A Black-Box Watermarking Modulation for Object Detection Models

Mohammed Lansari, Lucas Mattioli, Boussad Addad, Paul-Marie Raffi, Katarzyna Kapusta, Martin Gonzalez, Mohamed Ibn Khedher

A Black-Box Watermarking Modulation for Object Detection Models

Mohammed Lansari[1,2*], Lucas Mattioli[3*], Boussad Addad[1,3*], Paul-Marie Raffi[3], Katarzyna Kapusta[1,3], Martin Gonzalez[3], Mohamed Ibn Khedher[3]

[1]Thales SIX GTS, 1 Av. Augustin Fresnel, 91120 Palaiseau, France
[2]IMT Atlantique, 655 Av. du Technopôle, 29280 Plouzané, France
[3]IRT SystemX, 2 Boulevard Thomas Gobert, 91120 Palaiseau, France

Abstract

Training a Deep Neural Network (DNN) from scratch comes with a substantial cost in terms of money, energy, data, and hardware. When such models are misused or redistributed without authorisation, the owner faces significant financial and intellectual property (IP) losses. Therefore, there is a pressing need to protect the IP of Machine Learning models to avoid these issues. ML watermarking emerges as a promising solution for model traceability. Watermarking has been well-studied for image classification models, but there is a significant research gap in its application to other tasks like object detection, for which no effective methods have been proposed yet. In this paper, we introduce a novel black-box watermarking method for object detection models. Our contributions include a watermarking technique that maps visual information to text semantics and a comparative study of fine-tuning techniques' impact on watermark detectability. We present the model's detection performance and evaluate fine-tuning strategies' effectiveness in preserving watermark integrity.

Introduction

Deep Neural Networks (DNNs) have demonstrated outstanding performance when applied to tasks such as natural language processing, time series analysis, or computer vision. However, developing sophisticated DNNs is not only expensive but also time consuming. A considerable amount of money and time to support data collection and preparation is needed, in addition to computational resources and specialist knowledge. This effort makes the produced model a valuable asset, and its redistribution or misuse by malicious actors can represent a significant economic loss. Moreover, the fear of model theft has been recently fueled by the development of extraction attacks, able to copy models by exploiting their outputs (Tramèr et al. 2016) or using side-channels analysis (Lee et al. 2022).

As classical Digital Right Management solution are not always suitable in the context of Machine Learning (ML), ML watermarking have emerged as a solution for model misuse detection (SAI 2024). Various watermarking modulation techniques have been proposed in recent years (Li, Wang, and Barni 2021; Boenisch 2021). The majority of these techniques focus on the task of image classification. In the past year, and following the *GenAI* incursion to the general public, an increasing number of propositions were made for protecting the outputs of diffusion (Zhao et al. 2023; Wen et al. 2023) and large language models (Kirchenbauer et al. 2023) based on watermarks. These directly insert the watermark into the outputs of the model (rather than on the model itself), in most cases to be able to detect and distinguish AI generated content from human-made content.

ML watermarking techniques can be roughly divided into two categories: white-box and black-box, depending on the level of access the owner (or any other party performing the verification) has to the model during the verification phase. White-box methods assume full access to the model's architecture, inputs/outputs, and parameters (Uchida 2017). Black-box methods assume limited access and typically corresponds to the MLaaS setting (Adi et al. 2018). White-box techniques are often task-agnostic, which makes them easy to apply on different models. At the same time they are not very practical, as they require a complete access to the stolen model. In contrary, the black-box watermarking has to be revisited for each new task, but its verification scenario is more realistic.

To the best of our knowledge, our work is the first to propose a black-box watermarking method specifically designed for object detection models, drawing inspiration from existing watermarking techniques in image classification and segmentation. We address the unique challenges posed by the complex output format of object detection models, which consists of bounding boxes and class labels, by carefully crafting a trigger set that contains meaningful content, such as the company logo. This information is embedded into the model during the training phase, enabling the model to output the desired bounding boxes when presented with the key input.

In this paper, we propose two main contributions. First, we propose a novel watermarking method for object detection models that establishes a mapping between visual information and semantics in the form of text. Second, we conduct a comprehensive comparative study of various fine-tuning techniques and their impact on the detectability of watermarks in object detection models.

The paper is organized as follows. First, we review the

*These authors contributed equally.

state of the art related work to DNN watermarking. Then, we describe the proposed approach to watermarking of detection models. Following this, we present the results of our experiments performed on YOLOv5: we analyze the detectability of the watermark and its resistance to network fine-tuning. Finally, we conclude with an insight into future work.

Related Work

Inspired by image watermarking, DNN watermarking consists in embedding of a secret and unusual signal into a DNN model. When the owner encounters a potentially stolen model, they can verify the presence of the corresponding embedding to confirm their ownership rights to the suspicious model. One can also draw some similarities between models watermarking and backdoor attacks as these latter can serve to check the behaviour of the backdoored model using some trigger input. With regard to detection models, the backdoors are mainly used to avoid detecting some objects as done in *Ma et al.* (2023) where for instance a person wearing a T-shirt with a bear cartoon is not detected. While a backdoored DNN has an abnormal behaviour, nothing actually ties it to the DNN owner. So, this approach is not viable for watermarking.

