WaterMAS: Sharpness-Aware Maximization for Neural Network Watermarking

Carl DE SOUSA TRIAS[1][0009−0000−4700−1682], Mihai MITREA[1][0000−0003−4666−6847], Attilio FIANDROTTI[2,3][0000−0002−9991−6822], Marco, CAGNAZZO[4,3][0000−0001−6731−3755], Sumanta CHAUDHURI[3][0000−0002−8337−079X], and Enzo TARTAGLIONE[3][0000−0003−4274−8298]

[1] Telecom SudParis, Institut Polytechnique de Paris, Palaiseau, France
[2] Università degli Studi di Torino, Turin, Italy
[3] LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France
[4] Università degli Studi di Padova, Padua, Italy

Abstract. Nowadays, deep neural networks are used for solving complex tasks in several critical applications and protecting both their integrity and intellectual property rights (IPR) has become of utmost importance. To this end, we advance WaterMAS, a substitutive, white-box neural network watermarking method that improves the trade-off among robustness, imperceptibility, and computational complexity, while making provisions for increased data payload and security. WasterMAS insertion keeps unchanged the watermarked weights while sharpening their underlying gradient space. The robustness is thus ensured by limiting the attack's strength: even small alterations of the watermarked weights would impact the model's performance. The imperceptibility is ensured by inserting the watermark during the training process. The relationship among the WaterMAS data payload, imperceptibility, and robustness properties is discussed. The secret key is represented by the positions of the weights conveying the watermark, randomly chosen through multiple layers of the model. The security is evaluated by investigating the case in which an attacker would intercept the key. The experimental validations consider 5 models and 2 tasks (VGG16, ResNet18, MobileNetV3, SwinT for CIFAR10 image classification, and DeepLabV3 for Cityscapes image segmentation) as well as 4 types of attacks (Gaussian noise addition, pruning, fine-tuning, and quantization). The code will be released open-source upon acceptance of the article.

Keywords: Watermarking · Neural Networks · Sharpness-aware optimisation · IPR.

1 Introduction

The deployment of deep neural networks became massive for both industrial and end-user-oriented applications. Such tasks are instantiated in a wide variety

of application domains including but not restricted to image/video classification [24,25], object detection [41], speech recognition and synthesis [36], and audio-visual content compression [32]. Furthermore, deep neural networks can also serve the purposes of critical tasks, such as autonomous driving for unmanned vehicles [7]. Coming across with the effort to make such models more and more efficient in their tasks, the interest in protecting their intellectual property rights (IPR) and in verifying their integrity emerged some 7 years ago. On the one hand, these models are costly in terms of human skills and computing resources, and protecting their intellectual rights is not only an ethical issue but also an economic one. On the other hand, such deep models can voluntarily or involuntarily be corrupted, resulting in the malfunctioning of the system. Watermarking can be a solution in both cases.

Inherited from the multimedia realm [9], watermarking regroups a family of methodological and applicative tools allowing for imperceptible and persistent insertion of some metadata (watermark) into original content, according to a secret key and under some prescribed security performances. The subsequent watermark detection can serve various purposes among which the most relevant for our study is IPR management, understood as the possibility of unambiguously identifying semantically similar yet digitally different contents, like those obtained after compressing or cropping a video sequence, for instance. The main properties of watermarking are the data payload, the imperceptibility, and the robustness. The *data payload* represents the quantity of information (in bits) that can be inserted and detected for serving the targeted applicative scope (copyright and/or integrity certification). The *imperceptibility* refers to the preservation of the quality of the original content. The *robustness* refers to the property of recovering the mark even when the protected content was subjected to malicious or mundane operations (commonly referred to as attacks). The security property relates to the watermarking system behavior when some partial or total information about the key is available to the attacker.

Watermarking solutions can also be designed and deployed for deep neural networks [40,1,28]. To this end, the generic watermarking properties are reconsidered and extended. First, the data payload concept is kept unchanged, yet the practice results in one-bit solutions (a binary, detected/undetected decision is generally made). Secondly, the imperceptibility property is evaluated by comparing the applicative performances of the watermarked model to the ones provided by a watermarked model trained in similar conditions. Thirdly, the robustness is assessed against transformations involved in the neural network life-cycle, like pruning or quantization, for instance. Finally, the watermark could be either retrieved from the parameters of the model (*white-box* methods [40,5,21,39]) or from its inference (*black-box* methods [1,43]). The neural network watermarking method security is not specifically studied and by default considered to be linked to the key size (as an attempt to avoid the brute force attack) and to its impossibility to be known by the attacker.

In this study, we introduce a new neural network watermarking method that belongs to the white-box category. On the one hand, from the conceptual point

of view, we establish synergies between the very concept of neural network optimization and neural network watermarking. From the neural network optimization point of view, we use the high dimensionality of models to lock a subset of weights without impacting the final performance of the model. To this end, the findings in [13] are leveraged for watermarking purposes, where the sharpness of the loss landscape for the watermarked weights should be maximized. This way, the watermark can be inserted through the entire model, without exploiting any of its peculiarities (architecture, activation function, regularization term, etc.). By considering now that deep neural networks are typically over-parameterized, the watermark size can be thus significantly extended, virtually till the limit set by the model redundancy [29,14,31]. On the other hand, from the neural network watermarking point of view, we insert a watermark (represented as an image) in a subset of weights. The robustness property is obtained by the inference sensitivity on the watermarked weights: the force of the attacks that can be applied on the watermarked weights is implicitly reduced by the imperceptibility property. The same sensitivity also makes the method reach extreme security: even assuming the attacked identifies the key, they cannot modify/erase the watermark without destroying the model inference quality (this *a priori* consideration being experimentally proved in Table 4).

