Protecting ownership rights of ML models using watermarking in the light of adversarial attacks

Katarzyna Kapusta, Lucas Mattioli, Boussad Addad, Mohammed Lansari

Protecting ownership rights of ML models using watermarking in the light of adversarial attacks

Katarzyna KAPUSTA[1,2], **Lucas MATTIOLI**[2], **Boussad ADDAD**[1,2], **Mohammed LANSARI**[1]

[1] Thales SIX GTS France, [2] IRT SystemX France

Abstract

In this paper, we present and analyze two novel - and seemingly distant - research trends in Machine Learning: ML watermarking and adversarial patches. First, we show how ML watermarking uses specially crafted inputs to provide a proof of model ownership. Second, we demonstrate how an attacker can craft adversarial samples in order to trigger an abnormal behavior in a model and thus perform an ambiguity attack on ML watermarking. Finally, we describe three countermeasures that could be applied in order to prevent ambiguity attacks. We illustrate our works using the example of a binary classification model for welding inspection.

Introduction

The value of a Machine Learning model will first and foremost depend on the quality of its training dataset. Expert knowledge of the data scientists involved in its creation plays then a second but also important role. Last but not least factor are the computational resources required to run the training, which for some particular use cases may be staggering (training GPT-3 would cost over $4.6M using a Tesla V100 cloud instance[1]).

Attackers unable to gather data or perform the costly training, may be tempted to steal an existing model. They may then use it in secret for their own purposes or monetize it by making it available as a service. In the latter case, the legitimate owner will discover the theft and will claim the ownership of the model. However, providing a proof of its ownership may be quite tricky as model architectures or parameters may be changed by the attacker (while its performance will not be significantly impacted). Moreover, the attacker will most probably not be willing to reveal them if not forced by law.

ML models watermarking is a rapidly developing research field, which main goal is to help identify the ownership of a model. Inspired by the watermarking in the multimedia domain, it consists in inserting a secret change into a model during its training that once revealed by the model owner will allow the model identification. This secret change is the ML model watermark and can have the form of a modification of the model look (typically models parameters) or model behaviour. Verification of model ownership relies then on a demonstration of the knowledge of this secret and abnormal change by the model creator, ex. using a secret key to extract the watermark embedded into model parameters or querying the model with specially prepared inputs that will trigger the abnormal behaviour.

Independently, adversarial attacks became one of the main cybersecurity threats to the trustworthy AI. It was demonstrated that even a slight - but specially crafted - perturbation to the model inputs can result in an erroneous prediction. Thus, an attacker may force the model misbehaviour without even touching neither the training data nor its internals. While the main danger of adversarial attacks is their ability to compromise the trust in a model (Akhtar and Mian 2018), they can also be leveraged to produce counterfeit ML watermarks. To our knowledge, this application of adversarial attacks remains little studied.

Our first objective is to analyze ML watermarking in terms of its impact on the model performance and robustness against removal erasure attack. We focus on the behavioral ML watermarking as it allows the identification of a model in a MLaaS (ML as a Service) setting and therefore seems to be the most practical of watermarking techniques. Our second goal is to demonstrate how adversarial attacks can be leveraged to create counterfeit watermarks. Finally, we propose solutions to the identified problem.

Outline We start with introducing relevant work from the domain of ML watermarking and adversarial attacks. Then, we present our preliminary results on watermarking using public datasets. We analyze ML watermarking applied on an industrial use case. Finally, we show how to perform an ambiguity attack on ML watermarking using adversarial patches and describe possible solutions to that problem. We conclude with insights into future work.

Relevant work

ML watermarking

ML watermarking, first introduced in (Uchida et al. 2017), aims at providing intellectual property protection of a vendor by embedding a proof of ownership into the model. This proof should remain secret to anyone but the model owner and revealed only during verification of a suspicious model. It also has to be robust to potential model modifications,

[1] https://lambdalabs.com/blog/demystifying-gpt-3

such as fine-tuning or pruning. In recent years, the development of Ml watermarking was fueled by the appearance of extraction attacks, capable of copying models deployed as a MLaaS. The big majority of ML watermarking techniques focuses on the image classification task.

