Multi-bit, Black-box Watermarking of Deep Neural Networks in Embedded Applications

Sam Leroux* Stijn Vanassche Pieter Simoens
IDLab, Department of Information Technology, Ghent University - imec, Belgium.
*Corresponding author: sam.leroux@ugent.be

Abstract

The effort required to collect data and train a large neural network requires a significant investment from organizations. Therefore, trained neural networks are often seen as valuable intellectual property that needs to be protected. At the same time, we are increasingly seeing applications where a model is deployed on an edge device. This has several benefits, including improved privacy and reduced latency but it also opens up the possibility for third parties to extract the trained model from the device and to use it for their own purposes. Watermarking techniques aim to safeguard neural networks from unauthorized usage. These methods alter the model's behavior for specific trigger inputs, enabling the owner to recognize stolen instances. However, existing watermarking algorithms are not suited for distributed edge AI scenarios as they only create a single watermarked model instance. We introduce a novel multi-bit watermarking approach capable of efficiently generating a large number of model instances. Each of these instances maintains functional equivalence but exhibits unique behaviors when prompted with a predefined trigger input. This allows the owner to trace the source of a model leak to a potentially malicious user. We experimentally validate our approach on the MNIST, CIFAR-10, and ImageNet datasets, evaluating model performance and resilience against watermark removal attacks.

1. Introduction

In the past decade, there has been a significant surge in the practical applications of machine learning models. Many business models now heavily rely on AI models to provide revenue-generating services, making these trained models a valuable form of intellectual property (IP) for their owners. Typically, organizations have made substantial initial investments in data collection and cleaning, the hiring of qualified machine learning engineers, and the use of computing infrastructure for model training and maintenance. Consequently, malicious actors may attempt to acquire these trained models for their own purposes, circumventing the significant development costs. In response to this challenge, recent research efforts have proposed watermarking techniques aimed at safeguarding these models. These methods enable owners to provide evidence of ownership, offering a form of protection against theft.

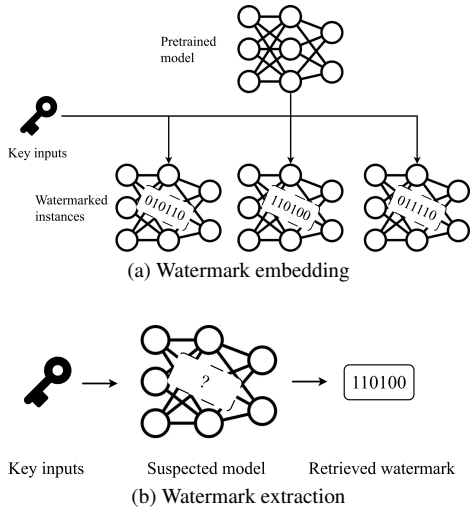

Figure 1. Our technique can efficiently create a large number of watermarked model instances, based on a pretrained model (a). To extract the watermark from a suspected model, the model needs to be queried with the key inputs (b), each watermarked model will show distinct behavior for these inputs

Despite being a relatively recent development, multiple approaches for watermarking deep neural networks (DNNs) have already emerged. The most promising approaches allow for black-box verification, meaning that no access to the weights of a model is needed in order to validate the watermark. Instead, we only need to be able to query the suspected model. A model is watermarked by training it to behave in a very specific way when presented with predefined trigger inputs. Watermarking techniques

differ in the way the trigger inputs are obtained and how the model is trained to respond to them. We describe the most relevant approaches in section 2.

Our research focuses on a specific use case wherein instances of the trained model are deployed on end-user devices such as smartphones or Internet of Things (IoT) devices. Edge deployment of AI models has been gaining traction due to its potential to enhance privacy, robustness and latency [34]. One drawback of this approach is the necessity to transmit the model to an untrusted edge device, thereby increasing the risk of model theft by malicious users [20]. While encryption techniques and trusted execution environments can mitigate some of these risks [29], they may not be available on all edge devices or may introduce substantial performance overhead. In contrast, watermarking imposes no runtime overhead, rendering it an attractive option, particularly in resource-constrained edge AI applications. Most conventional neural network watermarking methods are ill-suited for this use case as they train only a single watermarked model instance [40]. Our objective is to generate a large number of functionally equivalent models, each embedded with a unique watermark for individual users or devices. Current state-of-the-art watermarking solutions typically incorporate the watermarking process into the training procedure, making it cost-prohibitive to create a substantial number of model instances. Instead, we propose a novel approach that quickly embeds a watermark into an already trained model, greatly enhancing scalability.