The main contributions of this study can be summarized as follows.

1. Starting from the SAM (Sharpness-Aware Minimization [13]) setup, where the loss landscape is enforced being maximally flat, we state and solve the inverse problem (Sharpness-Aware Maximization - further referred to as MAS) (Sec. 3) where the loss landscape can be made steep.
2. We define a new neural network watermarking method, referred to as WaterMAS, that leverages the MAS principle for turning the produced inference intrinsically sensitive to the watermarked weights (Sec. 4).
3. We carry out a comprehensive set of experiments on conventional neural network watermarking properties (imperceptibility, robustness, and computational complexity) as well as a discussion on data payload and method security (Sec. 5).

The paper is structured as follows. Sec. 2 starts with an analysis of the basic concepts for neural network watermarking and exemplifies the key white-box neural network watermarking methods: insertion methods, types of inserted information, secret key, functional properties, and different attacks. It follows by introducing the sharpness-aware minimization grounds. The next Sec. 3 presents the first contribution of the paper, namely the definition of the sharpness maximization problem and the deriving of the underlying algorithm solving this problem. Sec. 4 presents a new neural network watermarking method based on the findings in Sec. 3, as illustrated in Fig. 1. The experimental Sec. 5 starts by imperceptibility evaluation against different methods of the literature, on different tasks and architectures, then assesses the robustness, and lastly, explores the size of information that can be inserted in a specific model and its relationship with security aspects. Finally, Sec. 6 concludes this work and opens perspectives for future works.

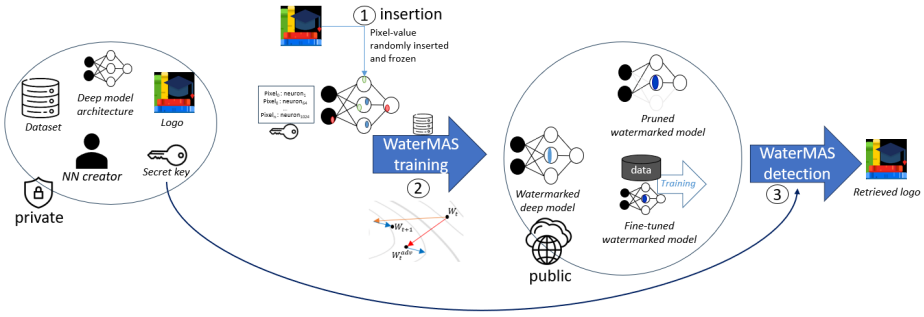


Fig. 1: Usage of WaterMAS for sharing a watermarked neural network model, while tracking its usage.

2 Related Works

Watermarking tools allow imperceptible and persistent insertion of some metadata into original content. The neural network watermarking field emerged in 2017 with the work of Uchida et al. [40], followed by Adi et al. [1] and Zhang et al. [43] in 2018, thus establishing the earliest taxonomy: white-box *vs* black-box watermarking. In the white-box case, the watermark is retrieved from the parameters of the model while the black-box scenario corresponds to the situation where the watermark is retrieved from the inferences of the model.

2.1 Neural network watermarking

Uchida et al. [40] is a white-box watermarking method inserting the watermark into an arbitrarily selected layer, by using a regularization term that projects the parameter on a space (the secret key) a binary watermark. The regularization term is obtained by computing the binary cross-entropy between the projected weights on the secret space and the targeted watermark. At the detection phase, the watermarked layer is projected on the secret space to obtain the watermark. To meet the imperceptibility criterion, the regularization term is weightily added to the loss of the original task. The robustness is expected to be met by the design of the term which spreads the information of the watermark within all the weights of the layer. In other words, this method implicitly considers that thanks to the regularization term, no matter how a modification would be applied on the watermarked layer, the watermark will be detected as far as the inference would not be severely downgraded. The imperceptibility is evaluated by comparing the final accuracy of un-watermarked and watermarked models. The robustness is assessed against fine-tuning (performed as additional training for half the number of embedding/training epochs) and magnitude pruning (setting to zero up to 95% of the layer). Methods with similar principles to [40] can be found in [5,27]. For instance, in [5] the projection is done on the output of the layer

instead of its weights. In [27], spread-transform dither modulation is considered as the insertion side.

Tartaglione et al. [39] follows a different approach. It no longer considers a specific layer but randomly selects a set of parameters to be watermarked. For the insertion procedure, the pixel values of an image (watermark) are inserted and frozen in randomly selected weights: the correspondences among the pixels in the watermark and the locations of the parameters are kept secret (and represent the key). The design of this method is meant to ensure sensitive watermarked weights: their small variation will impact the performance of the model. During training, at each step, R replicas are created by adding noise to the watermarked weights, and the loss of the R replicas is maximized, thus acting as a regularization term. Hence, such regularization terms can be weightily added to the loss of the original task to meet the imperceptibility criteria. The robustness criterion is ensured by the replicas which randomly explore the loss landscape around the watermarked weights and maximize it. During training, the watermarked parameters are frozen and the original cost function is computed. At the detection phase, the watermarked parameters are retrieved by looking at their location. Note that the robustness of [39] differs from the one in [40] in its very nature: this time, the potential force of the modifications of the watermarked weights is restricted, as it would implicitly downgrade the inference performance. The imperceptibility is evaluated by comparing the final accuracy of un-watermarked and watermarked models, while the robustness is assessed against fine-tuning (performed as additional training for half the number of embedding/training epochs) and quantization (reducing the bits representation of the weights).