ML watermarking techniques can be roughly divided into two main categories - white-box and black-box. In white-box watermarking techniques the proof of ownership relies on secret change embedded into the model parameters (Uchida et al. 2017) or the model architecture (Fan, Ng, and Chan 2019). Thus, they will allow verification of the model only if the verification authority (the legitimate owner of the model or a trusted third-party) has access to the whole concerned model.

In black-box watermarking, the proof of ownership is based on a secret change embedded into the model behavior (Adi et al. 2018; Zhang et al. 2018). The method can be compared to legitimate model backdooring. In backdooring, an attacker poisons the training data in order to create a hidden modification to the model behaviour: when triggered with the poisonous data, the model will deviate from its normal behavior. In the case of watermarking, its the model owner who in purpose introduces the modification to his own model. Compared to white-box watermarking, the technique is more practical as it requires only access to the model's API and not its whole internals.

In more detail, black-box watermarking uses the overparametrization of neural networks. It modifies their behavior by poisoning the dataset with a set of data samples denoted as *key inputs* that are regrouped into a *trigger set*. There are three main approaches to the generation of the key inputs and their associated labels (illustrated in Figure 1). They were all conceived to be used for image classification models. In the first strategy, the model creator chooses a number of samples from the dataset as the trigger set and embeds onto these samples a meaningful content, such as text or logo. Then, they change the labels of the modified samples. The second strategy consists in constructing the trigger set from unrelated data labeled as a selected class (or multiple random classes). The last strategy is a variation on the content method, where noise is embedded within the key inputs.

An efficient black-box watermarking should meet several requirements, such as being resistant to model transformations that could lead to watermarks erasure or being clearly tied to owner's identity in order to avoid confusion (Kapusta, Thouvenot, and Bettan 2020). Moreover, their impact on the network performance should be ideally negligible. During the verification process, the key inputs will be necessarily revealed and thus watermarks are hardly reusable. Therefore, it is important to embed a sufficient amount of watermarks into the model in order to allow multiple verification checks (usually a trigger set with dozens of key inputs is used).

Attacks on watermarking

We identify three main categories of attacks against watermarking (Kapusta, Thouvenot, and Bettan 2020; Wang and Kerschbaum 2019):

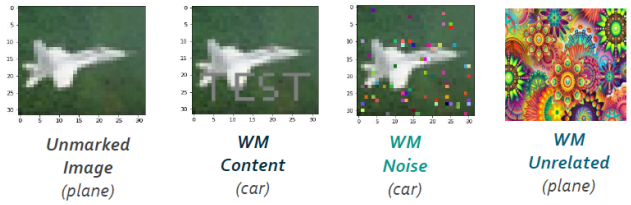

Figure 1: Illustration of the three main black-box watermarking techniques present in the literature. From left to right: clean sample from the dataset (correctly labeled image), **content** watermark, **noise** watermark, and **unrelated** watermark.

Watermark removal : an attacker removes completely the watermark from the model or weakens its strength. More precisely, they use techniques such as fine-tuning, compression or fine-pruning, which can weaken the strength of the marks. In case of black-box watermarking, techniques used for backdoor detection and removal can also serve to erase watermarks. Moreover, any transformation (intentional or not) of the network may potentially have an impact on the watermarks.

Ambuiguity attack : an attacker casts doubt on the legitimate ownership by providing counterfeit watermarks. The easiest way to create confusion is to insert a different set of watermarks into an already watermarked model.

Evasion attack : an attacker tries to escape watermarks verification in the context of black-box techniques. The evasion can be achieved using a query detector inspecting if a query is a possible verification attempt (in this case, a random prediction will be output).

Adversarial attacks

Machine learning models, including neural networks, have been shown to exhibit non robust behavior. They are vulnerable to evasion attacks aka adversarial examples (Goodfellow et al. 2020): they misclassify specially crafted inputs that are very similar to correctly classified ones. Indeed, images that differ only in a few pixels may be classified differently. Although invisible adversarial examples (very slightly modified images, achieved by adding specially crafted noise to a clean sample) have received strong attention from the computer vision community (Szegedy et al. 2013; Biggio et al. 2013; Kurakin, Goodfellow, and Bengio 2017; Madry et al. 2018; Cohen, Rosenfeld, and Kolter 2019), Nguyen, Yosinski, and Clune (2015) and Brown et al. (2017) have managed to introduce a new threat. Named adversarial patches, it consists in modification of a single, small region of the input image, able to strongly perturb the behavior of a deep network (Figure 2) (Brown et al. 2017).

