha : \mathcal{B} \to \mathcal{W}$ can be taken to be holomorphic, and hence possesses the highest degree of smoothness. Then, since holomorphy implies that the composition of any function with α preserves its regularity, any attack at the white-box level, $f : \mathcal{W} \to \mathbb{C}$, can be upgraded to an attack at the black-box level, $\hat{f} = f \circ \alpha : \mathcal{B} \to \mathbb{C}$, with the same regularity. Therefore, for every white-box attack to a canonical-form representation not only there exists a black-box attack with the same performance (as we showed in Observation 1), but also with the same regularity. Our second result is an explicit construction of a holomorphic map α through the following theorem:

Theorem 2. *For an MPS defined by a collection of tensors* $\{A_{s_j}^{\beta_{j-1},\beta_j}\}_{j=1}^N$*, a global, univocal, holomorphic canonical form* $(\alpha \circ \pi) : \mathcal{W} \to \mathcal{W}$ *is given by* $\hat{A}_{s_1}^{\beta_1} = \mathbb{1}$*,* $\hat{A}_{s_N}^{\beta_{N-1}} = \mathbb{1}$*, and* $\hat{A}_{s_j}^{\beta_{j-1},\beta_j} = \sum_\gamma L_{s_j}^{\beta_{j-1},\gamma} C_{\gamma,\beta_j}$*, where*

$$L_{s_j}^{\beta_{j-1},\gamma} = \sum_{\{\alpha\}} A_1^{\alpha_1} \cdots A_1^{\alpha_{j-3},\alpha_{j-2}} B_{\beta_{j-1},s_j,\gamma}^{\alpha_{j-2},\alpha_{j+1}} A_1^{\alpha_{j+1},\alpha_{j+2}} \cdots A_1^{\alpha_{N-1}},$$

$$B_{\beta_{j-1},s_j,\gamma}^{\alpha_{j-2},\alpha_{j+1}} = \sum_{\alpha_{j-1},\alpha_j} A_{\beta_{j-1}}^{\alpha_{j-2},\alpha_{j-1}} A_{s_j}^{\alpha_{j-1},\alpha_j} A_\gamma^{\alpha_j,\alpha_{j+1}},$$

$$C = L^{-1}, \text{ with } L_{\gamma,\beta_j} = L_\gamma^{1,\beta_j}.$$

This function induces a global, holomorphic map $\alpha : \mathcal{B} \to \mathcal{W}$.

The proof of this theorem and a graphical representation of the proposed canonical form can be found in appendices C and D, respectively. Informally, Theorem 2 indicates that, for MPS, not only the canonical form stores as much information as the black box, but also extracting such information is equally hard in both cases.

5 Experiments

As an illustration of the protective power of the canonical form of MPS, we depict in Fig. 2 the training of one of these MPS architectures in the real-world dataset used in the demonstration with neural networks, as well as the probability of success of attacks based on shadow training [1, 14],

both before and after computing its canonical form. This sort of attacks provides upper bounds to the efficacy of realistic ones because it allows the attacker to have much more information than reasonable. Details on the training and attacks can be found in appendices E and F, respectively. Both types of models, neural networks and MPS architectures, perform very similarly in the target task (see Fig. 2a), while the vulnerability to attacks is markedly different: as can be seen in Fig. 2b, both neural networks and MPS architectures straight out from training are equally vulnerable but MPS in canonical form are not. For instance, at 80% imbalance of the irrelevant feature, attacks on both models have $\sim 78\%$ of accuracy, while in the canonical-form description of the MPS the accuracy drops to $\sim 56\%$, close to the limit of random guessing. Notably, as described in Appendix F, this happens despite the attacks on MPS being more general than those on neural networks.