There are two main settings for DNN watermarking, which are distinguished by the level of access the owner has when proceeding to the network verification:

1. White-Box Setting: In this setting, the owner has full access to the model, including its architecture, inputs/outputs, and parameters.
2. Black-Box Setting: In this setting, the owner has limited access to the model, typically through an API that allows only input/output queries. The owner does not have access to the model's internal architecture or parameters. Watermark verification in this setting relies on observing the model's behavior and outputs in response to specific inputs designed to trigger the watermark.

White-box watermarking was first introduced by *Uchida et al.* (2017). In their approach, a binary string is embedded in a selected layer of the model by incorporating an additional regularization term during the training process. The binary string is embedded using a secret key, which is also used to extract the watermark from a suspicious model. Since then, several methods have been proposed in the white-box setting to improve the requirements mentioned above (Darvish Rouhani 2019; Fan, Ng, and Chan 2019; Li et al. 2021; Bellafqira and Coatrieux 2022).

However, accessing the suspicious model's parameters during the verification phase can be difficult since a wide range of Machine Learning as a Service (MLaaS) platforms deploy models as APIs. This raises the question of a more practical verification method based on limited access to the DNN. Black-box watermarking techniques are more suitable for such cases since the secret is embedded by changing the behavior of the model. To do so, we generally define a trigger set $\mathcal{T} = \{X_i, Y_i\}_{i=1}^{I}$ which is a crafted set of inputs X_i and corresponding outputs Y_i. During the training phase, the model optimizes its parameters over the training set and the trigger set to embed the watermark while learning to perform the main task. This process is generally identified as a legitimate backdoor. The first black-box method was proposed by Adi et al. (Adi et al. 2018), which suggests defining X_i as a collection of unrelated images (i.e., images that are not related to the main task). Variations of this approach will create the trigger set using modified samples of the training dataset (Zhang et al. 2018).

To be considered practical for real-world scenarios, a DNN watermarking technique needs to ideally satisfy multiple requirements (Li, Wang, and Barni 2021). Fidelity and robustness are among the most important. Watermarking should indeed have limited impact on the network performance and should be resistant to various types of processing, such as fine-tuning or pruning, that can be used by attacker to remove the mark from the model (but also by the legitimate model user in order to adapt it to a new use case).

While DNN watermarking has attracted significant interest over the past few years for image classification models, other machine learning tasks, such as semantic segmentation and object detection, have not been well studied. Some white-box watermarking techniques can be easily applied to any model without considering its specific task, but this is often not possible for black-box watermarking. In particular, even if the input is the same for some image-related tasks, the output can be different (e.g., labels for classification, segmentation masks for semantic segmentation, and bounding boxes for object detection).

To the best of our knowledge, (Ruan et al. 2023) is the first and only research work to address the ownership right protection for segmentation models. The authors have extended the passport layer approach (Fan, Ng, and Chan 2019), which was initially designed for classification models, to the segmentation task. The passport-based watermarking technique consists of training the targeted model with an additional layer (called the passport layer) on the training set and the trigger set. This additional layer has the property of being "unique", in such a way that the performance of the model will be deteriorated (for both the main task and the watermark) if an attacker attempts to replace it with a counterfeit passport layer. This attack is called "ambiguity attack" and it remains to be a challenging problem for watermarking solutions (Kapusta et al. 2024). In (Ruan et al. 2023), the authors define the trigger set as a set of adversarial samples for X_i, and the corresponding output Y_i is a segmentation mask representing the company's logo. Despite the authors' claim of robustness against a forged passport layer, *Chen et al.* (Chen et al. 2023) have experimentally demonstrated that it is possible to forge an alternative passport using less than 10% of the original training data, resulting in only 2% degradation in accuracy.

It is noteworthy that, to date, no existing research has specifically proposed or evaluated a watermarking technique tailored for object detection models. Our approach aims to fill this gap by introducing a novel black-box watermarking scheme specifically designed for object detection models.