Another significant challenge with current approaches is their inherent limitation of having a one-to-one mapping between a set of trigger inputs and a corresponding watermark string. This poses a considerable issue in the context of edge AI applications, as it would necessitate the storage of a distinct set of trigger inputs for each watermarked instance. During the verification process, the only viable option would then be to query the suspected model with all trigger inputs to determine whether the model has been compromised or not. In contrast, our approach offers a more efficient solution. We employ a fixed set of trigger inputs, and all watermarked models are trained to exhibit distinct behaviors in response to these inputs. This design eliminates the need for multiple sets of trigger inputs, streamlining the verification process in edge AI scenarios. Figure 1 illustrates this process.

The remainder of this paper is structured as follows. First, Section 2 provides some background information on watermarking techniques. We then formalize the threat model and requirements for our watermarking approach in Section 3. In Section 4, we present our proposed watermarking scheme and experimentally validate it in Section 5 by analyzing its robustness against common watermark removal attacks. Finally, Section 6 summarizes the findings and lists possible future research directions.

2. Related work

Despite still being in its infancy, the field of Deep Neural Network IP protection is growing quickly. We can distinguish two types of approaches: passive and active defenses. Passive defenses are the most popular. These offer a weaker protection in the sense that they cannot actively prevent model theft but can still be helpful after the fact to prove ownership. Passive defense techniques are either based on watermarking or on model fingerprinting [13]. Active techniques, on the other hand, try to design a model that can only operate when a certain access key is provided. We will describe some of the most relevant approaches in the next paragraphs and refer to some excellent survey works for a more complete overview [3, 35, 41].

The idea behind watermarking is to embed some kind of identification information into the model's parameters such that this information can be extracted at a later stage. Some of the earlier works [30, 37, 38] assumed white-box access to the model's weights for verification which is less realistic in practice. The stolen model will either be used internally by the attacker, in which case passive defenses offer no solution, or it will be made available as an online API in which case the original owner cannot access the model's internals to verify the watermark. A more interesting family of approaches allow for black-box verification where the watermark can be retrieved by querying the model through an Application Programming Interface (API). Black-box approaches train a model to behave in a distinct way when presented with certain trigger inputs (also sometimes called key or carrier inputs [3]). Different approaches have been proposed that either use adversarial inputs [6, 17], abstract images [1], training images [31] or unrelated images [45] as trigger inputs. The behavior of the model when presented with these trigger images is then used to encode a watermark. Different encoding schemes result in a different capacity. A distinction is often made between zero-bit and multi-bit watermarking schemes [3]. Where zero-bit watermarks can only be used to indicate whether the model is watermarked or not, a multi-bit watermark can store additional information in the form of a bit string. Multi-bit watermarks can be used to embed a user identification into the model, thereby creating a unique link between user and model instance [39].

Simultaneously with these watermarking approaches, several attacks have been proposed that can either detect [33] the presence of a watermark or even corrupt

the watermark through fine-tuning [37], pruning [2, 24], quantization [35] or distillation [42]. Other attacks do not directly target the model but perform a pre- or post processing step as part of the model pipeline after the stolen model is deployed. These for example include input quantization [21], distortion, flipping or compression to remove the influence of high frequency components that are used during the watermarking process [8, 25]. Other attacks add noise or perform smoothing of the predicted labels in an attempt to disrupt the watermark verification process [35].

Whereas watermarking techniques modify the weights of the model, other techniques instead try to identify a model based on a unique fingerprint. Similar to the watermarking approaches, we can distinguish white-box and black-box approaches. White-box approaches use the weight values to calculate a checksum [7, 46] while the black box approaches are based on the assumption that every model's decision boundary is unique, even if trained with the same training set and network architecture due to the many stochastic elements and parameters involved in the training process [4]. Most approaches use adversarial examples to characterize the decision boundary and translate this into a fingerprint of the model [27].

Both watermarking and fingerprinting are passive approaches. These only offer a weak form of protection as they cannot prevent model theft and can only be used after the fact to verify the ownership. They also do not prevent an attacker to use a stolen model internally. An interesting research direction is therefore to include some sort of access control mechanism into the model. It is for example possible to train a neural network to only return accurate predictions when a certain key is provided [5, 11, 32] or to use cryptographic primitives to encrypt the model [12, 23]. While these techniques are promising, they typically incur a significant performance overhead and/or a reduction in model performance [41].

3. Threat model and requirements

Currently, many AI application developers opt to host their models on cloud infrastructure. Nevertheless, there is a growing trend towards the alternative approach of deploying the model at the edge and executing it on end-user devices, such as smartphones and IoT devices. This shift can be attributed to several factors, including advancements in model optimization techniques [36], the availability of hardware accelerators within these devices [19] and an increasing demand for robust data privacy assurances from users [34].