6 Discussion

This work shows that global information about the training dataset is hidden in the parameters of deep learning models, even if this information is irrelevant for the task at hand. More importantly, it points at physics, and more concretely at the tensor networks used in quantum many-body physics, as a favourable framework where to find architectures that are robust to privacy leaks. In this aspect, our results are encompassed in Observation 1 and Theorem 2, which informally can be understood as:

Observation 1 (Informal version)**.** *If the sets of parameters that leave a model invariant can be characterized, then each white-box attack to a representative of the set is equally performing (in terms of accuracy) as an attack to the corresponding black box.*

Theorem 2 (Informal version)**.** *There is a canonical form for the set of MPS architectures so that every white-box attack to such canonical-form set of parameters is "as good" (in terms of the attack accuracy and its regularity as a function) as an attack to the black-box representation.*

For neural networks, one could imagine obtaining a canonical form by proceeding in the spirit of model extraction attacks [16, 17]. The privacy vulnerability showcased is a priori different to those typically dealt with in the literature via differential privacy. However, whether differential privacy protects against the vulnerability demonstrated remains to be understood. In contrast with defenses based on differential privacy, the canonical form guarantees that defense in tensor networks can be achieved without compromising on model accuracy. Still, more effort must be put into training tensor networks efficiently. Finding new, gauge symmetric architectures (within [18] or outside tensor networks), which are also enough expressive and easy to train, will be a crucial future task.

Acknowledgments and Disclosure of Funding

We thank Thomas Vidick, Juan-José García-Ripoll, Sofyan Iblisdir, J. Ignacio Cirac and Esther Cruz for discussions and comments. This work is supported by the European Union (Horizon 2020 research and innovation programme-grant agreement No. 648913 and ERDF "A way of making Europe"), the Spanish Ministry of Science and Innovation ("Severo Ochoa Programme for Centres of Excellence in R&D" CEX2019-000904-S and ICMAT Severo Ochoa project SEV-2015-0554, and grants CEX2019-000904-S-20-4, MTM2014-54240-P, MTM2017-88385-P, PGC2018-098321-B-I00 and PID2020-113523GB-I00), the Spanish Ministry of Economic Affairs and Digital Transformation (project QUANTUM ENIA, as part of the Recovery, Transformation and Resilience Plan, funded by EU program NextGenerationEU), Comunidad de Madrid (PEJ-2021-AI/TIC-23267 and QUITEMAD-CM P2018/TCS-4342), and the CSIC Quantum Technologies Platform PTI-001. The authors declare no competing interests.

Impact statement

This work describes a fundamental connection between privacy in machine learning and the concept of gauge symmetry. As a foundational work, it will take several steps in order to be applied to societal issues. However, if tensor networks are eventually used for machine learning, this work could motivate its use in applications where privacy is to be conserved, both ethical or unethical. Moreover, this work describes a previously undocumented privacy vulnerability in deep feedforward neural networks, that could be exploited by malicious parties to extract global information about the training dataset that was not intended to be revealed.

References

[1] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, in *2017 IEEE Symposium on Security and Privacy (SP)* (2017) pp. 3–18.

[2] C. Song and V. Shmatikov, in *International Conference on Learning Representations* (2020).

[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith, J. Priv. Confid. **7**, 17 (2017).

[4] C. Collberg, J. Davidson, R. Giacobazzi, Y. X. Gu, A. Herzberg, and F.-Y. Wang, IEEE Intell. Syst. **26**, 8 (2011).

[5] F. Verstraete, V. Murg, and J. I. Cirac, Adv. Phys. **57**, 143 (2008).

[6] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete, Rev. Mod. Phys. **93**, 045003 (2021).

[7] E. Stoudenmire and D. J. Schwab, in *Advances in Neural Information Processing Systems*, Vol. 29, edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., 2016) pp. 4799–4807.

[8] A. Novikov, M. Trofimov, and I. Oseledets, Bull. Pol. Acad. Sci.: Tech. Sci. **66**, 789 (2018).

[9] E. M. Stoudenmire, Quantum Sci. Technol. **3**, 034003 (2018).

[10] I. Glasser, N. Pancotti, and J. I. Cirac, IEEE Access **8**, 68169 (2020).

