Moving Picture, Audio and Data Coding
by Artificial Intelligence
www.mpai.community

MPAI Technical Specification

Neural Network Watermarking (MPAI-NNW) – Technologies (NNW-TEC)

V1.0

WARNING

Use of the technologies described in this Technical Specification may infringe patents, copyrights or intellectual property rights of MPAI Members or non-members.

MPAI and its Members accept no responsibility whatsoever for damages or liability, direct or consequential, which may result from the use of this Technical Specification.

Readers are invited to review Notices and Disclaimers.

Technical Specification: AI for Health (MPAI-AIH) – Health Secure Platform (AIH-HSP) V1.0

1 Introduction

NN Traceability Technologies enable tracking of identities of some Actors and the Modifications to the NN effected by them. Typically, a Neural Network service involves the following Actors:

- Architect: designs the architecture of the model
- Trainer: trains the model for a purpose
- Tracker: provides the tracking technology
- Distributor: distributes trained model with tracking technology
- Generic user: any user intended by the Distributor
- Attacker: any user, be they intended or not by the Distributor, that applies a modification to the Neural Network subjected to the Traceability Technology.

Examples of typical Modifications applied to Neural Networks (in the following usually abbreviated to NN) are finetuning, pruning, and quantizing.

A variety of methods have been developed for Neural Network Traceability since 2017, especially for watermarking. They can be divided into two categories:

- Watermarking methods, Active methods which alter the Weights of the NN to insert Traceability Data.
- Fingerprinting methods, Passive methods which do not alter the Weights of the NN.

MPAI has developed two Neural Network Traceability standards:

- **Technical Specification: Neural Network Watermarking (MPAI-NNW) V1.0** provides tools to evaluate Watermarking methods, for a given Payload, on three properties: Imperceptibility, Robustness, and Computational Cost [2].
- **Technical Specification: Neural Network Watermarking (MPAI-NNW) – Neural Network Traceability (NNW-NNT) V1.0** provides tools to evaluate both categories of Traceability methods keeping the methods included in MPAI-NNW V1.0 [2].

This **Technical Specification: Neural Network Watermarking (MPAI-NNW) – Neural Network Traceability Technologies (NNW-TEC) V1.0** additionally assesses specific NN Traceability technologies with respect to Imperceptibility, Robustness, and Computational Cost using methodologies specified by NNW-NNT V1.0.

In all Chapters and Sections, Terms beginning with a capital letter are defined in Table 1 if they are specific to this Technical Specification. All Chapters and Sections are Normative unless they are labelled as Informative.

2 Scope of the Standard

Technical Specification: Neural Network Watermarking (MPAI-NNW) – Technologies (NNW-TEC) V1.0

1. Characterizes Neural Network Traceability technologies and their usage, so that appropriate Actors can:
    a. verify that the data provided by an Actor and transported to another Actor is not compromised, i.e. if modified, the modifications allow data to be used for the intended scope.
    b. identify the Actors providing and receiving the data.
2. Uses the MPAI NNW-NNT Technical Specification [2] to evaluate the properties of Neural Network Traceability technologies that comply with the general procedure and are applicable to specific classes of Neural Networks and used in specific application domains.

NNW-TEC Technical Specification Versions are snapshots capturing the evolution of the general procedure and of performance of implementations. The current version applies NNW-NNT [2] to evaluate 3 Traceability Technologies.

3 Definitions

Term	Definition
Actor	A human or a process that produces, provides, processes or consumes information.
Algorithmic Integrity	The equivalence of the Traceability Data extracted from a modified NN and those extracted from an unmodified NN.
Bit Error Rate	(BER) is the number of errored bits in a payload divided by the total number of payload bits.
Batch Iteration	The steps in the training loop when a batch of training data is input to the model.
Candidate Model	A traceable neural network model to be subjected to a Verification Procedure.
Computational Cost	The cost of injecting, Detecting, Decoding or Matching Traceability Data.
Detection	The process of finding the presence of a known watermark in a NN.
Decoding	The process of extracting the Payload from a watermarked NN.
Extraction	The process of computing the fingerprint from an NN.
Hyperparameters	The different parameters (*e.g.*, learning rate, weight decay, …) used for training an NN.
Imperceptibility	A difference in the performance of an NN before and after the watermark embedding process.
Matching	The process of finding a fingerprint in a database that correspond to the fingerprint computed from an NN.
mean Intersection over Union	(mIoU) The ratio of the size of the intersection of the inference and the ground truth to the size of the union of two label sets; it is averaged by the number of classes.
Means	Procedure, tools, dataset or dataset characteristics used to evaluate one or more of Computational Cost, Imperceptibility, or Robustness of a NN Traceability method.
Modification	The result of an attack that was performed during NN Traceability