Li et al.[26] embeds a binary watermark in the sign of the most significant weights of a model. For the insertion procedure, the binary watermark of length m is mapped to $\{-1, 1\}_m$ and modulated by a pseudo-random generator. The selected neurons are chosen according to three pruning methods namely: network slimming, efficient filter, and entropy. In detail, the network slimming method selects the neurons according to the sum of their absolute parameters value, the efficient filter method selects the neurons according to the magnitude of the scale in the batch norm layer, and the entropy method selects the neurons according to the entropy value of their output for a subset of inputs. Hence, each method selects r neurons, then the final list of selected neurons is obtained by intersecting the three obtained lists, finally, m weights are randomly selected to carry the watermark, their position is kept secret (and represents the key). The mark is inserted in the sign of the parameters with the highest $\mathcal{L}_1$-norm of the selected neurons. After inserting the watermark the model is fine-tuned for an additional 10 epochs without constraint on the training. At the detection phase, the selected parameters are retrieved and their sign is recovered, modulation is reversed to obtain the watermark. To evaluate imperceptibility the final accuracy of watermarked and unwatermarked models is compared while the robustness is evaluated against fine-tuning, pruning, and watermark overwriting.

Lv et al.[33] embeds the weight of the encoder of an autoencoder model (HufuNet) in the watermarked model. First, the HufuNet model is trained to reconstruct images of a given dataset. For the insertion procedure, the position of the watermarked weights is obtained by computing the hash function over the Decoder (secret key) while the parameter values of the encoder are inserted. During training, the model is trained according to a specific loss that constrains the evolution of the parameter. Since the watermarked weights might evolve, in each n epoch the watermark parameters are retrieved and the retrieved HufuNet model is fine-tuned with a frozen decoder until reaching back performance, the watermarked weights are inserted again, and the training of the watermarked model resumed. At the detection phase, the watermark parameters are extracted using the Decoder, and the reconstructed HufuNet is evaluated on its tasks. The imperceptibility criteria is evaluated by comparing unwatermarked and watermarked models while the robustness is evaluated against fine-tuning, transfer learning, and pruning.

2.2 Sharpness-aware minimization

Model generalization in deep learning is a critical area of research, recurrently addressed by the Deep Learning community [3,35]. Various advanced techniques have been created, focusing either on adjusting the model itself [19] or on enhancing the dataset through augmentation [20,42]. The connection between flat regions in the loss landscape and the underlying model generalization capability was brought forth back already in 1995 [17] and it has been empirically studied afterward [23,35]. Subsequently, multiple approaches leading to maximally flat regions have been identified, in the seek for better generalization [4,34,13].

In the last few years, Sharpness-Aware Minimization (SAM) was advanced with the goal of being both efficient and effective in enforcing flatness for the achieved solutions, employing a local linear approximation for the loss [13]. Starting from this, other variants of SAM have been proposed, achieving outstanding results in several demanding tasks which include continual learning [11] and federated learning [2,10].

In our work, we are building on top of [13] for considering the shape of the loss landscape in an efficient way. Specifically, we like to ensure *sharpness* for a subset of parameters where we embed our watermark. Unlike approaches like [39] that rely on memory and computation-wise expensive sampling process, we leverage the locally linear approximation of the loss to already find the "worst-case" direction of the gradient in the watermarked parameters space.

Table 1: Table of notations.

$X \in \mathbb{R}^N$	the watermark
x_i	a single element of the watermark
W	The parameters of the model
$W_X \subset W$	a subset of the parameters
$\overline{W}_X$	the complementary subset
$\mathcal{L}$	the loss of the model
∇	the gradient
η	the learning rate

3 Sharpness-aware optimization

This section starts by reconsidering the SAM algorithm presented in [13] and designed to flatten the loss landscape of all the weights of the model for increasing the model generalization. Then, our algorithm for Sharpness-aware Maximization **MAS** is introduced: it aims at sharpening the loss landscape of a subset of weights which will serve in the next section 4 to carry the watermark. Notations are introduced in Table 1.

3.1 Sharpness-aware minimization (SAM)

In [13], a sharpness-aware minimization optimization algorithm (SAM) was advanced to achieve model generalization by using only two gradient computations per iteration, as synoptically presented in Figure 2. The first gradient is computed from all the parameters of the model ($\nabla_W \mathcal{L}$). Then, W_t^{adv} is obtained by maximizing $\mathcal{L}$, i.e. by following the gradient $\nabla_W \mathcal{L}$ properly scaled using ρ:

$$W_t^{adv} = W_t + \frac{\rho}{\|\nabla_W \mathcal{L}\|_2} \nabla_W \mathcal{L}. \quad (1)$$

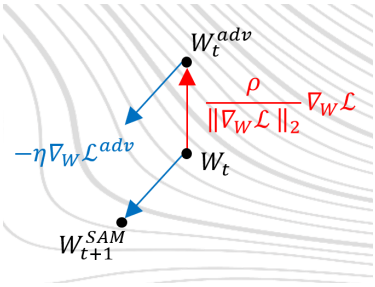


Fig. 2: Schematic of the parameter update in SAM method.