[11] R. Selvan and E. B. Dam, in *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, Proceedings of Machine Learning Research, Vol. 121, edited by T. Arbel, I. Ben Ayed, M. de Bruijne, M. Descoteaux, H. Lombaert, and C. Pal (PMLR, 2020) pp. 721–732.

[12] J. Wang, C. Roberts, G. Vidal, and S. Leichenauer, Anomaly detection with tensor networks (2020), arXiv:2006.02516 .

[13] Global.health, a data science initiative, `https://global.health` (2021), accessed: 2021-03-22.

[14] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, Int. J. Secur. Netw. **10**, 137 (2015).

[15] K. B. Marathe and G. Martucci, *The mathematical foundations of gauge theories* (North Holland, 1992).

[16] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, in *25th USENIX Security Symposium (USENIX Security 16)* (USENIX Association, 2016) pp. 601–618.

[17] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, in *29th USENIX Security Symposium (USENIX Security 20)* (USENIX Association, 2020) pp. 1345–1362.

[18] A. Molnar, J. Garre-Rubio, D. Pérez-García, N. Schuch, and J. I. Cirac, New J. Phys. **20**, 113017 (2018).

[19] I. V. Oseledets, Dokl. Math. **80**, 495 (2009).

[20] I. F. Oseledets, SIAM J. Sci. Comput. **33**, 2295 (2011).

[21] D. Pérez-García, F. Verstraete, M. M. Wolf, and J. I. Cirac, Quantum Info. Comput. **7**, 401 (2007).

[22] G. Vidal, Phys. Rev. Lett. **91**, 147902 (2003).

[23] J. Haegeman, M. Mariën, T. J. Osborne, and F. Verstraete, J. Math. Phys. **55**, 021902 (2014).

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See sections 2-5.
   (b) Did you describe the limitations of your work? [Yes] See section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the Impact statement.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See appendices A and C.
   (b) Did you include complete proofs of all theoretical results? [Yes] See appendices A and C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code used for the experiments is available in an online repository that will be made explicit after the review process.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figure 2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See mentions to Ref. [13].
   (b) Did you mention the license of the assets? [No] This is explicit in Ref. [13].
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of Observation 1

In order to rigorously prove that a complete characterization of gauge symmetries protects against revealing unintended information, we must first define some mathematical objects. For a fixed learning architecture (this is, a functional ansatz depending on n parameters), call $\mathcal{W} \in \mathbb{C}^n$ the set of all possible values of its parameters. The architecture, along with a point $\theta \in \mathcal{W}$, determines a model. Thus, we will refer to $\mathcal{W}$ as the set of white-box representations for a given architecture. Analogously, let the set of black-box representations $\mathcal{B}$ be the set of oracular functions (i.e., seen as input-output pairs) corresponding to each $\theta \in \mathcal{W}$. In general, there exists a function $\pi : \mathcal{W} \to \mathcal{B}$ that assigns every white-box representation to its corresponding black-box oracle. In certain cases one

can define a right inverse, $\alpha : \mathcal{B} \to \mathcal{W}$, that assigns a set of parameters to a black-box representation. This function satisfies $\pi \circ \alpha = \mathrm{id}_{\mathcal{B}\to\mathcal{B}}$, this is, that the oracular function associated to a white-box representation of a black box is the black box itself. Due to redundancies in the description, there can be many $\theta \in \mathcal{W}$ that lead to the same black-box representation, so in general, even if an α can be defined, it is not true that $\alpha \circ \pi = \mathrm{id}_{\mathcal{W}\to\mathcal{W}}$. This function, which takes all white boxes describing the same black box to the same element of $\mathcal{W}$, is what is commonly known as canonical form.

Also, we define an attack in a generic way, as a function applied on a white-box representation whose output provides information about features of the training data. Formally, we can model this as a (smooth) function $f : \mathcal{W} \to \mathbb{C}$ that maps the parameters of a model to a complex number.