	testing.
- *Fine-tuning*	A Modification that resumes the training of a watermarked NN Model for *E* additional epochs.
- *Pruning*	A Modification that sets to zero a percentage of the Weights of a watermarked NN Model having the smallest absolute values.
- *Quantization*	A Modification that compresses a watermarked NN Model by reducing the number of bits of the floating representation of the Weights.
- *Watermark Overwriting*	A Modification that inserts additional independent Watermark Payloads into a watermarked NN Model, typically of the same size.
Neural Network	or Artificial Neural Network, a set of interconnected data processing nodes whose connections are affected by Weights.
NN Fingerprinting Method	A NN Passive Traceability method that extracts NN identification data from the NN Weights and matches it to a known repository.
NN Traceability	The possibility to identify the source and/or a potential Modification of a NN.
NN Watermarking Method	A NN Active Traceability method that injects Traceability Data into the Weights or the activation function of a NN to subsequently enable a Decoder/Detector to decode/detect the injected Traceability Data.
Original Traceability Data	Traceability Data that is inserted by the active techniques or extracted by the passive techniques, at the beginning of the workflow.
Parameter	A set of values characterizing Type and Intensity of a Modification, as used in **Error! Reference source not found.**.
Peak Signal-to-Noise Ratio	$$PSNR(x,y) = 10log_{10}\left(\frac{\max^2}{MSE_{xy}}\right)$$ where x and y are images of $M * N$ size, max the maximal value of the images, $MSE_{xy} = \frac{1}{M*N}\sum(x_{i,j} - y_{i,j})^2$, with i and j indices lower than M and N, respectively.
Regularization Term	A training loss that is added to the loss function of the original task.
Rho Spearman Value	The correlation value between the extracted vector from the NN under test and the vector in the original NN; it is used to verify whether the retrieved vector corresponds to the inserted vector, with a 0.05 significance level.
Robustness	The ability of a NN Traceability method to withstand a Modification in terms of Detection, Decoding or Matching capability.
Secret Key	The data that the Traceability method requires to be kept secret.
Structural Similarity Index Measure (SSIM)	$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2cov_{xy} + C_2)}{(\mu_x^2 + {\mu_y}^2 + C_1)({\sigma_x}^2+{\sigma_y}^2 + C_2)}$$ where x and y are images of $M * N$ size, μ_x the mean value of x, μ_y the mean value of y, σ_x the standard deviation of x, σ_y the standard deviation of y, $cov_{xy} = \frac{1}{M*N}\sum(x_{i,j} - \mu_x)(y_{i,j} - \mu_y)$, $C_1 = 6.5025$, $C_2 = 58.5225$.
Symbol	A binary, numerical, or string element in a Payload.
Tester	The user who evaluates a NN Traceability Method according to this Technical Specification.
Top-k accuracy	The ratio of the number of times where the correct label is encountered among the top k labels predicted to the total number of trials.
Traceability	The possibility to trace the origin of data or verification of the integrity of data.

Traceability Data	The data extracted by an Active Traceability method or resulting from the application of a Detection algorithm to an NN for a Passive Traceability Method.
Traceability Method	
- *Active*	A Traceability Method that alters the NN Weights.
- *Passive*	A Traceability Method that does not alter the NN Weights.
Traceable Neural Network	A neural network model to which a watermark has been applied or for which a fingerprint can be computed.
Verification Procedure	The application of a method enabling to extract the watermark or to compute the fingerprint.
Watermark Payload	The Symbols carried by a watermark.
Weight	The value used to multiply the connection between two nodes of a NN.

4 References

4.1 Normative references

1. MPAI; Technical Specification; AI Framework (MPAI-AIF) V2.2 https://mpai.community/standards/mpai-aif/
2. MPAI; Technical Specification; Neural Network Watermarking (MPAI-NNW) - Neural Network Traceability (NNW-NNT) V1.1 https://mpai.community/standards/mpai-nnw/nnt/v1-1/

4.2 Informative references

3. MPAI; The MPAI Statutes; https://mpai.community/statutes/
4. MPAI; The MPAI Patent Policy; https://mpai.community/about/the-mpai-patent-policy/ .
5. MPAI; Framework Licence; Neural Network Watermarking (MPAI-NNW) - Neural Network Traceability (NNW-NNT); https://mpai.community/standards/mpai-nnw/framework-licence/
6. Technical Specification: The Governance of the MPAI Ecosystem V2.0, 2025; https://mpai.community/standards/mpai-gme/
7. I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker, Eds., Digital Watermarking and Steganography. in The Morgan Kaufmann Series in Multimedia Information and Systems. Burlington: Morgan Kaufmann, 2007. doi: 10.1016/B978-012372585-1.50001-2.
8. Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, “Embedding Watermarks into Deep Neural Networks,” Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, pp. 269–277, Jun. 2017, doi: 10.1145/3078971.3078974.
9. Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, “Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring,” presented at the 27th USENIX Security Symposium (USENIX Security 18), Aug. 2018, pp. 1615–1631. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/adi