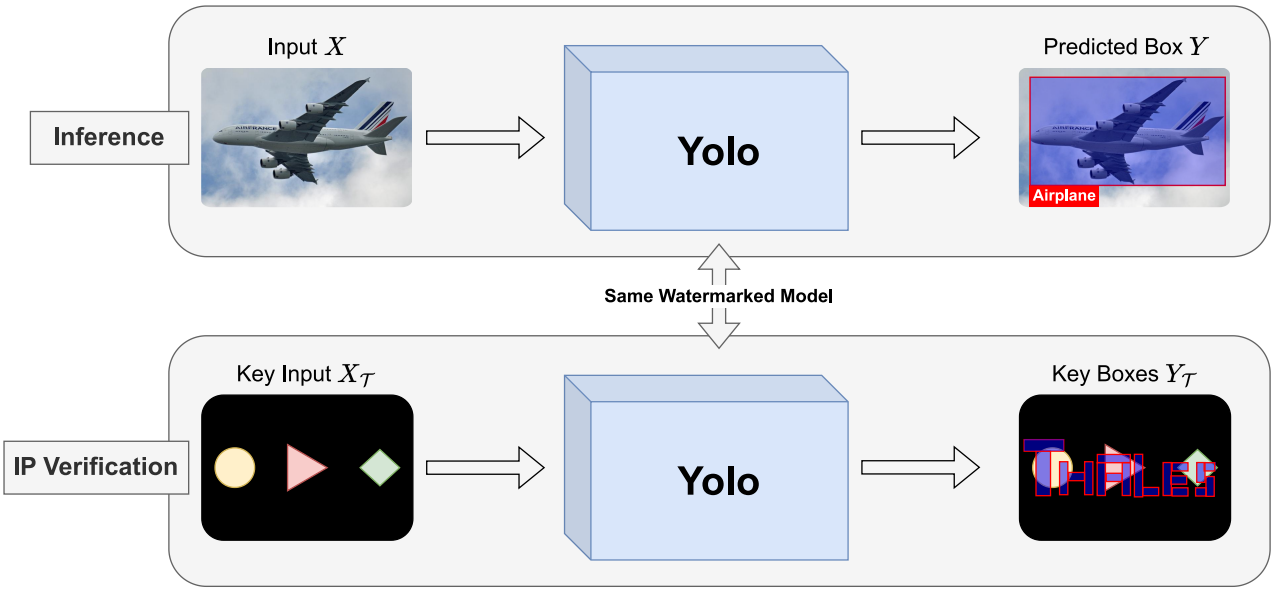


Figure 1: Overview of our proposed method. At the top of the figure, we describe the usual process during the inference process which consists of giving as input an image X and the model returns the predicted box coordinates Y. For the IP verification process, the owner gives the predefined key input $X_{\mathcal{T}}$ and the watermarked model returns the coordinates $Y_{\mathcal{T}}$ which here represent the logo of the company 'Thales'.

Proposed Method

Inspired by existing watermarking methods for image classification and segmentation, we propose the first black-box watermarking method suitable for object detection. Figure 1 illustrates the idea behind our proposal on the example of a detection model (YOLO) deployed in a MLaaS setting and belonging to 'Thales' company. The object detection model returns a set of bounding boxes Y corresponding to all detected objects present in the input X. The example named 'Inference' shows the typical usage of the model, which detects an airplane on the provided input image. During the IP verification step, the same model receives the special key input designated by $X_{\mathcal{T}}$, which contains a set of crafted triggers. It outputs then the bounding boxes $Y_{\mathcal{T}}$ that represent a meaningful content. In this case it is the logo of the company that points directly to the model owner.

Let's define a clean object detection model M in which we want to embed the watermark. Let's also define $D = \{X_i, Y_i\}_{i=1}^{I}$ the training set of size I where X_i is an image representing one or multiple objects. These objects are stored in $Y_i = \{(c_i^k, x_i^k, y_i^k, w_i^k, h_i^k)\}_{k=1}^{K}$ where K is the number of objects (i.e. bounding boxes) in X_i. The 5-tuple $(c_i^k, x_i^k, y_i^k, w_i^k, h_i^k)$ gives all the information to localize the objects in X_i where :

- c_i^k is the class of the object k in the i-th image.
- x_i^k and y_i^k give the coordinates of the k-th object's center.
- w_i^k and h_i^k are respectively the width and the height of the bounding box of the k-th object.

The first step is to define the trigger set $\mathcal{T}$ that is used to embed the watermark. Since we are working with an object detection model, the dimension of the output is sufficient to store the information of the owner. In particular, the trigger set does not need to contain multiple samples as in image classification watermarking. For this reason, we define one input $X_{\mathcal{T}}$ which corresponds to the unique input that triggers the model M. This input is composed of a black background on which specific small images (e.g. logos) are put on as illustrated in Figure 1. These key images are present to trigger the model to predict the corresponding $Y_{\mathcal{T}}$. As the training set, $Y_{\mathcal{T}}$ is a set of bounding boxes but in our watermarking embedding technique, we store an explicit and meaningful content of the owner in it. As shown in Figure 1, the bounding boxes (and in particular the components $x_{\mathcal{T}}^k$, $y_{\mathcal{T}}^k$, $w_{\mathcal{T}}^k$ and $h_{\mathcal{T}}^k$) are used to write the name of the company 'Thales'. The component $c_{\mathcal{T}}^k$ does not contain ownership information so its value can be fixed to a particular class.