With these, we can now restate the result and provide its proof:

Observation 1. *Consider the set of white-box representations for some architecture, $\mathcal{W} \in \mathbb{C}^n$, and its associated set of black-box representations, $\mathcal{B} = \pi(\mathcal{W})$. If a right inverse $\alpha : \pi \circ \alpha = id_{\mathcal{B}\to\mathcal{B}}$ can be defined, then for every function f*

$$f(\hat{\theta}) = \hat{f}[\pi(\hat{\theta})],$$

where $\hat{\theta} = \alpha \circ \pi(\theta) \in \mathcal{W}$ are the parameters denoting a canonical form for the model described by θ and $\hat{f} = f \circ \alpha$ is the application of f in black boxes.

Proof. By expanding $f(\hat{\theta})$:

$$\begin{aligned} f(\hat{\theta}) &= f \circ \alpha \circ \pi(\theta) \\ &= f \circ \alpha \circ \mathrm{id}_{\mathcal{B}\to\mathcal{B}} \circ \pi(\theta) \\ &= f \circ \alpha \circ \pi \circ \alpha \circ \pi(\theta) \\ &= \hat{f}[\pi(\hat{\theta})], \end{aligned}$$

where we have decomposed the identity in the space of black boxes as $\mathrm{id}_{\mathcal{B}\to\mathcal{B}} = \pi \circ \alpha$. □

B Matrix product states

The field of quantum many-body physics has been developing manageable ways of simulating the states and evolutions of quantum systems composed of many particles, which have recently gained attention in machine learning as alternative parametrizations of high-dimensional, convoluted functions. These are the so-called tensor network architectures and, notably within them, the matrix-product state (MPS) representations [5, 7]. In fact, these parametrizations of complex functions were later rediscovered in the field of numerical analysis, under the name of tensor trains [19, 20]. An MPS architecture is best defined when viewing functions as hyperplanes on high-dimensional feature maps of the input. This is, when we consider (vector) functions $f(\boldsymbol{x})$ of the input $\boldsymbol{x}$ as taking the form

$$f(\boldsymbol{x}) = M \cdot \Psi(\boldsymbol{x}). \tag{1}$$

MPS architectures correspond to the family of functions illustrated in Fig. 3. These are obtained when the vector feature map, $\Psi(\boldsymbol{x})$ (typically non-trainable), has a tensor-product form with one component per dimension of the input, $\Psi(\boldsymbol{x}) = \psi_1(x_1)\otimes\psi_2(x_2)\otimes\cdots\otimes\psi_N(x_N)$, and the hyperplane is expressed as a product of in general complex matrices (hence the name), namely

$$M^{\ell}_{s_1,s_2,\ldots,s_N} = \sum_{\{\alpha\}} A^{\alpha_1}_{s_1} A^{\alpha_1,\alpha_2}_{s_2} \cdots A^{\alpha_j,\alpha_{j+1}}_{\ell} \cdots A^{\alpha_N}_{s_N}. \tag{2}$$

These architectures have a very well-characterized gauge symmetry group, which characterizes the sets of parameters that describe the same function. If between every two consecutive matrices one inserts a decomposition of the identity $\mathbb{1} = Y_j \cdot Y_j^{-1}$ for an invertible matrix Y_j, the set of matrices given by $B_{s_j} = Y_j^{-1} \cdot A_{s_j} \cdot Y_{j+1}$ produce M as well. Importantly, it is well-known from quantum many-body physics that these are the only symmetries of the MPS architectures [21], and that it is possible to fix a value of the gauge for each MPS. This fixing is known as "choosing a canonical form", and it is generally obtained via singular value decompositions [21, 22].