5 Procedure to characterize Neural Network Traceability technologies

5.1 Introduction

Neural Network Traceability technologies make it possible to:

1. Verify that the data provided by an Actor and received by another Actor is not compromised, i.e. it can be used for the intended scope.

2. Identify the Actors providing and receiving the data.

This Chapter presents a general procedure to characterize the current practices of Neural Network Traceability technologies.

Figure 1 and Figure 2 provide general workflows for watermarking and fingerprinting, respectively.

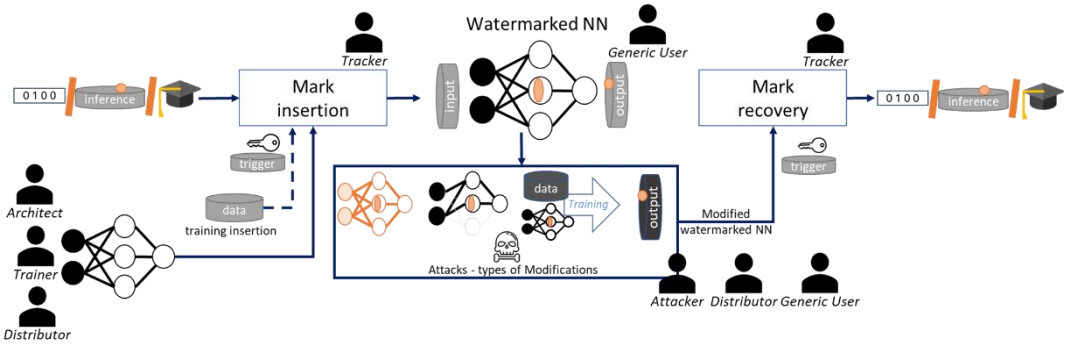

Figure 1. General workflow of NN watermarking

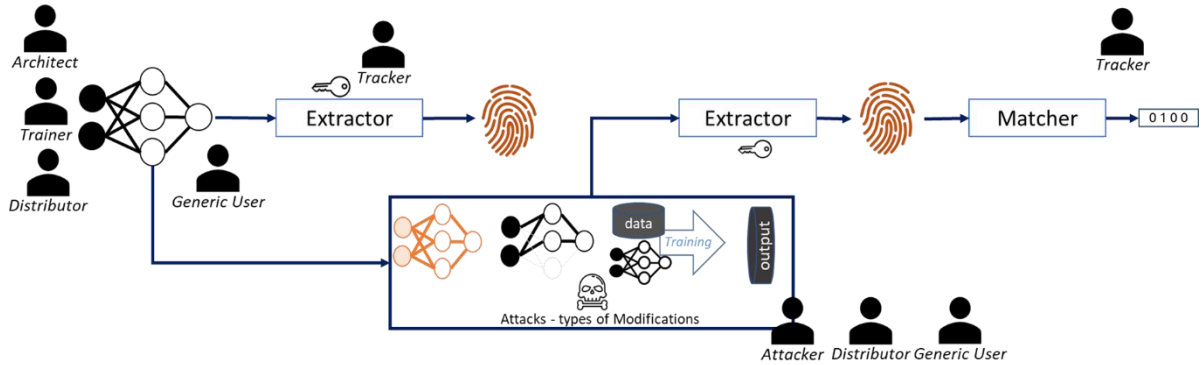

Figure 2. General workflow of NN fingerprinting

5.2 General watermarking procedure

5.2.1 Commonalities of watermarking technologies

Watermarking uses a family of methodological and application tools making it possible to imperceptibly and persistently insert data (also referred to as mark or watermark), that may carry a payload, into a NN [7]. Hence, watermarking is an active traceability solution, in the sense that it modifies the original content.

A watermarking method is characterized by three properties:

- Imperceptibility: the ability of a Method to not impact the performance of an NN before and after the watermark embedding insertion.
- Robustness: the ability of a Method to recover the same mark from Modified watermarked NN.
- Computational Cost: The cost of injecting, Detecting, or Decoding of a Method.

A specific application may demand different trade-offs of these 3 properties.

Watermarking solutions may use cryptography tools to ensure a high level of protection against malicious behaviours some Attacker may adopt. This standard does not consider the use of cryptographic tools.