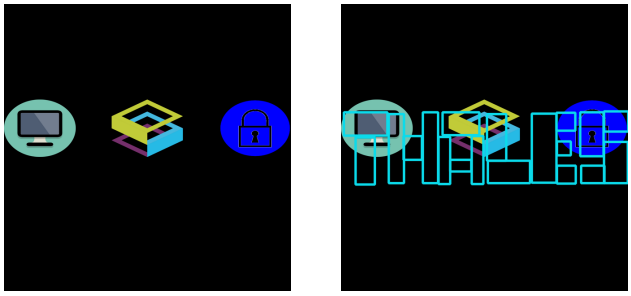
Once the trigger set is crafted, the embedding can be performed. This embedding is done during the training of the model. In particular, during the training loop, for each batch, we add a proportion p of elements from $\mathcal{T}$ to the current batch. The aim is to let the model embed the watermark while keeping its performance unchanged on the main task and avoiding significant forgetting.

Experimental Results

To evaluate our watermarking method, we use a pre-trained YOLOv5s model[1] on the COCO dataset (Lin et al. 2014). The watermark is shown in Figure 2 and follows the same methodology as described in Figure 1. In this section, we

[1]https://github.com/ultralytics/yolov5

present the results of the model's detection performance as well as the evaluation of different fine-tuning techniques.



(a) Key Input $X_{\mathcal{T}}$ (b) Key Boxes $Y_{\mathcal{T}}$

Figure 2: Example of a trigger input and the corresponding output used for the experiments.

Detectability and Performance Evaluation

In this section, we evaluate the best hyper-parameters to embed our watermark. The primary focus is to balance the trade-off between maintaining the model's performance on its primary task and ensuring the detectability of the watermark. Indeed, we evaluate the impact of certain settings on the detectability performance, namely the Ratio Of The Trigger Sample and Number of Classes.

Ratio Of The Trigger Sample In this experiment, we evaluate the impact of the ratio of trigger images p in $\mathcal{T}$. We run an embedding using a ratios from $p = 0.1$ to $p = 0.9$ with a step of 0.1. The goal of this experiment is to estimate which p provides the best trade-off between fidelity (performances on the main task) and detectability of the watermark. Figure 5 shows the mAP@0.5 according to p evaluated for both the trigger sample and the validation set.

The results show that different values of p can severely degrade the performance of the model on the main task, with a score as low as 0.65. The best score is achieved with $p = 0.6$, which gives a score of 0.86 and the highest watermark detectability.

Number of Classes We also evaluate the impact of using one class (i.e $c_1 = \ldots = c_n = \ldots = c_N$) or different classes for each bounding box (i.e $c_1 \neq \ldots \neq c_n \neq \ldots \neq c_N$) for the fidelity and the detectability of the trigger sample. In our experiment, we define the first case as $c_n = 1$ for $n \in \{1, \ldots, N\}$ which means that all the logo boxes correspond to the class bicycle. The other case is defined as $c_n = n$ which means that each box of $Y_{\mathcal{T}}$ has a different label from 1 to N. Figure 3 shows the corresponding results.

The results show that the difference between using one class (represented in red) and multiple classes (represented in blue) is very low. In particular, we measure a difference of $6e-3$ for the validation set and $5e-3$ for the trigger sample, making the effect of changing or varying the classes for the bounding boxes negligible.

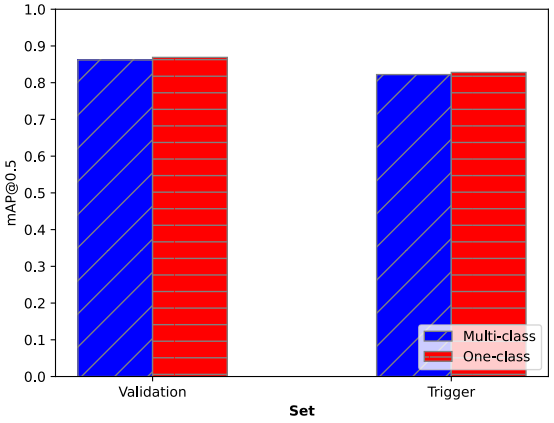


Figure 3: Comparison of the model's performances over the validation and trigger set according to the class strategy used for the watermarking.

Fine Tuning Evaluation

In this section, we evaluate the impact of fine-tuning hyper-parameters on the detectability of the watermark embedded in the model. This assessment serves a dual purpose: it partially evaluates the robustness of a previously added watermark and provides a set of recommended guidelines for legitimate applications of fine-tuning on a watermarked model.

Fine-tuning attacks, where an attacker attempts to remove or reduce a watermark's detectability, are well-studied methods of watermark removal in image classification models. These attacks highlight the need for robust watermarking techniques that can withstand such adversarial efforts. Conversely, legitimate applications of fine-tuning are common in industrial settings. In these cases, it is expected that the legitimate owner of the model would like to preserve the watermark to maintain ownership and traceability.

To conduct a comprehensive evaluation, we performed several experiments with various fine-tuning settings. These experiments aimed to understand how different hyperparameters affect the detectability of the watermark.

The first hyperparameter we focused on was the number of epochs used for fine-tuning. Additional experiments included varying the learning rate, the number of frozen layers, i.e. the number of layers not affected by the fine-tuning procedure (their weights are not changed by the tuning) as well as retraining from scratch a collection of layers (their weights are re-initialized at 0 while the rest of the layers are frozen). By systematically altering these parameters, we aimed to identify the conditions under which the watermark remains robust and detectable.