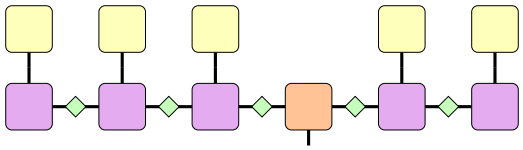


Figure 3: Depiction of a matrix product state classifier. Each element is a tensor with as many dimensions as legs. The purple squares in the lower row represent the parameters of the MPS. They are arranged in three-dimensional tensors, which are multiplied by their neighboring tensors and by the input. This is encoded in the one-dimensional vectors depicted by the yellow squares in the top row. The final tensor after multiplications via Eq. (2) is a vector encoding the output, because the bottom leg of the orange square in the bottom row is free. The gauge symmetry that allows to erase information about irrelevant features is the decomposition of the identity, represented by the green diamonds, in invertible matrices that are later absorbed by the original tensors.

C Proof of Theorem 2

In this section we give an explicit construction for the holomorphic map α in MPS architectures. Recall that MPS architectures process the input $\boldsymbol{x}$ via $f(\boldsymbol{x}) = M \cdot \Psi(\boldsymbol{x})$, where $\Psi(\boldsymbol{x})$ is a (typically non-trainable) feature map with a tensor-product form and M factorizes as per Eq. (2), with each entry of the A tensors in the left-hand side being a trainable parameter. Thus, when considering MPS architectures, the set $\mathcal{W}$ is a subset of $\mathbb{C}^n$ (where typically $n = Nb^2d$ for some number of sites N, bond dimension b, and physical dimension p, see Refs. [6, 23] for descriptions of these quantities and their connections with the simulation of quantum many-body systems) given by all the possible values of all the elements of the A tensors in the right-hand side of Eq. (2). The function π that maps a white-box representation of an MPS to its corresponding black box is, precisely, Eq. (2). Then, the set $\mathcal{B}$ is defined as $\pi(\mathcal{W})$, i.e., the subset of $\mathbb{C}^{n'}$ (with $n' = d^N$) defined by all allowed values of the entries of the M tensor in the left-hand side of Eq. (2).

In order to provide the map α, we will construct a specific canonical form for MPS. We want this canonical form to be (i) univocal, so all gauge-equivalent models are mapped to the same canonical form, (ii) holomorphic, and hence possessing the highest degree of smoothness, and (iii) global, so the exact same procedure can be applied to any point in $\mathcal{W}$ (with the potential exception of subsets of measure zero). We require these properties in order to ensure that at no point there are discontinuities in the construction of the canonical form that could be exploited by an attacker. Formally, the canonical form is an holomorphic map $\mathcal{W} \to \mathcal{W}$, transforming each MPS to its canonical form, so that all MPS related by a gauge transformation are mapped to the exactly same canonical-form MPS. This map then induces uniquely the desired $\alpha : \mathcal{B} \to \mathcal{W}$, which will also be holomorphic, as shown in the following proof.

Theorem 2. *For an MPS defined by a collection of tensors $\{A_{s_j}^{\beta_{j-1},\beta_j}\}_{j=1}^N$, a global, univocal, holomorphic canonical form $(\alpha \circ \pi) : \mathcal{W} \to \mathcal{W}$ is given by $\hat{A}_{s_1}^{\beta_1} = \mathbb{1}$, $\hat{A}_{s_N}^{\beta_{N-1}} = \mathbb{1}$, and $\hat{A}_{s_j}^{\beta_{j-1},\beta_j} = \sum_\gamma L_{s_j}^{\beta_{j-1},\gamma} C_{\gamma,\beta_j}$, where*

$$L_{s_j}^{\beta_{j-1},\gamma} = \sum_{\{\alpha\}} A_1^{\alpha_1} \cdots A_1^{\alpha_{j-3},\alpha_{j-2}} B_{\beta_{j-1},s_j,\gamma}^{\alpha_{j-2},\alpha_{j+1}} A_1^{\alpha_{j+1},\alpha_{j+2}} \cdots A_1^{\alpha_{N-1}},$$

$$B_{\beta_{j-1},s_j,\gamma}^{\alpha_{j-2},\alpha_{j+1}} = \sum_{\alpha_{j-1},\alpha_j} A_{\beta_{j-1}}^{\alpha_{j-2},\alpha_{j-1}} A_{s_j}^{\alpha_{j-1},\alpha_j} A_\gamma^{\alpha_j,\alpha_{j+1}},$$

$$C = L^{-1}, \text{ with } L_{\gamma,\beta_j} = L_\gamma^{1,\beta_j}.$$

This function induces a global, holomorphic map $\alpha : \mathcal{B} \to \mathcal{W}$.