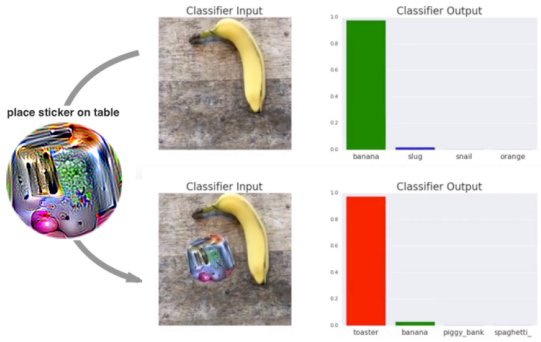

Figure 2: Figure from (Brown et al. 2017): a toaster patch is added to a banana image leading the targeted network to predict toaster instead of banana.

An important feature of adversarial patches is that they can be created in the physical world. Contrary to invisible adversarial examples, which rely on exact modification at the pixel level, patch attacks rely on a very salient modification which can be more easily constrained to be physically possible. For instance, attackers may craft an adversarial patch in form of a sticker that, once put on an physical object, will mislead cameras with embedded AI for images recognition.

Motivations

ML watermarking is a rapidly developing research trend. Our first objective was to test and evaluate the most relevant watermarking techniques on a real-world use case. We have chose black-box watermarking aka watermarking from backdooring for our tests, as it seems to work in the most realistic scenario, where the access to the suspected model is limited to its inputs/outputs. We wanted to answer the three following questions: Are the watermarks well detectable? What is the impact of watermarking on the model's accuracy? Is it possible to remove the watermarking using fine-tuning?

The second goal was to explore the similarities between watermarking and adversarial patches. We have observed that, although the mechanisms behind the two techniques are quite different, they give similar results: a specially crafted input triggers abnormal behavior of the model. Therefore, the following questions arises: Does an attacker can leverage adversarial attacks to attack watermarking? Is it easy to craft a counterfeit watermark that is in reality an adversarial patch?

Preliminary results on public datasets

The three black-box watermarking techniques mentioned in the state-of-the-art section were implemented and tested. Preliminary tests were realized on the CIFAR-10 dataset [2]. As watermarking content and noise are similar (watermarking noise can be seen as a variant of the content technique), the focus was put on the comparison between the baseline model and models watermarked with content or unrelated techniques.

Results presented in the Table 1 show that the watermarks are clearly detectable after being embedded during training. The accuracy on the trigger sets for both content and unrelated techniques reaches 100%. When testing the trigger sets on a non-marked model or on a model marked with a different watermarking technique, the accuracy drops significantly.

The impact of watermarking on the model accuracy is shown in Figure 3. Although at the beginning of the training there are some differences between the baseline and the three marked models, the final accuracy does not significantly differ. The slightly better accuracy of the model marked with the content technique can be at first surprising, but it comes from the fact that initial watermarks where included in the test dataset: the model learns easily to recognize them without mistake (it has 100% accuracy on these images) and therefore it increases the overall accuracy.

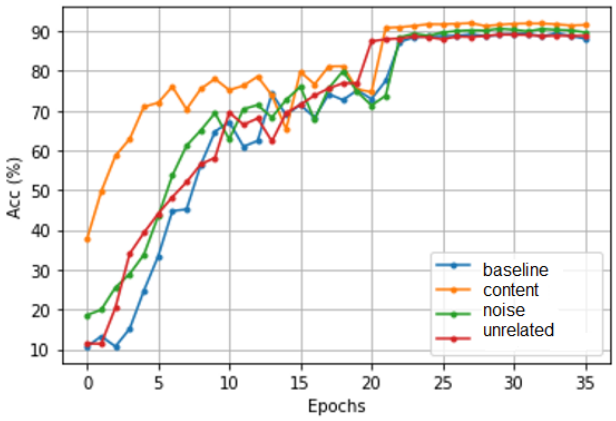

Figure 3: Comparison of the three main watermarking approaches during training.