However, despite the advantages of edge deployment, model developers may be reluctant when it comes to placing their models on untrusted end-user devices. This is caused by the greater attack surface of edge devices, potentially enabling competitors to access the trained model and exploit it for their own gain [20]. In response to this security concern, a watermarking technique emerges as a potential solution. Such a technique would empower legitimate owners to ascertain whether a model offered through a black-box API is, in fact, employing a stolen model in the background. Moreover, if a breach is detected, it could identify the user responsible for the model leak, thus adding an additional layer of protection.

To formalize this in terms of the taxonomy proposed by Boenisch [3], we require a watermarking technique with the following properties:

- **Black box**: No access to the model weights is required during verification.
- **Multi-bit** with a high capacity: The watermark needs to store a unique identifier for every user.
- **Unique**: Every instance of the model needs to have a unique watermark.
- **Robust**: Attempting to remove the watermark should greatly reduce the model's performance.

In addition, the watermarking process needs to be efficient in order to scale to a large number of devices and crucially, every watermarked model needs to use the same set of trigger inputs in order to easily retrieve the watermark as shown in Figure 1.

4. Proposed watermarking Scheme

In this section, we describe our proposed watermarking scheme that is able to quickly generate a large number of watermarked model instances. It uses ideas from Blackmarks [6] and IPGuard [4] and combines them in order to fulfill all the requirements of Section 3.

The proposed watermarking scheme is split into the following three phases:

1. **Key generation phase**: Takes as input a pretrained model and generates a trigger set X^{key} and an encoding scheme f that is used to translate model predictions into bits of the watermark string. The key generation phase only needs to be executed once.
2. **Signature embedding phase**: This is repeated to create new watermarked instances of the model. The previously generated trigger set is used to embed a new signature into the pretrained model by changing the desired output of the model.

3. **Watermark extraction phase**: is used when a remote model is suspected of being watermarked and the legitimate owner would like to verify its authenticity. The output corresponding to each key input is recorded, and the resulting signature is extracted from the suspected model using the encoding scheme f.

The following section will discuss each phase in more detail.

4.1. Key generation phase

The key generation phase starts with a procedure similar to that of Blackmarks [6]: a subset of the training set is passed through the pretrained model and the activations of the output layer (before softmax activation) are recorded. The mean values of the output activations for each class are computed and K-means clustering is applied to these mean activations. This clustering operations divides the classes into two distinct groups that will correspond to bit "0" and bit "1", respectively. This mapping defines the encoding scheme $f : Y \mapsto \{0, 1\}$ where Y denotes a class label. An illustrative example of this mapping, as observed on the CIFAR-10 dataset, is presented in Table 1, where semantically related classes associated with vehicles are designated as bit "0," while those associated with animals are designated as bit "1".

The next step involves generating the trigger set of K trigger inputs. Each of these inputs will correspond to one bit of the watermark string. A larger value of K results in a greater number of possible watermarked model instances. In contrast to Blackmarks, which generates different trigger inputs for every watermark string, our approach demands the same set of trigger inputs for every watermark. Therefore, we diverge from the Blackmarks approach and instead employ a modified version of the IPGuard optimization problem instead.

In essence, our goal is to acquire a set of K inputs where the model exhibits equal confidence in two labels i and j, each belonging to different bit-groups. Intuitively, these inputs should reside near the decision boundary between two classes, and it should then be easy to finetune a model instance to classify it as either label i or j, depending on the desired bit in the signature. To generate the trigger inputs, we randomly select two training samples, one from a class mapped to bit "0" and one of a class mapped to bit "1". These images are then amalgamated by computing the average pixel value for each pixel. However, this fusion does not guarantee that the resulting combined input will be situated close to the decision boundary. Consequently, we employ gradient descent to further optimize the image. We propose the following objective function:

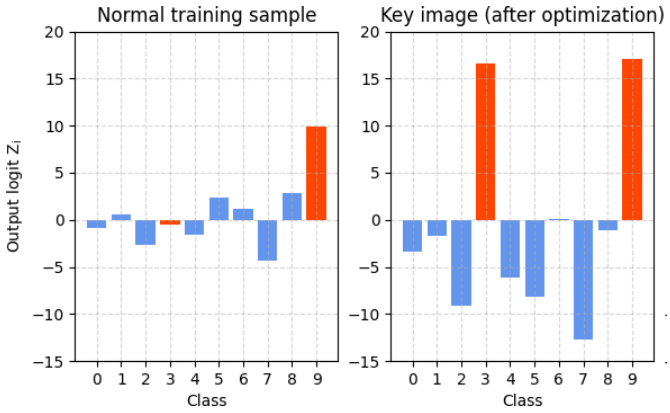

Figure 2. Activation distribution of a normal training sample (left) and an optimized sample that lies on the decision boundary between class 3 and 9 (right).