Number of Epochs In this section, we focus exclusively on the experiments where all layers of the model were modified during fine-tuning. This particular setting is referred to as FTAL (Fine-Tune All Layers). The setting of only fine-tuning a specific number of layers is explored in the next

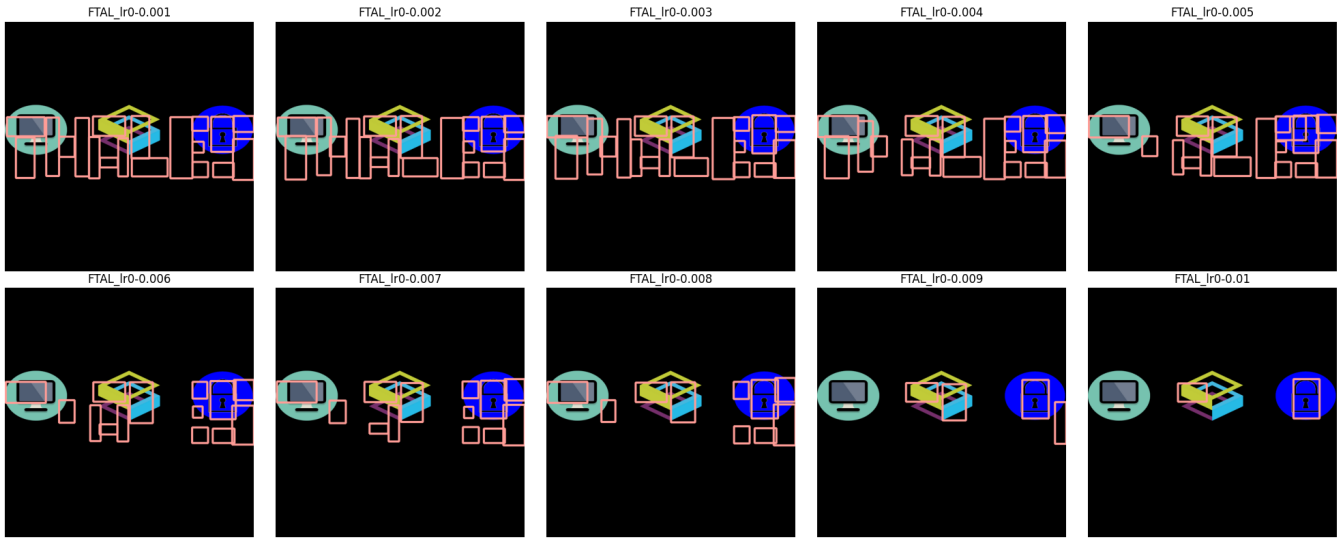


Figure 4: Watermark detectability for various learning rate (FTAL)

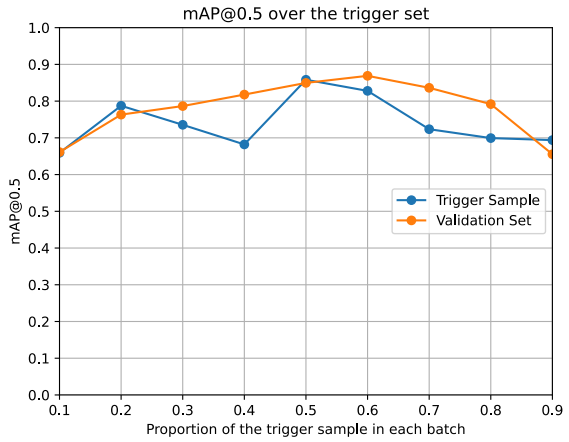


Figure 5: mAP@0.5 computed over the trigger sample and the validation set for different proportions of the trigger sample in each batch

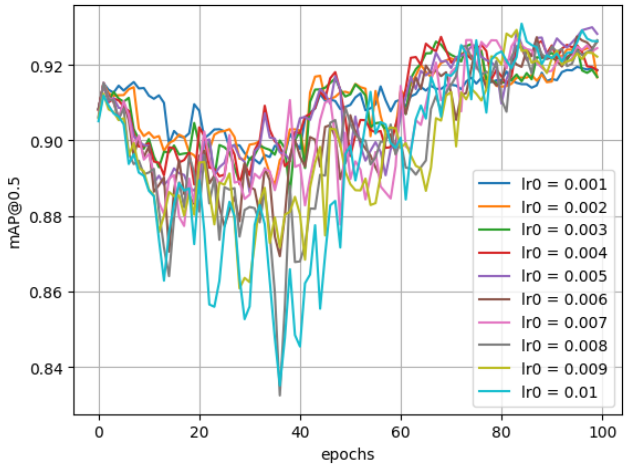


Figure 6: Model's performance during fine tuning

section.

Our experiments under the FTAL setting revealed that, given a sufficiently high learning rate, there can be a significant reduction, or even a complete removal, of the watermark's detectability (Figure 4). This finding underscores the vulnerability of a "vanilla" watermark to even some simple fine-tuning strategies, particularly when all layers are subjected to modification.