A different test aimed at investigating the noise watermarking technique. In more detail, the amount of the generated noise used to mark the key inputs of the watermarks was varied and the accuracy of the watermark detection was measured (see illustration in Figure 4 and comparison in Table 2). It shows that clearly detectable watermarking can be achieved with 17% (170 pixels) of noise perturbation introduced in the key inputs.

Results on the Welding Inspection use case

We implemented and tested the three same black-box watermarking techniques on a binary classification dataset for welding inspection of automobile parts (Braunschweig, Gelin, and Terrier 2022) (containing two classes: "Normal" and "Retouch"). This dataset was provided by the French automobile industrial Renault within the Confiance.ai program (Confiance.ai et al. 2022). For each of them, we compared the performances of the corresponding watermarked model with a baseline model of the same architecture: a Resnet model composed of 4 hidden layers that we will refer to as

[2] https://www.cs.toronto.edu/kriz/cifar.html

Table 1: Watermarking accuracy for the content and unrelated watermarking approaches (100 key inputs in the trigger dataset).

Model	Dataset	Trigger set: content	Trigger set: unrelated
model_baseline	91.21%	4%	8%
model_content_100	91.57%	100%	12%
model_unrelated_100	89.2%	11%	93%

Table 2: Watermarking accuracy for the noise watermarking technique in function of the noise strength (measured as the number of pixels changed in a 32x32 image).

Number of pixels changed	100px	170px	256px	500px	1000px
Accuracy	44%	96%	100%	100%	100%

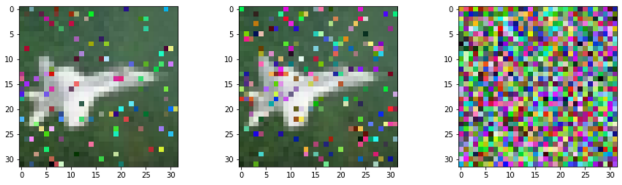

Figure 4: Varying the amount of noise in the watermarking noise approach by changing pixels to random values. From left to right: 100 pixels changed, 170 pixels changed, 1000 pixels changed .

"baseline_model" in this document. We use the same training procedure for each watermarking technique, as for the baseline_model. It consist of 50 epochs with a SGD optimizer (Stochastic Gradient Descent with momentum) with a starting learning rate of 0.001. The only difference between the marked and the baseline model will be in the presence of various trigger sets during training.

Unrelated watermarking consists in injecting unrelated data to the original dataset during training. For a classification problem, the idea is to create a trigger set that contains a small collection of pairs of (key input, label) unrelated to the training dataset and to train the model with both the training set and the trigger set. The unrelated key inputs can be all labelled as one class or have randomly associated labels.

Then, during test time, each key input of the trigger set will be evaluated. In the perfect case, the model should be able to correctly identify each key input of the trigger set correctly (i.e. to its corresponding class).

The strength of the proof of ownership depends on the statistical probability that a random model (i.e. not specifically trained on the trigger set) would correctly classify each key input of the trigger set correctly. For a classification problem with M different classes and for a trigger set of N elements: the probability of a random model to correctly identify all N data points is $\frac{1}{N^M}$. Thus the strength of the ownership proof depends both on the number of classes of the dataset and the size of the trigger set. Here we tested these watermarking techniques on a binary classification problem thus containing the smallest possible number of classes for a classification problem.

We tested various sizes of the trigger set to evaluate the unrelated watermarking technique's **strength** and proof **capacity** for our model, and to compare the nominal performances of our watermarked model to the baseline model. The proof capacity corresponds to the number of simultaneous watermarks added to a single model. As each watermark can only be used once because using it as a proof of ownership requires to expose it thus trivializing it's removal.

We hypothesized that the nominal performance should slightly decrease as we increase the size of the trigger set. For our evaluation, we used pictures from an open-source dataset: the Stanford-AI Cars dataset (Krause et al. 2013). It is composed of 16 000 labeled cars images. We created two different trigger sets, one with only 10 car images and one with 100 images to evaluate the impact of the size of the trigger set during the watermarking of the model.