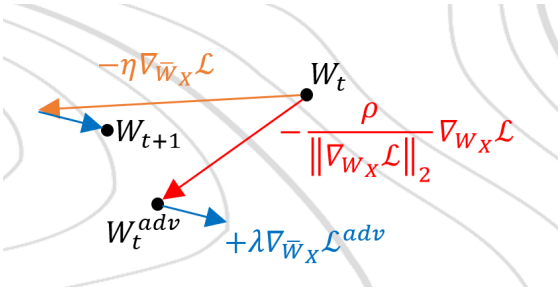
This way, W_t is projected to the point, on the hypersphere of radius ρ centered to W_t, where the loss is maximal (under the assumption the loss is locally linear, meaning that this assumption is valid for low values of ρ). The objective of SAM is to ensure the local flatness of the loss: by minimizing the loss at W^{adv}, this can be successfully achieved. This gradient is the projected to W_t:

$$W_{t+1}^{SAM} = W_t - \eta \nabla_W \mathcal{L}^{adv}. \quad (2)$$

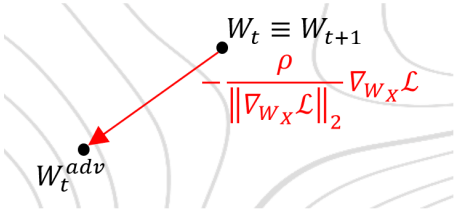
Empirical results on multiple common architectures, including ResNets and state-of-the-art image classification datasets, validate the approach and quantitative analyses demonstrate an improved achieved flatness. In the next subsection, we will build on top of this idea to enforce sharpness on a subset of parameters.

3.2 Sharpness-aware maximization (MAS)

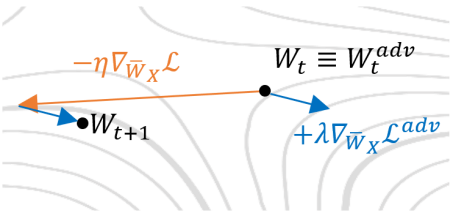
Let us split the set of parameters W belonging to our model in two distinct subsets of weights: W_X, consisting of some frozen parameters to which we want to enforce sharpness for a target loss $\mathcal{L}$ (while still minimizing it), and $\overline{W}_X = W \setminus W_X$. The choice of the loss function depends on the specific problem the model is designed to address, *eg.* cross-entropy for classification models. The objective is thus to drive the subset of weights W_X (by properly modifying $\overline{W}_X$ only) to a



(a) Schematic of the parameter update in MAS.



(b) Projection in the space W_X.



(c) Projection in the space $\overline{W}_X$.

Fig. 3: Schematic of the parameter update in MAS method for the different loss landscape (a) W (b) W_X and (c) $\overline{W}_X$.

sharp region. We follow a three-step computation strategy, which is summarized in Fig. 3a.

First, we solely minimize the loss $\mathcal{L}$ wrt. W_X, by projecting W_t to an hypersphere of radius ρ limitedly to the subspace W_X, thus obtaining W_t^{adv}:

$$W_t^{adv} = W_t - \frac{\rho}{||\nabla_{W_X}\mathcal{L}||_2}\nabla_{W_X}\mathcal{L}. \tag{3}$$

This projection is represented in Fig. 3b. Note that in the space $\overline{W}_X$, the projections of W_t and of W_t^{adv} are at the same point.

Then, we aim at maximizing the loss on W_t^{adv}. This way, we are enforcing that, at a given distance ρ, the loss increases when moving in the W_X space.

Finally, we join the adversarial loss maximization term computed in the previous step with the traditional loss minimization on $\overline{W}_X$ (as shown in Fig. 3c):

$$W_{t+1} = W_t - \eta\nabla_{\overline{W}_X}\mathcal{L} + \lambda\nabla_{\overline{W}_X}\mathcal{L}^{adv}. \tag{4}$$

In conclusion, the model will minimize the loss of the given task while aiming for a sharpened region of the W_X' loss landscape. In the next section, (4) will be leveraged for watermark insertion: the non-watermarked weights can be updated while minimizing the loss function.

4 WaterMAS

This section advances a new neural network watermarking method **WaterMAS**. It consists of three steps: (i) selecting the parameters of the model that will carry the watermark (note that unlike most of the white-box neural network watermarking methods, in WaterMAS, the watermarked weights can be located in the whole model rather than in a specific layer); (ii) inserting the watermark in those parameters, and (iii) ensuring the trade-off between robustness and imperceptibility through training. This section is organized as follows. First, information related to data payload, secret information, and weight selection is presented in Sec. 4.1. Secondly, the very training method is advanced by using principles introduced in Sec. 3.2. Finally, the detection issues are discussed in Sec. 4.3.

4.1 Data payload, secret information, and weight selection

WaterMAS inserts an image of size $Height \times Width \times 3$ (X) obtained by shifting the integer pixels values towards floating point $x_i \in [0, 1]$. The underlying data payload is 1-bit (image inserted or not) but the impact and the possibility of the different image size will be discussed in Sec. 5.4. This image is to be inserted in some model parameters that should be selected in a manner that makes them indistinguishable from the other parameters. To achieve this, two mechanisms will be considered. The first mechanism consists of randomly sampling W_X parameters to carry the watermark throughout the whole model; this way, a random association between the i-th element of the watermark x_i and the j-th parameter of the neural network w_j is established. This association serves as a secret key. The second mechanism refers to the way in which these parameters are modified so as to carry the watermark. Assuming, as presented in [15], that all the parameters in the model are distributed according to a Gaussian distribution with mean μ and standard deviation σ, each inserted element of the watermark follows the equation:

$$w_j = 2\sigma(x_i - 0.5) + \mu \quad \forall\ w_j \in W_X \tag{5}$$

4.2 Training procedure

The training procedure is designed so as to (1) keep the watermarked parameters unchanged (so as to preserve the watermark) and (2) to maximize their sharpness (so as to turn the inference very sensitive on those values). This is achieved by using **MAS** introduced in Sec. 3.2 and according to the **while** loop of Algorithm 1. The two subsets of weights are obtained after the selection and insertion occurred (lines 1&2) and each iteration of training consists of four steps. (i) A first gradient is computed on the model W_t. (ii) W_t^{adv} is obtained by updating the watermarked weights W_X in the direction that minimize the loss. (iii) A second gradient is computed on this obtained model W_t^{adv} and maximized. (iv) The second obtained gradient is added using a hyperparameter λ (strength of the

Algorithm 1 WaterMAS algorithm

Require: $\mathcal{L}$(the loss function) x_i, y_i(a pair of inputs, labels) X(the watermark)
1: Select a subset of weight to carry the watermark $\boldsymbol{W}_X$
2: Substitute the value of $\boldsymbol{W}_X$ by X using a mapping function M (kept secret)
3: **while** Not converged **do**
4: **for** $\forall x_i, y_i$ **do**
5: Compute gradient on the model $\nabla_W \mathcal{L}$
6: Obtain W_t^{adv} by only updating the watermarked weights using (3)
7: Compute gradient on the adversarial model$\nabla_{\overline{W}_X} \mathcal{L}^{adv}$
8: Add the obtained gradient in 7 to the original cost function
9: Update only the non-watermarked weights according to as in (4)
10: **end for**
11: **end while**

additional terms) and applied only to the non-watermarked weights to obtain W_{t+1}. Hence, the watermarked parameters do not change during the training process while aiming to shrink the loss landscapes, making them sensitive in inference.

4.3 Detection procedure

The extraction of the watermark is obtained by retrieving the value of the watermarked weights and by inverting the association between those weights and the inserted image. This extracted image can be compared to the inserted image while using qualitative (human decision) or quantitative (Pearson's correlation). In the end, a binary decision is made (that is, whether the recovered and inserted images are identical or not), and the data payload is 1 bit.

In this section, we have introduced a new watermarking method that inserts a watermark by substituting the value of the weights according to secret information only dependent on the number of elements in the watermark (Sec. 4.1). During training, the watermark weights are frozen and the *MAS* term is used to enhance the robustness of those watermarked parameters (Sec. 4.2). In the next section, an experimental study asses the WaterMAS performances.

5 Experimental results

In this section, WaterMAS performances are experimentally assessed. The number of watermarked weights W_X remains fixed for the imperceptibility and robustness sections, namely $32 \times 32 \times 3$. We empirically set $\rho = 10^{-2}$ and $\lambda = 10^{-5}$ through a grid-search evaluated on a ResNet-18 trained on CIFAR-10. WaterMAS is benchmarked against four state-of-the-art methods, namely [40,39,26,33], with their reference hyperparameters; [40,39,33] codes were available while [26] code has been implemented for the present study purposes.

5.1 Testbed

The results are performed on 4 architectures for classification, namely ResNet18 [16], VGG16 [22], MobileNetV3s [18], and SwinT [30] on CIFAR-10 [24], and on one architecture for image segmentation, namely DeepLabV3 [6] on Cityscapes [8]. Note that throughout the experiments, we tried to impose reproducible conditions by setting the seed; yet, the speculative execution of CUDNN often introduces undesirable sources of randomness negligible to the final performance evaluation [37]. For performance evaluation of the task, the complementary accuracy (acc) (†) defined as $(1 - \text{acc})$ is used for the classification tasks, and the complementary mean Intersection over Union (mIoU) (‡) defined as $(1 - \text{mIoU})$ is used for the image segmentation task. For the robustness evaluation, 4 removal attacks are considered:

1. **Gaussian noise addition**: a random noise is added to the model to impact the watermark. Our hyperparameter $S \in [1, 5]$ corresponds to the ratio between the standard deviation of the added noise to the standard deviation of the aimed layer while both means are equal.
2. **Fine-tuning**: the training of the model is resumed for some additional epochs $E \in [1, 25]$.
3. **Magnitude pruning**: a portion $P \in [0.1, 0.9]$ of neurons as set to zero according to their $L1$-norm. This attack aims to compress the model.
4. **Quantization**: compress the model by reducing the number of bits $B \in [2, 16]$ of the floating representation of the parameters.

Finally, a discussion about the security, data payload, and computational complexity is devoted in the last section presenting a cryptography attack.

5.2 Imperceptibility evaluation

Table 2: Imperceptibility evaluation. Each model has been initialized and trained with the same hyperparameters. All experimental values are multiplied by 100.