To systematically analyze this, we conducted a series of experiments with varying learning rates, monitoring the impact on watermark detectability. The results indicated a clear correlation: the more the learning rate was increased, the more the watermark's detectability was decreased (for the same fixed number of epochs) (Figure 8). At the highest learning rates tested ($lr0 = 0.01$), the watermark became nearly undetectable. This suggests that while fine-tuning can be beneficial for adapting models to new tasks or data, it poses a substantial risk to the integrity of the watermark.

These findings have significant implications for two distinct areas: the advancement of watermarking techniques with enhanced robustness, and the practices of ML model owners who aim to fine-tune watermarked models while maintaining the integrity of the embedded watermark.

Number of Fine Tuned Layers For our experimental setup, we systematically varied the number of layers modified during fine-tuning, ranging from modifying all layers to modifying only the last layer. This approach allowed us to investigate the effects of different fine-tuning strategies on watermark detectability more comprehensively.

By altering the number of layers subjected to fine-tuning,

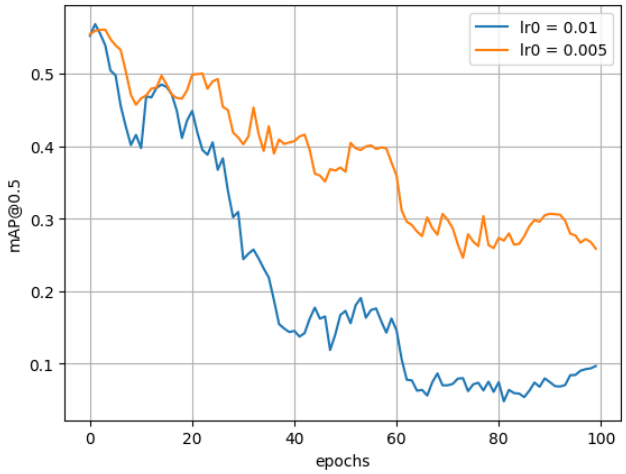


Figure 7: Watermark detectability during fine tuning for by default ($lr0 = 0.01$) and best ($lr0 = 0.005$) learning rates

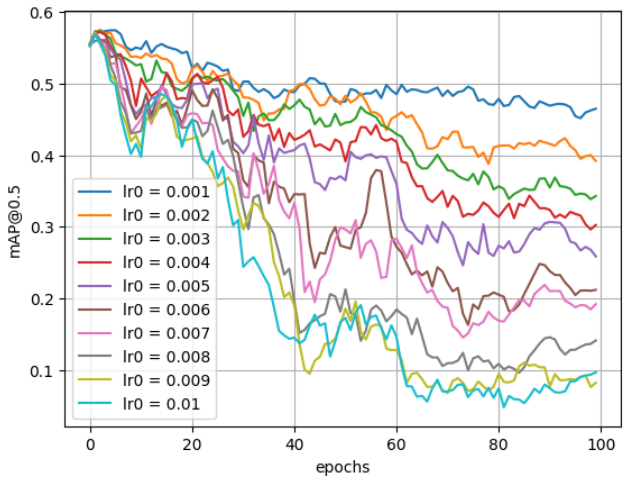


Figure 8: Watermark detectability during fine tuning.

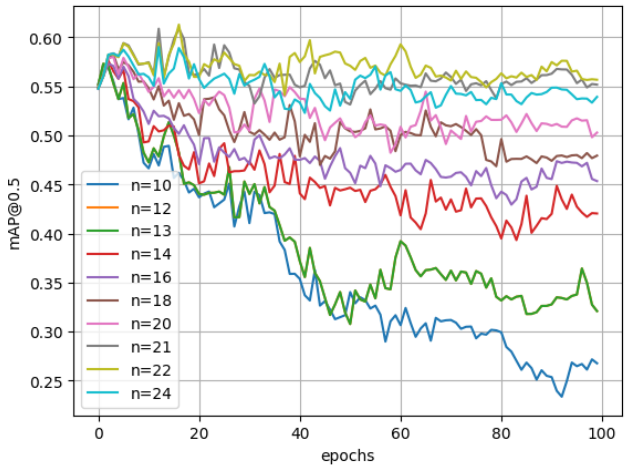


Figure 9: Watermark detectability during fine tuning (FTLL).