In our first exploration tests, we were able to retrieve the watermark completely for the trigger set of size 10, without impacting the model's performances. However, for the trigger set of size 100, we were never able to completely recover the watermark (the best model had a 76% accuracy for the watermark recognition). This approach should be investigated further as it isn't clear whether it is a hard limitation.

Content watermarking consist in adding a semantic information into a sample of the original dataset, and to train the model into recognizing the semantic information and to classify it in a predefined class. In our work we choose the logo depicted in Figure 5 as our semantic information. Other studies have also used text written directly on the images. We choose to always add it on the top left corner in order to be consistent with its placement, to help the network recognize the watermark. Again, we tested two sizes for the trigger set: one with 10 images and another with 100 images. For this watermarking method, we were able to incorporate the watermark without impacting the model's performances — compared to the "baseline_model", see Table 4. Further work could focus on adding multiple content watermarks to a single model in order to evaluate the proof capacity of such

Table 3: Nominal result of un-watermarked model (baseline)

Network	**Metrics**			
	Accuracy	Precision	Recall	F1 score
model_baseline	88.08	86.82	99.12	92.56

Figure 5: Logo used for content watermark.

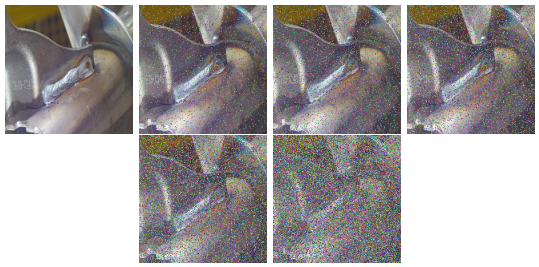

Figure 6: Varying noise in the noise watermarking technique.

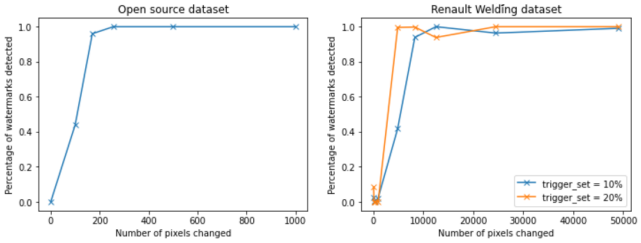

Figure 7: Comparison of noise watermark recognition accuracy based on the number of modified pixels for two different sizes of the dataset.

approach in the context of an industrial use case.

Noise watermarking is similar to the content method. It consists in adding a specific perturbation to the key inputs that are samples taken from the dataset, to make the model learn the association between this perturbation and a predefined class. Opposite to the content method, the added information does not contain any meaningful semantic information, but uses a random or key-generated noise. More precisely, for a data recognition problem, a fixed set of pixels will be changed to correspond to a specific random pattern. For each image of the trigger set the same noise is applied.

For the noise watermarking method, we tested various amount of noise to embed into the key inputs of the trigger set. We expected that, the more noise was added, the more identifiable the watermark would become (as it was the case for the preliminary results on the public dataset). On the other hand, adding too much noise could potentially decrease the performances of the model. Moreover, if the noise becomes too important and covers the majority of each picture of the trigger set, then, each key input of the trigger set will become approximately the same for the model. It would greatly reduce the strength of the ownership proof. In order to avoid that phenomenon: the images used for the trigger set should still contain some semantic information from the original images.

We chose the amount of noise according to previous studies summarized in Table 2 and decided to evaluate the watermark recognition accuracy for the same absolute total information of noise and the same relative information of noise. As the previous study was done on 32×32 images and we are evaluating it on 224×224 images we evaluated the watermark for the following number of modified pixels: {100, 170, 256, 500, 1 000, 4 900, 8 330, 12 540, 24 500, 49 000}.

We were able to reproduce the previous results obtained on the public dataset of 32×32 images for the same relative amount of modified data used for the noise watermark. Thus confirming that a minimal quantity of noise is necessary to be able to recognize the watermark. We also evaluated the influence of the size of the trigger set over the watermark recognition as we hypothesized that the size of the trigger set could have an influence over the recognition of the watermark. Figure 7 shows that a larger trigger set can result in a greater recognition of the watermark for some amount of noise included (here: 4 900 and 8 330 pixels changed).