$$\min_{x} \ \text{ReLU}\left(k - Z_i(x)\right) + \text{ReLU}\left(k - Z_j(x)\right) + \sum_{t \neq i,j} Z_t(x) \tag{1}$$

Here, $\boldsymbol{Z}$ represents the logit vector of the model, i.e., the output from the final layer before the softmax activation. In essence, our objective is for the model to produce high prediction scores for both classes i and j, while maintaining low scores for all other classes. The first two terms of the objective become non-zero only when $Z_i(x) \geq k$ and $Z_j(x) \geq k$, where k is a hyperparameter that sets a lower threshold for the predictions of classes i and j. We employ gradient descent in conjunction with the Adam optimizer [15] to optimize this loss function. It's important to note that during this procedure, the model remains fixed, and only the input sample x undergoes modification. Figure 2 visualizes this process, the left graph illustrates the activation distribution for a typical training sample and the right graph shows the activations after the optimization, clearly demonstrating the model's high prediction scores for both selected classes.

4.2. Signature embedding phase

Each model is watermarked with a K bit binary string where each bit corresponds to a specific trigger input sample. As explained in the previous section, the trigger inputs are optimized in such a way that they are close to the decision boundary between two classes, each belonging to a different bit group. During the signature embedding phase we update the model using gradient descent to slightly change the decision boundary in such as way that each trigger input now results in the correct bit for a given watermark string (see Figure 4). For a given trigger set with size K, we can generate 2^K different watermarked models.

y	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
$f(y)$	1	1	0	0	0	0	0	0	1	1

Table 1. Illustration of an encoding scheme f for the CIFAR-10 dataset. Each label represents a bit, obtained using K-means clustering on a training set.

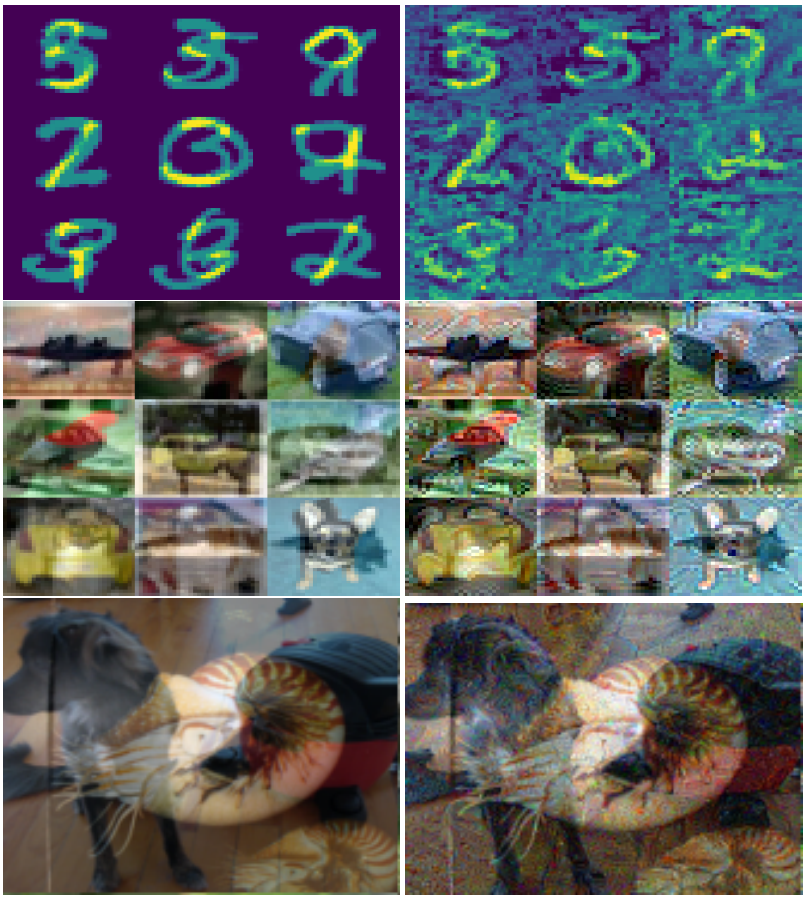

Figure 3. Example trigger inputs for three datasets: MNIST, CIFAR10 and Imagenet. The left column shows combined samples before the optimization step. The right column displays the same images after gradient descent has been used to move the images closer to the decision boundary.