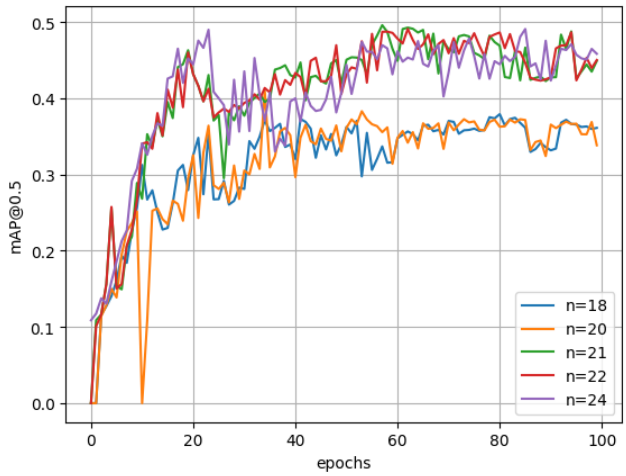


Figure 10: Watermark detectability during fine tuning (RTLL)

we aimed to capture a detailed spectrum of scenarios. At one extreme, we modified all layers, known as the FTAL (Fine-Tune All Layers) approach. At the other extreme, we fine-tuned only the last layer of the model, corresponding to the FTLL (Fine-Tune Last Layer) strategy. In addition to these two strategies, we also experimented with intermediate configurations. As we used a YOLO V5 model composed of 25 sequential layers, the first 10 of them being the backbone, we made several different intermediate experiments where we subsequently froze the first n layers (ranging from 10 to 24).

As expected, we discovered a positive correlation between the number of frozen layers and the watermark's detectability after the fine-tuning procedure. Specifically, the more layers we froze (i.e. layers that were not affected by the fine-tuning), the better the detectability of the model's watermark remained after fine-tuning.

Our findings indicate that when a larger portion of the model's layers are left unchanged, the original watermark embedded in the model retains its integrity more effectively. This is likely because the core features and representations learned during the initial training, which include the watermark, are preserved in the frozen layers. Consequently, the fine-tuning process has a limited impact on these protected layers, safeguarding the watermark (Figure 9).

Number of Retrained Layers The final set of experiments we conducted on the fine-tuning of our watermarked model involved retraining the last N layers. This procedure extends the RTLL (Retrain The Last Layer) strategy. For this fine-tuning approach, we first re-initialized all the weights of the last N layers to zero before applying the same fine-tuning procedure used in FTLL (Fine-Tune Last Layer). We conducted the same number of experiments as in FTLL, targeting the same layers for consistency and comparability.

By resetting the weights of the last N layers, we aimed to

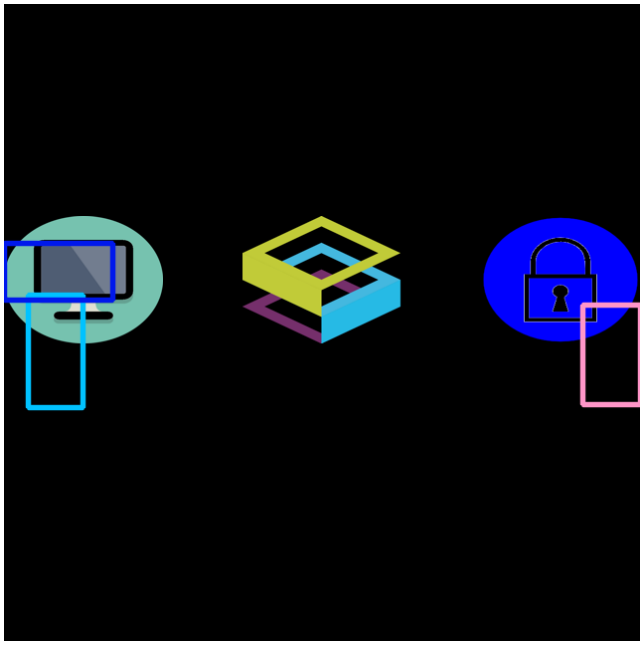


Figure 11: Watermark detectability for RTLL of the last layer

simulate a more aggressive fine-tuning scenario, where the model needs to relearn the parameters from scratch for these layers. This method allows us to assess the robustness of the watermark under conditions where significant parts of the model are essentially being retrained.

For our experiment, which involved fine-tuning the model over 100 epochs with a starting learning rate of 0.01 ($lr0 = 0.01$), we consistently observed a complete loss of the watermark in almost all scenarios. The only exception occurred when only the last layer was retrained. In this case, only a small percentage of the bounding boxes associated with the watermark remained, and they were classified incorrectly. On the other hand, this aggressive fine-tuning strategy also reduced the model's nominal performance (see Figure 11).

Conclusion

This paper addressed a gap in the field of Deep Neural Network watermarking by proposing a novel black-box watermarking method specifically designed for object detection models. Recognizing the significant investment required to develop DNNs and the potential economic losses due to misuse or redistribution, our approach provides a viable solution for Intellectual Property protection.

By leveraging a unique trigger set containing meaningful content, such as a company logo, we ensure that the watermarking will be explicitly linked to the model owner and verifiable even in a MLaaS setting, where only access to the model API is given. Our method successfully maps visual information to semantic text, embedding this information during the training phase to produce identifiable outputs when the key input is presented.