On the other hand, we also analyzed whether the addition of the noise watermark was any detrimental to the model's nominal performances. We observed that for a small noise the training performances of the model were completely reduced. For each watermark added with less than 4 900 pixels modified, the networks were not able to satisfyingly classify the welding pictures (see Figure 8).

We interpret this result as the fact that the watermark creates a competition for the classification of originally correct weld pictures (classified as "OK"). Thus, when we only add a small noise, it is more difficult for the model to differentiate between noisy and non-noisy images (watermarked and un-watermarked images). On the other hand, when the noise is more important, the model can more easily learn the noise pattern rather than the semantic information presented in the weld pictures used for the trigger set.

Table 4: Comparison of nominal performances for content watermark.

Network	Metrics			
	Accuracy	Precision	Recall	F1 score
model_baseline	88.08	86.82	99.12	92.56
model_content_10	88.74	86.92	100	93.00
model_content_100	89.40	88.80	98.23	93.28

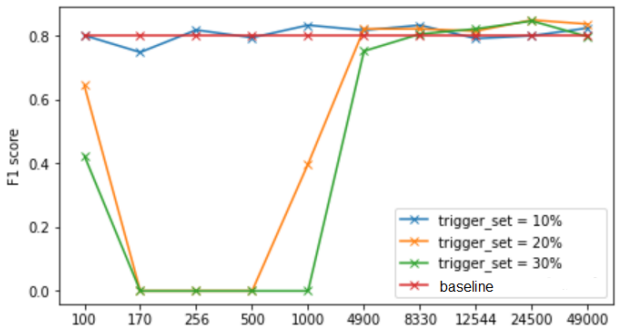

Figure 8: Comparison of nominal performance for noise watermark based on the number of modified pixels for different sizes of the trigger set.

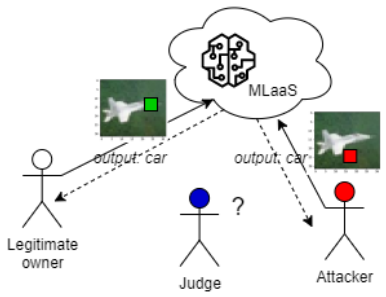

Figure 9: Ambiguity attack illustration: an attacker constructs an adversarial sample (with red rectangle used as an adversarial patch) that imitates the content watermark.

Attacking watermarking

In the section, we attack watermarking. First, we try a removal attack based on fine-tuning. Second, we show an ambiguity attack that can lead to confusion during watermarks verification (see Section on the attacks on watermarking for both attacks explanation).

Fine-tuning attack

We evaluated the impact of various number of fine-tuning epochs over a watermarked model. As with gentle fine-tuning we could not achieve the removal, we chose a more aggressive strategy. We chose a learning rate of 10^{-5} for 10, 15 and 20 epochs. With this procedure we again where not able to remove the watermark as all the test data used for watermark recognition were still correctly classified by the model, even though we were reducing the model nominal performances significantly (see Table 5).

	Accuracy	**F1 Score**	**WM recognition**
Before fine-tuning	93.23%	85.02%	100.00%
After 10 epochs	92.57%	82.82%	100.00%
After 15 epochs	91.05%	78.53%	100.00%
After 20 epochs	89.96%	74.99%	100.00%

Table 5: Evolution of model's performances during the fine-tuning attack.

Ambiguity attack using adversarial patches

An adversarial attack using the patch method can be very similar to the *content* watermarking technique (illustration in Figure 9). As a result, the legitimate owner may try to claim its property by revealing the abnormal behavior of its model and may end up being confronted with the attacker who will demonstrate another abnormal behavior of the same model.

We tested different types of perturbations that could be used as an adversarial patch and thus as a counterfeit watermark. We analyzed then their impact on model performance. A successful adversarial patch will make the model misclassify the input with embedded perturbation. In order to assess the effect of the patch size on the attacks success rate against the models, three patch sizes were considered: 8×8, 16×16, and 32×32 pixels. Also, in order to check if the degraded performance of the model is not merely caused by the presence of a perturbation, whatever is its type, we considered non optimized patches, simple stickers, generated randomly or simply one-color black or grey squares (illustrated in Figure 10). The results are given in Table 6.