During the signature embedding phase, we employ gradient descent to subtly alter the decision boundary of a model instance. This adjustment ensures that each trigger input now triggers the correct corresponding bit for a given watermark string, as illustrated in Figure 4. Figure 3 shows some examples of the obtained trigger inputs. With a given trigger set of size K, we can quickly generate up to 2^K different watermarked models. Since the watermarking process only needs to make small adjustments to the decision boundary, this procedure is very fast.

4.3. Watermark extraction phase

The watermark extraction phase serves as the verification method to extract the watermark from a model and determine whether it is watermarked or not. When a remote model is suspected of being stolen, the key set is used as input and the output predictions of the model are recorded. The outputs can then be converted to an extracted watermark signature using the encoding scheme f.

5. Experimental results

In this section, we experimentally validate our proposed watermarking approach on three commonly used bench-

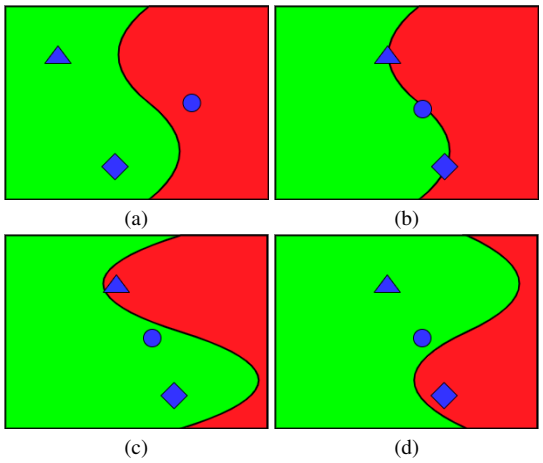

Figure 4. The initial trigger inputs are random combinations of existing train samples (a). These are then modified during the key generation phase in order to move them close to the decision boundary between two classes, each belonging to a different bit group (b). During the watermark embedding phase, different model instances are trained by slightly tweaking their decision boundary in order to move the trigger inputs to the desired bit (c, d). In this example, model (c) and (d) will achieve similar accuracies but will behave differently when prompted with the trigger inputs.

mark datasets: MNIST [10], CIFAR-10 [16], and ImageNet [9] and three typical neural network architectures: a plain Convolutional Network [18], a Wide Residual Network [43], and a ResNet50 [14]. In all our experiments, we used a watermark key size of $K = 30$. We first list some benchmarking results and then discuss the robustness of our watermarking approach against common attacks.

5.1. Performance

Table 2 shows some quantitative results for each of the three benchmark datasets. We can see that embedding a watermark into a pretrained model tends to decrease its validation accuracy with up to 2% on the CIFAR10 and ImageNet dataset. While this slightly reduced accuracy might be acceptable in some cases, it is significantly higher for our approach than for the Blackmarks technique which typically results in a drop of around 0.1%. The main benefit of our approach is that it can generate a large number of watermarked instances, all using the same trigger set where a technique such as Blackmarks is limited to creating one watermarked model. Another benefit of our approach is its scalability. Creating a new watermarked instance takes only a few seconds for our approach where the watermarking procedure of Blackmarks is reported to be much larger with 2% to 8% of the total training time of training the model from scratch [6].

5.2. Robustness

As proposed in [26], we evaluate the robustness of our watermarks against common watermark removal attacks:

- **Fine-Tune (FT)**: All weights are fine-tuned using the original training set.
- **Retrain (RT)**: The last layer's weights are initialized randomly, and then all the weights are fine-tuned using the original dataset.
- **Adversarial training (AT)** [28]: A (sub)set of the dataset is used to generate adversarial samples. The model is then fine-tuned such that the model now returns correct predictions for these samples..
- **Fine-pruning (FP)** [24]: This method first prunes the least active nodes and then fine-tunes the model to regain its previous drop in accuracy.
- **Input Noising (IN)** [44]: Gaussian noise is added to the image before passing it through the model.
- **Input Quantization (IQ)** [22]: The input pixel values are quantized before passing the image through the model.

Figure 5 shows the results of our proposed watermarking approach when subjected to six different types of attacks, across all three datasets. In these graphs, the red line displays the watermark accuracy, which measures the proportion of correctly retained watermark bits following each attack. The blue lines depict the model's validation accuracy. We repeated all experiments 15 times and show the mean accuracies, together with a confidence interval of one standard deviation.