Proof. The function $\{A\} \mapsto \{\hat{A}\}$ described is well defined whenever the L matrices are invertible. In such case, the only non-trivial operation is the computation of such inverses, which is an holomorphic operation in the elements of the matrix. The matrices L, at every step in the procedure, are just a projection on specific sites of the original MPS. While the projected states are not invertible in all MPS, the set for which this is not the case is an algebraic variety of dimension $< n$ (n being the total

number of parameters). This shows that the canonical form is global. Moreover, standard algebraic calculations lead to verifying that $\pi(\{\hat{A}\}) = \pi(\{A\})$. Note also that the canonical form obtained in this way will be the same for all MPS that are related by a gauge transformation, as required. This means that this canonical form indeed gives a well-defined map $\alpha : \mathcal{B} \to \mathcal{W}$ so that $\pi \circ \alpha$ is precisely the canonical form map $\{A\} \mapsto \{\hat{A}\}$.

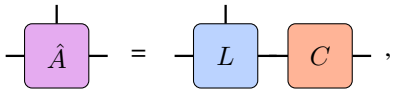
It thus remains to show that the α induced by the function $\alpha \circ \pi$ above is also holomorphic. In order for this statement to be meaningful, or even to define what it means for an attack on black boxes to be smooth, one needs to endow the set $\mathcal{B}$ with a smooth structure; more specifically, with a complex manifold structure. Note that $\mathcal{W}$ is trivially a complex manifold, since it is an open subset of some $\mathbb{C}^n$. There are a priori two ways to see $\mathcal{B}$ as a complex manifold. One is as a submanifold, generated by the π given by Eq. (2) as $\mathcal{B} = \pi(\mathcal{W})$ and embedded in the ambient (exponentially large) complex vector space of all possible input-output relations. The other is as the space of orbits defined by the gauge symmetries. This is, if we call $\mathrm{S}_{\mathrm{MPS}}$ to the complex Lie group of gauge symmetries, the space of orbits is nothing but the quotient set $\mathcal{W}/\mathrm{S}_{\mathrm{MPS}}$, where a natural complex manifold structure can be defined whenever the symmetry group action is reasonably good (holomorphic, free and proper). One can see, using well-known results in the theory of complex manifolds, that both ways to describe $\mathcal{B}$ are equivalent. Formally, the complex manifolds $\mathcal{W}/\mathrm{S}_{\mathrm{MPS}}$ and $\mathcal{B}$ are biholomorphic, and thus we write $\mathcal{B} = \mathcal{W}/\mathrm{S}_{\mathrm{MPS}}$ from now onwards. The analysis of these manifolds has been done in full detail for the set of "full-rank" MPS in Ref. [23], where we also refer to for the necessary definitions and background. Due to our extra condition of the matrices L being invertible, our sets $\mathcal{W}$ and $\mathcal{B}$ are (full measure) subsets of those considered in Ref. [23], but the exact same proof applies.

By the way the complex manifold structure is defined in the space of orbits $\mathcal{W}/\mathrm{S}_{\mathrm{MPS}}$, for any holomorphic map $\mathcal{W} \to \mathcal{W}$ that is invariant under the action of the gauge group $\mathrm{S}_{\mathrm{MPS}}$ –as it is the case for the canonical form defined– the uniquely defined associated map $\alpha : \mathcal{B} = \mathcal{W}/\mathrm{S}_{\mathrm{MPS}} \to \mathcal{W}$ is also holomorphic. All details can also be found in Ref. [23] and the references therein. □

Theorem 2 assumes, a priori, that the dimension of all legs are the same. However, it is easy to obtain decompositions with different dimensions, by using projectors (i.e., the sets of subindices in the A matrices) different than the one considered.