As presented in Table 6, even with a simple non optimized patch, the model performance drops with 60% or even 72% when using a 32×32 patch on data from class "Retouch". The grey and black patches are not that successful on data from class "Normal" but the random patches reach 35% misclassification on these data. The optimized patch does, with no surprise, much better than that (62% success rate) with 32x32 size but astonishingly less good results with the smaller patches 16x16 or 8x8. It is an interesting results that must be investigated more thoroughly in the future.

In order to get an idea about the different patches placement on the images and their effect on the classification, some samples are given in Figure 10 while both classes, "Normal" and "Retouch", are considered in evaluation.

ML Patch	Patch size 32x32	Patch size 16x16	Patch size 8x8
Normal	62.90%	15.58%	7.34%
Retouch	92.89%	27.23%	8.02%
Black patch	**Patch size 32x32**	**Patch size 16x16**	**Patch size 8x8**
Normal	6.45%	9.62%	4.96%
Retouch	72.79%	31.83%	8.55%
Grey patch	**Patch size 32x32**	**Patch size 16x16**	**Patch size 8x8**
Normal	8.23%	10.62%	4.66%
Retouch	67.13%	22.49%	5.56%
Rdm patch	**Patch size 32x32**	**Patch size 16x16**	**Patch size 8x8**
Normal	34.62%	12.10%	8.04%
Retouch	59.85%	64.14%	49.70%
Rdm patch	**Patch size 32x32**	**Patch size 16x16**	**Patch size 8x8**
Normal	33.63%	12.70%	6.75%
Retouch	60.58%	65.10%	50.38%

Table 6: Patch attacks (first row) and simple perturbations (following four rows) effect on welding UC ML model performance.

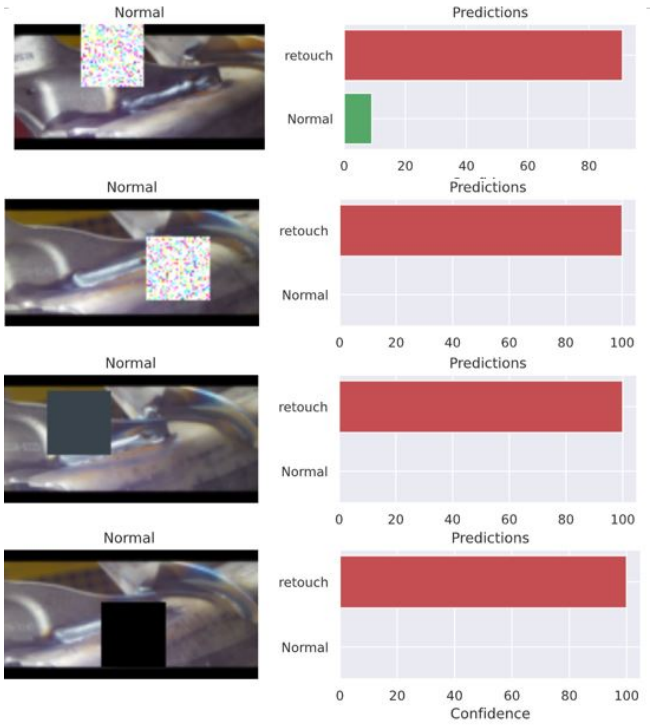

Figure 10: Samples of attacks results using different types of patches.

How to prevent ambiguity attacks?

In its basic version, ML watermarking enables ML owners to verify if their models were stolen. However, if used naively, it may not provide a sufficient proof of ownership of a model in a situation where the model was indeed stolen, as an attacker may retrain the model in order to embed its own watermarks. Even worse, as presented in the previous section, adversarial attacks can be used by attackers to claim the ownership of models that were not even stolen. Therefore, in order to be integrated within real-world use cases, watermarking should be reinforced with the right defense mechanisms against false ownership claims.

Combining black-box with white-box

In (Fan, Ng, and Chan 2019), a countermeasure against ambiguity attacks was proposed that combines the black-box approach with an additional white-box model marking. Model performance strongly depends on specially crafted model layers, knowledge of which provides the final proof of model ownership. More generally, using white-box technique as a next step after black-box verification reinforces the ownership proof. However, as it implicates the model revealing to a sort of verification authority, its use may be costly and therefore should be limited.