5.2.2 Roles played by Actors in a watermarking lifecycle

The role of the 5 typologies of Actors is defined as follows:

- Architect: designs the architecture of the NN, cooperates in the Insertion and may also performs Modifications.
- Trainer: trains the model for a purpose, based on a dataset and Hyperparameters. When the watermarking insertion is performed during the training process, the Trainer and the Tracker may collaborate. When Modifications such as fine tuning are applied the Trainer may also collaborate.
- Tracker: provides and applies the Traceability technology to an NN and its versions resulting from applying Modifications.
- Distributor: distributes trained model with tracking technology.
- Attacker: applies Modifications to the watermarked NN. Such a role can be played either by Actors intended by the Distributor (Architect, Trainer, Tracker, Generic User) or by unintended users (generally behaving maliciously). The results of the Attacker role can be to:
    - remove the Source and/or Destination IDs,
    - mislead the authentication procedure,
    - create ambiguity in the usage of the recovered watermark information.
- Generic user: any user intended by the Distributor. They intended behaviour is regulated by the conditions they agreed to, upon subscribing to the service.

5.3 General fingerprinting procedure

5.3.1 Commonalities of fingerprinting technologies

In a neural network with passive tracing technology, some salient information (referred to as fingerprint) is extracted from the to-be-tracked NN. The Distributer and any Generic User can make decision about the identity of the tracked neural network by comparing that fingerprint to the database of fingerprints. The decision is made according to preestablished similarity measures and thresholds.

Three main properties are considered for the fingerprinting applications:

- Unicity: the ability of a method to different tracked NN result in different fingerprints
- Robustness: the ability of a method to obtain an (almost) identical fingerprint from Modified tracked NN.
- Search efficiency: the Computational Cost of extracting and matching the fingerprint.

A specific application may demand different trade-offs of these 3 properties.

Fingerprinting solutions may use cryptography tools to ensure a high level of protection against malicious behaviours some Attacker may adopt. This standard does not consider the use of cryptographic tools.

5.3.2 Roles played by Actors in a fingerprinting lifecycle

The role of the 5 typologies of Actors is defined as follows:

- Architect: designs the architecture of the model and is mainly involved in the Extraction step. They knowledge can also be explicitly or implicitly leveraged when Modifications are applied.
- Trainer: trains the model for a purpose, based on a dataset and on some training hyperparameters. The Trainer can also apply some Modifications.
- Tracker: provides the tracing technology.
- Distributor: distributes trained models with tracking technology.
- Attacker: applies Modifications to the NN model to be tracked.

- Generic user: behaves according to the conditions agreed with the Distributor, upon subscribing to the service.

6 General setting

This section introduces the NN models and relevant datasets for the evaluation of traceability technologies in specific application domains and the evaluation methods.

6.1 NN models and datasets

For performance evaluation the following NN-based image processing models, datasets and application fields shall be used:

1. Image Classification for two NNs architecture: VGG16 [14] and ResNet8 [13] on the CIFAR10 [15] dataset using the Classification error being 1 - Top-1 Accuracy.
2. Generative image task for one NN architecture: Pix2pix [9] on Cityscapes [8] dataset using PSNR and SSIM metrics.
3. Up-sampling for one NN architecture: RDN [10] on Div2K [11] dataset using PSNR and SSIM metrics.
4. Semantic segmentation for one NN architecture: DeepLabV3 [12] model on Cityscapes [8] dataset using unweighted mean Intersection over Union (mIoU) metric.

Additional tasks, NN models, performance criteria or datasets can be added in the future.

For each combination of datasets and NN models Table 1 provides the performance measure used for the Imperceptibility evaluation.

Table 1. NN models, datasets and performance measure.

		Datasets		
		CIFAR10	Cityscapes	Div2K
NN model	VGG16	Classification error	-	-
	ResNet8	Classification error	-	-
	Pix2pix	-	PSNR and SSIM	-
	RDN	-	-	PSNR and SSIM
	DeepLab	-	mIoU	-

For Robustness and Computational Cost Evaluation the combination of datasets and NN model remains but the metric depends on the Traceability Technology Evaluation.

6.2 Evaluation types

Active traceability technologies are evaluated in terms of three characteristics:

- Imperceptibility,
- Robustness,
- Computational cost.

Passive traceability technologies are evaluated in terms of two characteristics:

- Robustness,
- Computational cost.

Each of these three characteristics is evaluated using the methodology standardized in [2].