Method	**CIFAR10**				**Cityscapes**
	VGG16	ResNet18	MobileNet	SwinT	DeepLab-v3
Unwatermarked	*9.56*†	15.69†	*27.11*†	*16.28*†	*29.52*‡
Uchida et al. [40]	*9.68*†	*13.50*†	*27.16*†	*21.76*†	*36.23*‡
Tartaglione et al. [39]	*9.91*†	*12.77*†	*25.27*†	*18.47*†	*30.14*‡
Li et al. [26]	*9.63*†	*15.43*†	*25.38*†	*21.07*†	*29.84*‡
Lv et al. [33]	*14.80*†	*19.60*†	*33.90*†	$\star$	$\star$
WaterMAS	*10.64*†	*13.58*†	*26.34*†	*21.15*†	*29.90*‡

Imperceptibility evaluates the impact of the watermark insertion in the inference. In neural network watermarking, some insertion methods act through

the training, thus making the creation of the model intrinsically linked to the watermark. Hence, the imperceptibility evaluation is done by comparing the results of watermarked and unwatermarked models. For all the setups, each model has been initialized with the same seed, trained for the same number of epochs, 200, using an SGD optimizer with $lr = 0.1$ (except for VGG16 $lr = 0.01$ and SwinT $lr = 0.001$), momentum= 0.9, weight decay=$10 - 4$, and a scheduler which divide the learning rate by 10 after epoch 100 and 150. The objective was not to obtain state-of-the-art accuracy but rather to show the impact of the different methods with an identical training setup (in an end-user setup, the hyperparameter can be fine-tuned for each setup to increase the performance of the models). The results are displayed in Table 2 and show similar imperceptibility for all the methods. For Uchida's method on two setups, namely SwinT and DeepLab, we can observe that the regularization term has a stronger impact on the performance, either positively or negatively. For Lv's method two configurations were not implemented (SwinT and DeepLab), as indicated by $\star$. For VGG16, ResNet18, and MobileNet, the watermarking procedure impacts the performance of the model, indicating that this method needs specific tuning of hyperparameters depending on the configuration.

Table 3: Robustness evaluation of WaterMAS against four removal attacks. The performance metric is multiplied by 100.

		CIFAR10				**Cityscapes**
		VGG16	ResNet18	MobileNet	SwinT	DeepLab
Gaussian	$\boldsymbol{S}=0$	*10.64*†	*13.58*†	*26.34*†	*21.15*†	*29.9*‡
	$\boldsymbol{S}=1$	*10.68*†	*13.66*†	*26.823*†	*21.30*†	*30.02*‡
	$\boldsymbol{S}=2$	*10.61*†	*13.56*†	*27.00*†	*21.13*†	*30.30*‡
	$\boldsymbol{S}=5$	*10.92*†	*23.29*†	*27.14*†	*21.49*†	*36.07*‡
Pruning	$\boldsymbol{P}=1$	*10.66*†	*13.65*†	*26.75*†	*20.98*†	*30.02*‡
	$\boldsymbol{P}=2$	*10.76*†	*14.05*†	*26.97*†	*21.73*†	*65.24*‡
	$\boldsymbol{P}=5$	*12.08*†	*14.91*†	*44.47*†	*29.98*†	*97.35*‡
	$\boldsymbol{P}=9$	*87.80*†	*78.39*†	*89.89*†	*86.20*†	*99.98*‡
Fine-tuning	$\boldsymbol{E}=0$	*10.64*†	*13.58*†	*26.34*†	*21.15*†	*29.9*‡
	$\boldsymbol{E}=5$	*9.76*†	*12.98*†	*24.88*†	*20.88*†	*28.5*‡
	$\boldsymbol{E}=10$	*9.73*†	*12.25*†	*25.38*†	*20.06*†	*27.77*‡
	$\boldsymbol{E}=25$	*9.58*†	*11.12*†	*25.43*†	*20.60*†	*27.21*‡
Quantization	$\boldsymbol{Q}=16$	*10.64*†	*13.58*†	*26.34*†	*21.15*†	*29.9*‡
	$\boldsymbol{Q}=8$	*11.78*†	*14.22*†	*26.56*†	*21.63*†	*30.10*‡
	$\boldsymbol{Q}=4$	*83.89*†	*36.64*†	*35.89*†	*89.26*†	*73.22*‡
	$\boldsymbol{Q}=2$	*90.00*†	*89.52*†	*86.49*†	*89.33*†	*97.33*‡

5.3 Robustness evaluation

Robustness evaluates the detector's capacity to retrieve the watermark of a watermarked content that has been altered. In Table 3, the watermarked neural network has been altered by the four attacks described in Sec. 5.1. The columns correspond to the different setups (dataset and architecture) while the lines are labeled by the attack parameter and provide the performance metric of the task when the watermark has been retrieved. The Table is entirely filled since the watermark can be retrieved even if the performance of the watermarked model is very low (for instance, quantization for the values 2 and 4 bits).

5.4 Security, data payload and computational cost

Table 4: Impact on the performance of removing the watermark using the secret key, depending on the number of watermarked weights. Values corresponds to the absolute variation of the complementary accuracy multiplied by 100.