The most straight-forward attack is to simply fine-tune (FT) the model further on the original training set in an attempt to remove the effects of the watermarking procedure. The watermark is however relatively robust against this attack. With a key length of 30 bits, this attack results in at most 2 bits being flipped on the MNIST and CIFAR10 dataset and at most 3 bits on the Imagenet dataset. The retraining attack (RT) goes one step further and first initializes the weights of the last layer randomly before finetuning the model on the original training data. This attack is more effective, resulting in up to 3, 1 and 10 incorrect watermark bits on the MNIST, CIFAR10 and Imagenet dataset respectively.

The adversarial training attack is more sophisticated as it involves generating adversarial inputs and retraining the model to classify them correctly. While the CIFAR10 model seems to be quite robust against this attack, both the MNIST and Imagenet model are more sensitive with a 20%

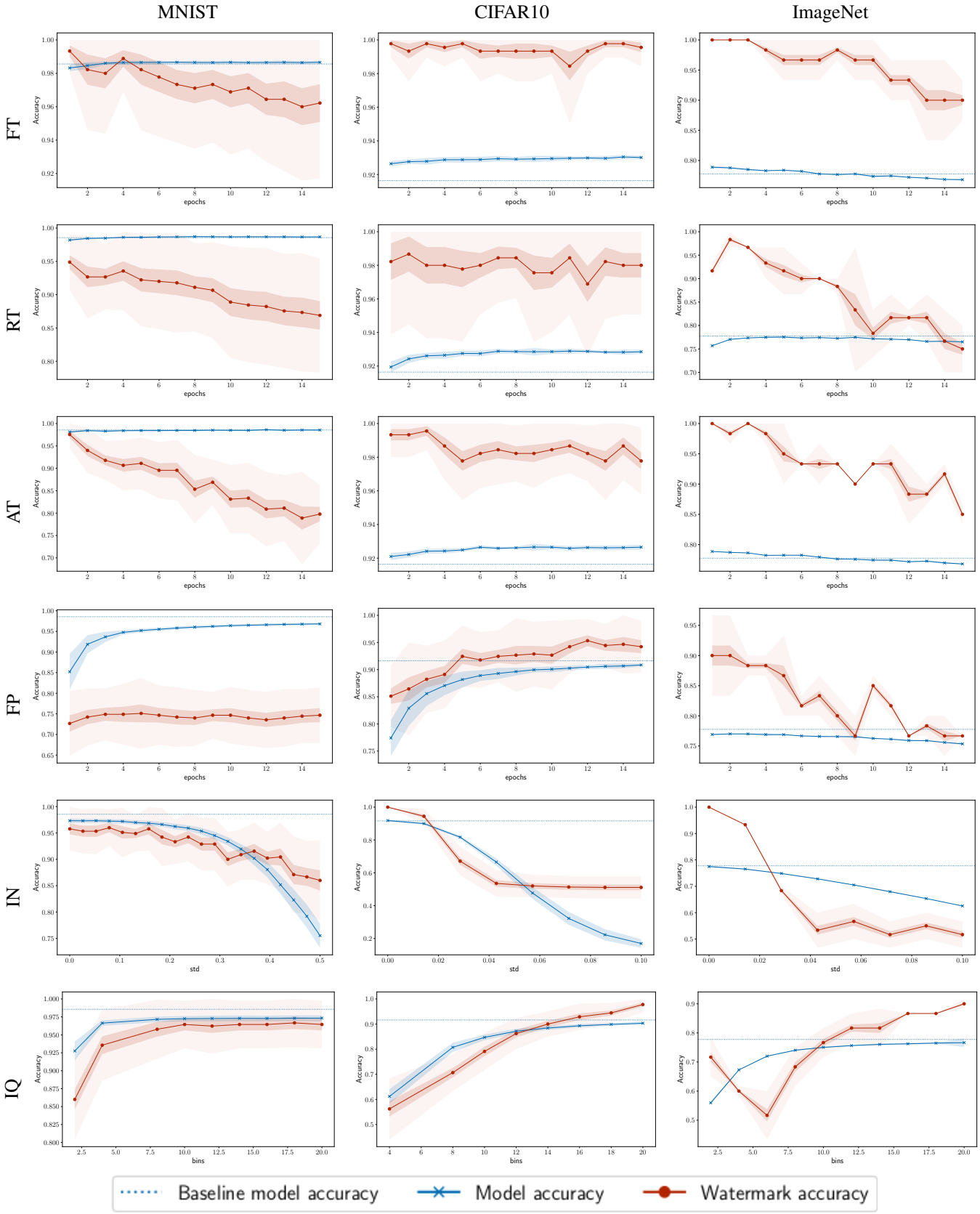

Figure 5. Watermark performance on four model modification attacks, on the best configurations found for MNIST, CIFAR-10, and ImageNet. the initial watermark accuracy is always one.