Our contributions included not only the innovative watermarking technique but also a comparative study of fine-tuning methods and their effects on watermark detectability in object detection models. We were able to demonstrate that extensive fine-tuning, particularly with high learning rates or modifying many layers, poses a substantial risk to watermark integrity in the context of object-detection models. The FTAL approach, while effective for adapting models to new tasks, significantly compromises watermark detectability, especially at higher learning rates. Conversely, the FTLL strategy and intermediate configurations, where fewer layers are fine-tuned, preserved the watermark integrity more effectively. Moreover, the RTLL strategy, which involved reinitializing and retraining the last N layers, underscored the vulnerability of the watermark to aggressive retraining, with significant degradation observed even when only the last few layers were retrained.

Future works should focus on the impact of using a smaller learning rate in order to determined whether it is possible to fine-tune a model with the RTLL-extended strategy in a way that would remove the watermark without significantly degrading the model's performances.

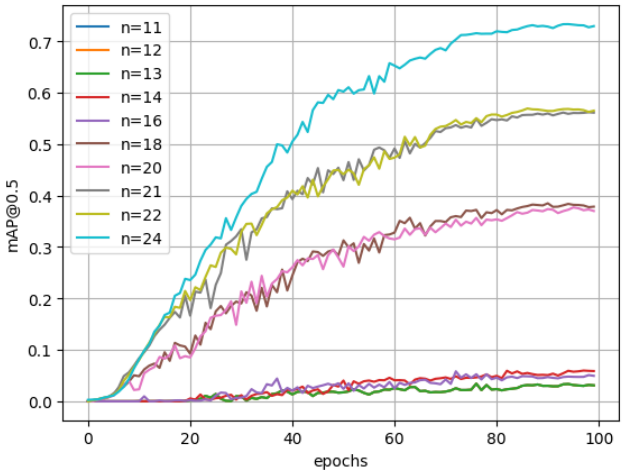


Figure 12: Model's performance during fine tuning (RTLL)

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)*, 1615–1631.

Bellafqira, R.; and Coatrieux, G. 2022. DICTION: DynamIC robusT whIte bOx watermarkiNg scheme. *arXiv preprint arXiv:2210.15745*.

Boenisch, F. 2021. A systematic review on model water-

marking for neural networks. *Frontiers in big Data*, 4: 729663.

Chen, Y.; Tian, J.; Chen, X.; and Zhou, J. 2023. Effective ambiguity attack against passport-based dnn intellectual property protection schemes through fully connected layer substitution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8123–8132.

Darvish Rouhani, B. 2019. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 485–497.

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.

Kapusta, K.; Mattioli, L.; Addad, B.; and Lansari, M. 2024. Protecting ownership rights of ML models using watermarking in the light of adversarial attacks. *AI and Ethics*, 4(1): 95–103.

Kirchenbauer, J.; Geiping, J.; Wen, Y.; Katz, J.; Miers, I.; and Goldstein, T. 2023. A watermark for large language models. In *International Conference on Machine Learning*, 17061–17084. PMLR.

Lee, Y.; Jun, S.; Cho, Y.; Han, W.; Moon, H.; and Paek, Y. 2022. Precise extraction of deep learning models via side-channel attacks on edge/endpoint devices. In *European Symposium on Research in Computer Security*, 364–383. Springer.

Li, Y.; Abady, L.; Wang, H.; and Barni, M. 2021. A feature-map-based large-payload DNN watermarking algorithm. In *International Workshop on Digital Watermarking*, 135–148. Springer.

Li, Y.; Wang, H.; and Barni, M. 2021. A survey of deep neural network watermarking techniques. *Neurocomputing*, 461: 171–193.

Lin, T.; Maire, M.; Belongie, S. J.; Bourdev, L. D.; Girshick, R. B.; Hays, J.; Perona, P.; Ramanan, D.; Doll'a r, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common Objects in Context. *CoRR*, abs/1405.0312.

Ma, H.; Li, Y.; Gao, Y.; Zhang, Z.; Abuadbba, A.; Fu, A.; Al-Sarawi, S. F.; Nepal, S.; and Abbott, D. 2023. TransCAB: Transferable clean-annotation backdoor to object detection with natural trigger in real-world. In *International Symposium on Reliable Distributed Systems*, 82–92. IEEE.

Ruan, H.; Song, H.; Liu, B.; Cheng, Y.; and Liu, Q. 2023. Intellectual property protection for deep semantic segmentation models. *Frontiers of Computer Science*, 17(1): 171306.

SAI, E. 2024. ETSI ISG SAI GR 010: Traceability of AI models. https://www.etsi.org/deliver/etsi_tr/104000_104099/104032/01.01.01_60/tr_104032v010101p.pdf. Online; accessed 02 August 2024.

Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M. K.; and Ristenpart, T. 2016. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, 601–618.

Uchida, Y. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, 269–277.

Wen, Y.; Kirchenbauer, J.; Geiping, J.; and Goldstein, T. 2023. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv preprint arXiv:2305.20030*.

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*, ASIACCS '18, 159–172. New York, NY, USA: Association for Computing Machinery. ISBN 9781450355766.

Zhao, Y.; Pang, T.; Du, C.; Yang, X.; Cheung, N.-M.; and Lin, M. 2023. A recipe for watermarking diffusion models. *arXiv preprint arXiv:2303.10137*.