	Method	$\lvert\mathbf{W_x}\rvert(\uparrow)$					
		768	3072	12288	49152	196608	786432
VGG	Uchida et al. [40]	*+0*	*+0*	*-0.01*	⋆	⋆	⋆
	Tartaglione et al. [39]	*+6.39*	*+5.15*	*+8.33*	*+7.04*	⋆	⋆
	Li et al. [26]	*+0*	*+0*	*+0*	*+0*	*+0*	*+0*
	Lv et al. [33]	*+0*	*+0*	*+0*	*+0*	*+0*	*+0*
	WaterMAS	*+0.34*	*+1.38*	*+4.19*	*+47.46*	⋆	⋆
ResNet	Uchida et al. [40]	*+ 0*	⋆	⋆	⋆	⋆	⋆
	Tartaglione et al. [39]	*+29.75*	*+24.93*	*+26.58*	*+22.15*	*+25.94*	*+25.96*
	Li et al. [26]	*+0*	*+0*	*+0*	*+0*	*+0*	*+0*
	Lv et al. [33]	*+0*	*+0*	*+0*	*+0*	*+0*	*+0*
	WaterMAS	*+0.04*	*+3.86*	*+28.49*	*+68.63*	*+68.87*	*+66.28*
MobileNet	Uchida et al. [40]	*+0.04*	⋆	⋆	⋆	⋆	⋆
	Tartaglione et al. [39]	*+44.57*	*+34.47*	*+38.58*	*+37.51*	*+34.47*	⋆
	Li et al. [26]	*+0*	*+0*	*+0*	*+0*	*+0*	*+0*
	Lv et al. [33]	*+0*	*+0*	*+0*	*+0*	*+0*	*+0*
	WaterMAS	*+3.24*	*+10.16*	*+50.01*	*+62.57*	*+55.98*	⋆
SwinT	Uchida et al. [40]	*-0.03*	*-0.17*	*-0.56*	*+0.04*	⋆	⋆
	Tartaglione et al. [39]	*+7.42*	*+11.16*	*+17.39*	*+25.46*	*+34.09*	⋆
	Li et al. [26]	*+0*	*+0*	*+0*	*+0*	*+0*	*+0*
	Lv et al. [33]	⋆	⋆	⋆	⋆	⋆	⋆
	WaterMAS	*+4.21*	*+36.64*	*+51.48*	*+64.2*	*+63.45*	⋆

Let's explore the number of watermarked weights that can convey the watermark. The results presented in Table 4, show the absolute variation of the performance when altering the watermarking weights. The watermark is destroyed in all scenarios while ⋆ indicates that the watermark could not even be inserted. [40] was designed to fit the size of the inserted watermarked to the size of the layer. Even if this adaptation is possible, the watermark can be removed with a low impact on the performance. For [26], the identified neuron can be multiplied

by -1 while its corresponding input channel of the next layer will also be multiplied by -1 resulting in an identic output value while the recovered watermark is destroyed. For [33], the access to the HufuNet decoder allows the attacker to fine-tune the HufuNet model to avoid watermark detection without any modification on the watermarked model. For [39], the performances of the model are lost when the watermark is removed. WaterMAS has a similar behavior as [39], but it also features an inverse linear dependency between the performance and the size of the removed watermark. This behavior opens the door for a larger data payload to be inserted by WaterMAS: while our experimental study does not explore this direction, the large span between the minimal and the maximal sizes can be exploited for increasing the data payload.

Table 5: Watermark insertion computational cost: values extracted during the training, for the same batch size and architecture. MiB - mebibyte (2^{20} Bytes).

Method		**CIFAR10**				**Cityscapes**
		VGG16	ResNet18	MobileNet	SwinT	DeepLab-v3
Unwatermarked	GPU Memory	858 MiB	502 MiB	622 MiB	1786 MiB	6861 MiB
	time/epoch	12”	12”	13”	40”	4’20”
Uchida et al. [40]	GPU Memory	858 MiB	502 MiB	622 MiB	1786 MiB	6877 MiB
	time/epoch	27”	49”	32”	1’14”	4’25”
Tartaglione et al. [39]	GPU Memory	4556 MiB	592 MiB	655 MiB	2122 MiB	7103 MiB
	time/epoch	1’33”	1’36”	8’32”	10’18”	16’10”
Lv et al. [33]	GPU Memory	1855 MiB	1354 MiB	736 MiB	$\star$	$\star$
	time/epoch	3’33”	4’06”	6’13”	$\star$	$\star$
WaterMAS	GPU Memory	1078 MiB	656 MiB	634 MiB	2074 MiB	7043 MiB
	time/epoch	27”	19”	57”	1’54”	8’44”

In Table 5, the memory footprint and the mean time of execution per epoch are presented for the four methods [40,39,33], the standard deviation does not appear since it was less than 1 second. [26] does not appear in the table since there is no constraint on the training.

6 Conclusion

With this paper, a new white-box watermarking method, referred to as WaterMAS, is advanced to reach the trade-off among robustness, imperceptibility, computational complexity, and data payload. First, by reconsidering and extending the SAM principles [13], a new regularisation term is designed for sharpening the watermarked weights landscape. This way, the strength of the attacks that can be applied to WaterMAS is intrinsically reduced, and the robustness is ensured. Second, the imperceptibility property is reached by tightly coupling this regularization term with the training process. Third, the extra computational complexity required is one back-propagation step. Finally, the insertion

occurs before training by randomly selecting a set of weights throughout the entire model, irrespective of the structures of layers. Compared to state-of-the-art methods [40,26,33], the main advantage is represented by the security, expressed as the possibility of keeping the watermark even when the secret key is intercepted, as presented in Table 4. Compared to [39,33], the main advantage is the computational cost reduction. Beyond watermarking, MAS opens the road to further explorations and applications of finding solutions in sharp loss minima, which can lead to sparse neural network representations [38] or even further exploration of properties of these minima [12].