Dataset	MNIST	CIFAR10	ImageNet
Model	CNN	Wide Residual Network	ResNet50
Accuracy before watermarking	98.4 %	93.2 %	79.5 %
Accuracy after watermarking	$98.6 \pm 0.2\%$	$91.8 \pm 0.5\%$	$77.5 \pm 0.2\%$
Watermark embedding time	2s	8s	47s

Table 2. Illustration of an encoding scheme f for the CIFAR-10 dataset. Each label represents a bit, which is calculated by using K-means clustering on a training set.

drop in watermark accuracy (6 watermark bits out of 30 are now misclassified). A similar observation can be made for the Fine-pruning attack (FP). Here both the watermark and validation accuracy drop significantly after pruning. On the MNIST and CIFAR10 datasets, it is possible to regain some of the lost model accuracy but on the Imagenet dataset, the accuracy can never reach its original value.

The final two attacks don't alter the model itself but rather disrupt the input. These attacks are relatively straightforward to execute because they don't rely on gradient-based optimization and don't necessitate access to a representative dataset. In the case of the noise attack, we introduce random Gaussian noise with a mean of zero and an increasingly larger standard deviation (indicated on the x-axis) to the input image. This attack leads to the most significant decrease in watermark accuracy. On the Imagenet dataset, the watermark accuracy even plunges to just 50%, rendering the watermark ineffective. However, simultaneously, the model accuracy also experiences a significant decline, greatly diminishing its utility for the attacker.

The last attack quantizes the input into a predefined number of bins (shown on the x-axis). Similar to the noise attack, it is possible to reduce the watermark accuracy significantly but this also results in a severe drop of model accuracy.

6. Conclusion and Future Work

In this work, we addressed the challenge of safeguarding intellectual property in the form of neural networks in distributed edge AI environments. We introduced an innovative deep neural network watermarking technique that allows for the rapid generation of numerous watermarked model instances, each carrying a distinct embedded watermark. Unlike conventional methods, our approach permits the retrieval of the watermark using a uniform trigger set for all model instances, facilitating the identification of the user responsible for leaking the model. We demonstrated the robustness of our approach against typical watermark removal attacks. The proposed technique can be improved further by incorporating redundancy into the watermark bit strings to improve the robustness against watermark removal attacks.

Our hope with this work is to draw attention to the potential risks of IP infringement in distributed edge AI applications and to inspire further research into this area. We argue that model watermarking, while being a relatively new technology, could be part of the solution but it is unlikely to provide sufficient protection on its own in real-world applications.

Statements and Declarations

This work was funded by the AI Research Program from the Flemish Government (AI Flanders). The authors declare that there were no other sources of funding or competing interests.

References

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1615–1631, 2018. 2

[2] William Aiken, Hyoungshick Kim, Simon Woo, and Jungwoo Ryoo. Neural network laundering: Removing black-box backdoor watermarks from deep neural networks. *Computers & Security*, 106:102277, 2021. 3

[3] Franziska Boenisch. A systematic review on model watermarking for neural networks. *Frontiers in Big Data*, 4, nov 2021. 2, 3

[4] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Ipguard: Protecting the intellectual property of deep neural networks via fingerprinting the classification boundary. *CoRR*, abs/1910.12903, 2019. 3

[5] Abhishek Chakraborty, Ankit Mondai, and Ankur Srivastava. Hardware-assisted intellectual property protection of deep learning models. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020. 3

[6] Huili Chen, Bita Darvish Rouhani, and Farinaz Koushanfar. Blackmarks: Blackbox multibit watermarking for deep neural networks. *CoRR*, abs/1904.00344, 2019. 2, 3, 4, 6

[7] Haozhe Chen, Hang Zhou, Jie Zhang, Dongdong Chen, Weiming Zhang, Kejiang Chen, Gang Hua, and Nenghai Yu.

Perceptual hashing of deep convolutional neural networks for model copy detection. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(3):1–20, 2023. 3

[8] Huajie Chen, Tianqing Zhu, Chi Liu, Shui Yu, and Wanlei Zhou. High-frequency matters: An overwriting attack and defense for image-processing neural network watermarking. *arXiv preprint arXiv:2302.08637*, 2023. 3

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009. 6

[10] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 6

[11] Brunno F Goldstein, Vinay C Patil, Victor C Ferreira, Alexandre S Nery, Felipe MG França, and Sandip Kundu. Preventing dnn model ip theft via hardware obfuscation. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(2):267–277, 2021. 3

[12] Laurent Gomez, Alberto Ibarrondo, José Márquez, and Patrick Duverger. Intellectual property protection for distributed neural networks. 2018. 3