Reinforcing the link between the watermark and the model owner

A naive black-box watermarking technique does not establish any link between the owner and watermarks. More sophisticated techniques embed owner's logos into the key inputs or use unique encoders to generate the watermarks (Li et al. 2019). Such watermarks should be harder - but not impossible - to imitate by an attacker. They can be coupled with explainability methods that allow to better understand the effect of watermarking on the marked models and, in consequence, to design more explicit watermarks (Lansari, Kapusta, and Thouvenot 2022).

The right verification protocol

In (Adi et al. 2018), authors proposed to register the watermarks in a secure cryptographic vault and reveal them only at time of the verification. This prevents ambiguity attacks, where an attacker manages to find abnormal behavior in the concerned model. It does not however eliminate totally the risk of sophisticated attacks, where an attacker will use the transferability properties of adversarial attacks to create its counterfeit watermarks in advance. A secure protocol for verification of ML watermarks was proposed in (Kapusta et al. 2021). Its first step consists of verification if the claimer of a model has the ability of proving that they possesses both a non-marked model and a watermarked one. If implemented wisely, this procedure will significantly reduce the success rate of ambiguity attacks.

Conclusion

In this paper we presented three black-box watermarking techniques from the literature. We evaluated them on an industrial dataset for a binary classification problem. We tested

their impact on the model performance, as well as their robustness to the fine-tuning removal attack. We also demonstrated how adversarial patches could be used to create ambiguity attacks for ML watermarking. Finally, we gave recommendations on how to avoid naive ambiguity attacks. These methods and tools are planed to be integrated in the trustworthy-AI engineering workbench provided by Confiance.ai. In the same program, future works are expected to extend the proposed approach from image classification to image detection.

Acknowledgments

This work has been supported by the French government under the "France 2030" program, as part of the SystemX Technological Research Institute within the Confiance.ai Program (www.confiance.ai).

References

Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; and Keshet, J. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. *In 27th USENIX Security Symposium (USENIX Security 18)*.

Akhtar, N.; and Mian, A. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6: 14410–14430.

Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Šrndić, N.; Laskov, P.; Giacinto, G.; and Roli, F. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, 387–402. Springer.

Braunschweig, B.; Gelin, R.; and Terrier, F. 2022. The wall of safety for AI: approaches in the confiance. ai program. In *Workshop on Artificial Intelligence Safety (SAFEAI)*.

Brown, T. B.; Mané, D.; Roy, A.; Abadi, M.; and Gilmer, J. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665*.

Cohen, J.; Rosenfeld, E.; and Kolter, Z. 2019. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 1310–1320. PMLR.

Confiance.ai; et al. 2022. Towards the engineering of trustworthy AI applications for critical systems - The Confiance.ai program.

Fan, L.; Ng, K. W.; and Chan, C. S. 2019. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in neural information processing systems*, 32.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144.

Kapusta, K.; Thouvenot, V.; and Bettan, O. 2020. Watermarking at the service of intellectualproperty rights of ML models. *Conference on Artificial Intelligence for Defense (CAID)*.

Kapusta, K.; Thouvenot, V.; Bettan, O.; Beguinet, H.; and 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*, IHMMSec '21, 171–176. New York, NY, USA: Association for Computing Machinery. ISBN 9781450382953.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3D Object Representations for Fine-Grained Categorization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2017. Adversarial Machine Learning at Scale.

Lansari, M.; Kapusta, K.; and Thouvenot, V. 2022. How to efficiently and explicitly watermark your Convolutional Neural Network. In *Conference on Artificial Intelligence for Defense*, Actes de la 4ème Conference on Artificial Intelligence for Defense (CAID 2022). Rennes, France: DGA Maîtrise de l'Information.

Li, Z.; Hu, C.; Zhang, Y.; and 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*, ACSAC '19, 126–137. New York, NY, USA: Association for Computing Machinery. ISBN 9781450376280.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Nguyen, A.; Yosinski, J.; and Clune, J. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 427–436.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Uchida, Y.; Nagai, Y.; Sakazawa, S.; and Satoh, S. 2017. Embedding watermarks into deep neural networks. *In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*.

Wang, T.; and Kerschbaum, F. 2019. Attacks on digital watermarks for deep neural networks. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M. P.; Huang, H.; and 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*.