References

1. Adi, Y., Baum, C., Cisse, M., Pinkas, B., Keshet, J.: Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In: 27th USENIX Security Symposium (USENIX Security 18) (2018)
2. Caldarola, D., Caputo, B., Ciccone, M.: Improving generalization in federated learning by seeking flat minima. In: European Conference on Computer Vision. Springer (2022)
3. Caruana, R., Lawrence, S., Giles, C.: Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. Advances in neural information processing systems **13** (2000)
4. Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., Zecchina, R.: Entropy-sgd: Biasing gradient descent into wide valleys. Journal of Statistical Mechanics: Theory and Experiment **2019**(12) (2019)
5. Chen, H., Rouhani, B.D., Fu, C., Zhao, J., Koushanfar, F.: Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In: Proceedings of the 2019 on International Conference on Multimedia Retrieval (2019)
6. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: European Conference on Computer Vision (2018)
7. Chng, Z.M., Lew, J.M.H., Lee, J.A.: Roneld: Robust neural network output enhancement for active lane detection. In: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE (2021)
8. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2016)
9. Cox, I., Miller, M., Bloom, J., Fridrich, J., Kalker, T.: Digital watermarking and steganography. Morgan kaufmann (2007)
10. Dai, R., Yang, X., Sun, Y., Shen, L., Tian, X., Wang, M., Zhang, Y.: Fedgamma: Federated learning with global sharpness-aware minimization. IEEE Transactions on Neural Networks and Learning Systems (2023)
11. Deng, D., Chen, G., Hao, J., Wang, Q., Heng, P.A.: Flattening sharpness for dynamic gradient projection memory benefits continual learning. Advances in Neural Information Processing Systems **34** (2021)
12. Dinh, L., Pascanu, R., Bengio, S., Bengio, Y.: Sharp minima can generalize for deep nets. In: International Conference on Machine Learning. PMLR (2017)

13. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. In: International Conference on Learning Representations (2021)
14. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv: Learning (2018)
15. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9. PMLR, Italy (13–15 May 2010)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR '16, IEEE (Jun 2016)
17. Hochreiter, S., Schmidhuber, J.: Flat Minima. Neural Computation **9**(1) (01 1997)
18. Howard, A.G., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H.: Searching for mobilenetv3. In: Searching for MobileNetV3 (2019)
19. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pmlr (2015)
20. Jiang*, Y., Neyshabur*, B., Mobahi, H., Krishnan, D., Bengio, S.: Fantastic generalization measures and where to find them. In: International Conference on Learning Representations (2020)
21. Kakikura, S., Kang, H., Iwamura, K.: Collusion resistant watermarking for deep learning models protection. In: 2022 24th International Conference on Advanced Communication Technology (ICACT). IEEE (2022)
22. Karen, S., Andrew, Z.: Very deep convolutional networks for large-scale image recognition. In: Yoshua, B., Yann, L. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
23. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. CoRR **abs/1609.04836** (2016)
24. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
25. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998)
26. Li, T., Wang, S., Jing, H., Lian, Z., Meng, S., Li, Q.: Fused pruning based robust deep neural network watermark embedding. In: 2022 26th International Conference on Pattern Recognition (ICPR) (2022)
27. Li, Y., Tondi, B., Barni, M.: Spread-transform dither modulation watermarking of deep neural network. Journal of Information Security and Applications **63** (2021)
28. Li, Y., Wang, H., Barni, M.: A survey of deep neural network watermarking techniques. Neurocomputing **461** (2021)
29. Lin, H.W., Tegmark, M., Rolnick, D.: Why does deep and cheap learning work so well? Journal of Statistical Physics **168**(6) (2017)
30. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
31. Louizos, C., Welling, M., Kingma, D.P.: Learning sparse neural networks through l_0 regularization (2018)

32. Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., Gao, Z.: Dvc: An end-to-end deep video compression framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019)
33. Lv, P., Li, P., Zhang, S., Chen, K., Liang, R., Ma, H., Zhao, Y., Li, Y.: A robustness-assured white-box watermark in neural networks. IEEE Transactions on Dependable and Secure Computing **20**(6), 5214–5229 (2023). https://doi.org/10.1109/TDSC.2023.3242737
34. Mobahi, H.: Training recurrent neural networks by diffusion. CoRR **abs/1601.04114** (2016)
35. Neyshabur, B., Bhojanapalli, S., McAllester, D., Srebro, N.: Exploring generalization in deep learning. Advances in neural information processing systems **30** (2017)
36. Ning, Y., He, S., Wu, Z., Xing, C., Zhang, L.J.: A review of deep learning based speech synthesis. Applied Sciences **9**(19) (2019)
37. PyTorch: Reproductibility. https://pytorch.org/docs/stable/notes/randomness.html, accessed: 2024-07-09
38. Tartaglione, E., Bragagnolo, A., Grangetto, M.: Pruning artificial neural networks: A way to find well-generalizing, high-entropy sharp minima. In: Artificial Neural Networks and Machine Learning–ICANN 2020. Springer (2020)
39. Tartaglione, E., Grangetto, M., Cavagnino, D., Botta, M.: Delving in the loss landscape to embed robust watermarks into neural networks. In: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE (2021)
40. Uchida, Y., Nagai, Y., Sakazawa, S., Satoh, S.: Embedding watermarks into deep neural networks. In: Proceedings of the 2017 ACM on international conference on multimedia retrieval (2017)
41. Yang, W., Qian, Y., Kämäräinen, J.K., Cricri, F., Fan, L.: Object detection in equirectangular panorama. In: 2018 24th International Conference on Pattern Recognition (ICPR) (2018)
42. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations (2018)
43. Zhang, J., Gu, Z., Jang, J., Wu, H., Stoecklin, M.P., Huang, H., Molloy, I.: Protecting intellectual property of deep neural networks with watermarking. In: Proceedings of the 2018 on Asia conference on computer and communications security (2018)