[13] Jiyang Guan, Jian Liang, and Ran He. Are you stealing my model? sample correlation for fingerprinting deep neural networks. *Advances in Neural Information Processing Systems*, 35:36571–36584, 2022. 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 6

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 4

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. 6

[17] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32:9233–9244, 2020. 2

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

[19] Jinsu Lee and Hoi-Jun Yoo. An overview of energy-efficient hardware accelerators for on-device deep-neural-network training. *IEEE Open Journal of the Solid-State Circuits Society*, 1:115–128, 2021. 3

[20] Sam Leroux, Pieter Simoens, Meelis Lootus, Kartik Thakore, and Akshay Sharma. Tinymlops: Operational challenges for widespread edge ai adoption. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1003–1010. IEEE, 2022. 2, 3

[21] Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness. *arXiv preprint arXiv:1904.08444*, 2019. 3

[22] Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness. *CoRR*, abs/1904.08444, 2019. 6

[23] Ning Lin, Xiaoming Chen, Hang Lu, and Xiaowei Li. Chaotic weights: A novel approach to protect intellectual property of deep neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(7):1327–1339, 2020. 3

[24] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21*, pages 273–294. Springer, 2018. 3, 6

[25] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. Sok: How robust is image classification deep neural network watermarking? In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 787–804. IEEE, 2022. 3

[26] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. Sok: How robust is image classification deep neural network watermarking? In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 787–804. IEEE, 2022. 6

[27] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. *arXiv preprint arXiv:1912.00888*, 2019. 3

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2017. 6

[29] Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. Darknetz: towards model privacy at the edge using trusted execution environments. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 161–174, 2020. 2

[30] Yuki Nagai, Yusuke Uchida, Shigeyuki Sakazawa, and Shin'ichi Satoh. Digital watermarking for deep neural networks. *International Journal of Multimedia Information Retrieval*, 7:3–16, 2018. 2

[31] Ryota Namba and Jun Sakuma. Robust watermarking of neural network with exponential weighting. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 228–240, 2019. 2

[32] April Pyone, Maung Maung, and Hitoshi Kiya. Training dnn model with secret key for model protection. In *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*, pages 818–821. IEEE, 2020. 3

[33] Masoumeh Shafieinejad, Nils Lukas, Jiaqi Wang, Xinda Li, and Florian Kerschbaum. On the robustness of backdoor-based watermarking in deep neural networks. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, pages 177–188, 2021. 2

[34] Raghubir Singh and Sukhpal Singh Gill. Edge ai: a survey. *Internet of Things and Cyber-Physical Systems*, 2023. 2, 3

[35] Yuchen Sun, Tianpeng Liu, Panhe Hu, Qing Liao, Shouling Ji, Nenghai Yu, Deke Guo, and Li Liu. Deep intellectual property: A survey. *arXiv preprint arXiv:2304.14613*, 2023. 2, 3

[36] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and

survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017. 3

[37] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pages 269–277, 2017. 2, 3

[38] Tianhao Wang and Florian Kerschbaum. Riga: Covert and robust white-box watermarking of deep neural networks. In *Proceedings of the Web Conference 2021*, pages 993–1004, 2021. 2

[39] XiangRui Xu, YaQin Li, and Cao Yuan. A novel method for identifying the deep neural network model with the serial number. *arXiv preprint arXiv:1911.08053*, 2019. 2

[40] Mingfu Xue, Shichang Sun, Can He, Y Zhang, J Wang, and W Liu. Activeguard: Active intellectual property protection for deep neural networks via adversarial examples based user fingerprinting. In *Proc. Int. Workshop Pract. Deep Learn. Wild (Workshop at AAAI)*, pages 1–7, 2022. 2

[41] Mingfu Xue, Yushu Zhang, Jian Wang, and Weiqiang Liu. Intellectual property protection for deep learning models: Taxonomy, methods, attacks, and evaluations. *IEEE Transactions on Artificial Intelligence*, 3(6):908–923, 2021. 2, 3

[42] Ziqi Yang, Hung Dang, and Ee-Chien Chang. Effectiveness of distillation attack and countermeasure on neural network watermarking. *arXiv preprint arXiv:1906.06046*, 2019. 3

[43] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. 6

[44] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, page 39–49, New York, NY, USA, 2017. Association for Computing Machinery. 6

[45] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia conference on computer and communications security*, pages 159–172, 2018. 2

[46] Yue Zheng, Si Wang, and Chip-Hong Chang. A dnn fingerprint for non-repudiable model ownership identification and piracy detection. *IEEE Transactions on Information Forensics and Security*, 17:2977–2989, 2022. 3