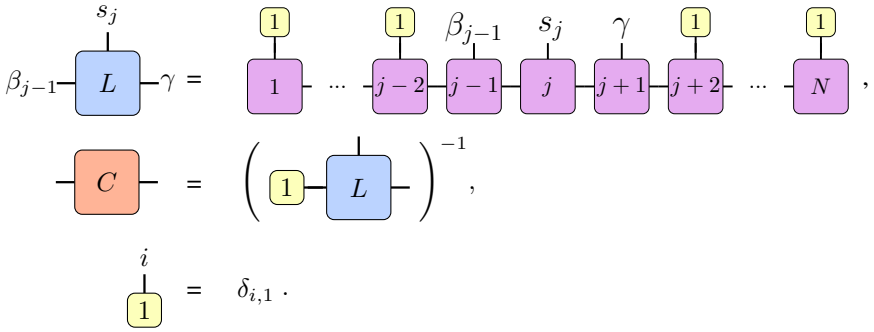
D Graphical description of the canonical form

Following the usual graphical notation for tensor networks (for its definition see, for instance, the explanations in Ref. [6]), the decomposition described in Theorem 2 is



(3)

with



E Dataset and training of models

To illustrate the fact that neural networks are vulnerable to irrelevant feature leaks and MPS are not, we have trained both architectures in a real-world dataset. This is part of the

`global.health` database of COVID-19 cases [13]. This is a very large database that allows us to make small partitions, so that we can train shadow models when illustrating the attacks. We take the dataset available on March 22nd 2021, use the data of two countries, Argentina and Colombia, for generating our database. The database built only contains the columns `location.country`, `events.outcome.value`, `events.confirmed.date`, `demographics.ageRange.start`, `demographics.gender`, and `symptoms.status` from the whole dataset, and all datapoints where at least one of these entries is empty are discarded. As a first balancing between countries, we take all entries for Argentina and, evenly, one out of every seven (which is approximately the ratio between the number of datapoints for each country) points for Colombia's cases where the column `events.outcome.value` takes the value `Death`, and the same amount for the cases where `events.outcome.value` takes any other value. As a second balancing, now between cases with odd and even registration dates, we take all cases where the column `events.outcome.value` takes the value `Death` (a total of 21 503), and the first 5 375 cases for each combination of country and parity from the subset of points where the column `events.outcome.value` takes the value `Recovered`.

The classification task in which both, neural networks and MPS, are trained, is in predicting the value for `events.outcome.value` when provided the rest. While all features are discrete in nature, we treat the age feature as continuous. As irrelevant feature, we choose the parity of the report date, extracted from `events.confirmed.date`. While this quantity is indeed expected to be irrelevant for the classification, we check that it is sufficiently uncorrelated with the remaining features (see Table 1).

Table 1: Pearson correlation coefficients between the columns of the dataset used for training the neural network and MPS models. The columns in the table correspond to the columns (in order) `events.confirmed.date`, `location.country`, `demographics.ageRange.start`, `demographics.gender`, `symptoms.status`, and `events.outcome.value` of the original database.

	parity	country	age	gender	symptoms	recovery
parity	1	0.005	0.001	0.001	−0.012	−0.002
country		1	0.059	0.033	0.161	−0.055
age			1	−0.010	0.128	−0.707
gender				1	0.004	0.078
symptoms					1	−0.143
recovery						1

The neural network model used consists of a five-layer architecture with structure 16-16-8-4-2, totalling 614 trainable parameters. Each intermediate layer has a rectified linear unit as activation. Training optimizes the cross-entropy between the predictions and the labels in batches of 8 datapoints, using the Adam optimizer with learning rate of $3 \cdot 10^{-4}$ and ℓ_2 regularization of magnitude $6 \cdot 10^{-3}$, for at most 1 250 epochs. The final model picked is that along the training history that achieves higher accuracy in a held-out validation set composed of 5 000 samples of the original database.

The matrix product state model is depicted in Fig. 3 and consist of six tensors where all the dimensions have cardinality 2. This is, the leftmost and rightmost purple squares in Fig. 3 are 2×2 matrices, and the remaining purple squares and the orange square are tensors of dimensions $2 \times 2 \times 2$. All entries in all tensors (a total of 40) are free, trainable parameters. Training optimizes the cross-entropy between the predictions and the labels in batches of 100 datapoints, using the Adam optimizer with learning rate of 10^{-1}, for 20 epochs.

The encoding of categorical variables in the input is different for both architectures. For neural networks we perform a traditional one-hot encoding. In contrast, for MPS we perform a noisy encoding, in such a way that for each datapoint the class is encoded in a random value in either $\left[0, \frac{1}{2} - \epsilon\right]$ or $\left[\frac{1}{2} + \epsilon, 1\right]$, with $\epsilon = 5 \cdot 10^{-2}$. Then, every dimension of the input is encoded in a different two-dimensional vector via $\psi(x) = (1 - x, x)$, and these are the objects that are input to the MPS (the yellow squares in Fig. 3). For the `demographics.ageRange.start` column, a min-max normalization is performed before producing the input vector. The different encoding of the variables for neural networks and MPS does not have severe impacts in the vulnerability of the models: as demonstrated in Fig. 2b, the MPS architectures are still vulnerable if no protection is applied.

Notably, the function computing the canonical form that is used in the experiments is the standard one based on singular value decompositions. As we commented in Section 4, it is left for future work to analyse whether the SVD-based canonical form fulfills similar regularity properties as the new one we construct here for the proof of Theorem 2. In any case, the analysis on Fig. 2 indicates that, in practice, the canonical form obtained via SVD guarantees a significant degree of privacy.

F Attacks

The attacks we consider are in the spirit of shadow modeling attacks [1, 14], whereby the attacker is provided with a large collection of models trained on datasets that share the statistics of the dataset where the victim model has been trained on. This is a situation where the attacker has much more power than what is realistic, so the results are upper bounds to realistic attacks.

In order to produce Fig. 2b, we generate, for every percentage of the majority class in the irrelevant variable, a total of 200 datasets (100 for each majority class) with that proportion, randomly sampled from the original dataset. On each of these datasets, a total of 100 models are trained according to the prescription described in Appendix E. This produces, for each of the percentages (the horizontal axis of Fig. 2b), 20 000 trained models. Out of all of them, those corresponding to 80 of the datasets are provided to the attacker along with their corresponding majority class, while the models for the remaining 20 datasets will be those to be attacked.

The attacker, depending on the type of models, does a different kind of attack: for neural networks the attack consists of a logistic regressor over the full set of models' weights and biases after a proper normalization, with ℓ_2 regularization and using the LBFGS solver. This attack is arguably simple, but this only reinforces the argument that neural networks are very vulnerable to leaking the nature of irrelevant features. Removing parameters and keeping those in the initial layer, or doing PCA to reduce dimensionality, did not offer advantages to using the full set of model parameters as input.

For MPS architectures, the attacker trains a deep feedforward neural network that is input the model weights (after a proper normalization) and outputs the corresponding label. This was done to provide the attack with more generality, so it could hopefully extract more information from the parameters of the model. The attack neural network consists of six layers with structure 20-20-10-10-2-2, using rectified linear units as activation functions for the intermediate layers. Training optimizes the cross-entropy between the predictions and the labels in batches of 1 000 datapoints, using the Adam optimizer with learning rate of 10^{-3} and ℓ_2 regularization of magnitude 10^{-4}, for at most 1 000 epochs. The set of 80 datasets is split randomly in an 80-20 proportion as to generate a validation set for early stopping.

In order to provide statistics, we train and evaluate the attacks for several random choices of the 80 datasets given to the attacker. We do a total of 1 000 of these repetitions, which provide the confidence intervals in Fig. 2.