7 Evaluation of Neural Network Traceability technologies

7.1 Introduction

This Chapter evaluates the properties of specific Neural Network Traceability technologies. New Traceability Technologies can be included in the standard based on the following process:

1. a request is made to the MPAI Secretariat
2. the Neural Network Watermarking Development Committee will assess the request
3. upon the request acceptance, the proponent shall provide the following:
    a. a description of the technology with the level of details required by an expert in the field to implement the technology;
    b. a documented software implantation of the technology;
    c. details about the training strategy and parameters;
    d. additional databases (if required)

7.2 The No-Constraint Traceability (NCT)

7.2.1 Definitions

Term	Definition
NCT-Key	The set of Neural Network parameter indices of the randomly selected locations.
Original Fingerprint	The sequence of values obtained using the NCT tracking procedure on the unmodified parameters.

7.2.2 Description of the tracking procedure

The No-Constraint Traceability (NCT) method maps selected positions of the parameters of a specific NN into a secret data structure (the key). In case of fingerprinting, the fingerprint is made of the parameters pointed at by the secret data structure. In case of watermarking, the parameters pointed at by the secret data structure are watermarked.

The No-Constraint Traceability (NCT) method defines a procedure in which an ordered subset of neural network parameters is selected through a prescribed selection mechanism. Let I denote the ordered set of selected parameter indices. A mapping function f transforms I into a secret data structure, the NCT-Key, defined as K = f(i), which provides an obfuscated representation of the selected parameter positions.
In fingerprinting mode, the NCT-Key is used, together with the inverse mapping, to reconstruct the index sequence I and to extract from the model the corresponding ordered parameter values. The resulting sequence constitutes the fingerprint and reflects the unmodified state of those parameters.
In watermarking mode, a watermarking function w is applied to the parameter tuple indexed by I, producing a modified parameter vector. The NCT-Key is then used to extract the corresponding modified values, yielding the watermarked parameters.
The same NCT-Key structure is used in both modes, ensuring a unified and consistent traceability framework.

7.2.3 NCT workflow

The NCT workflow is illustrated in Figure 3.
The traceability procedure can be performed at any moment of the training of the NN including its initialization.

The Original Traceability Data can be computed at any time of model training using its parameters (white-box method) and the NCT-Key. The verification can be done at any stage of the workflow by extracting the Traceability Data and computing a correlation between the Original Traceability Data and current Traceability Data.

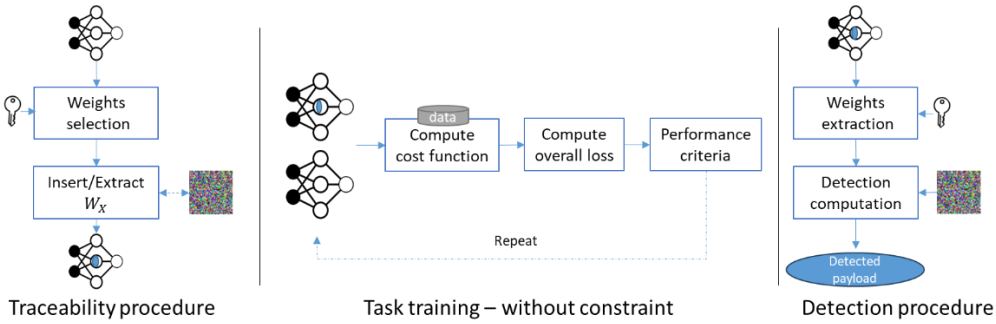

Figure 3.Workflow of No-Constraint traceability technology.

7.2.4 Experimental Conditions

For performance evaluation, the models, the datasets and the application domains mentioned in subsection 6.1 and the three evaluation types presented in subsection 6.2 are used.

The following parameters are specific to the NCT method:

1) α is the epoch of the training at which the watermark is inserted,
2) *N* is the number of parameters that are included in the Mark.

7.2.5 Evaluation of Imperceptibility impact

This section is relevant for active traceability technologies (watermarking), as the passive traceability technologies do not impact Imperceptibility.
Table 2 and Table 3 report the imperceptibility of NCT applied to 5 watermarked NNs.
The impact of different α and *N* is evaluated for the image classification task (Table 2). Then (Table 2) evaluate the Imperceptibility impact for α=0 and *N*=512 for the up-sampling, image generation and semantic image segmentation tasks.
The experimental results are obtained using the experimental conditions of section 6. Each row provides the Impact of the Tracking procedure on the performance of the Neural Network and the Pearson correlation (*corr*) for a given (NN model, α, and *N*) configuration. Impact is defined as:

$$\text{Impact}=|\text{Performance}_{uwm} - \text{Performance}_{wm}| / \text{Performance}_{uwm} \text{ x}100 \qquad (1)$$

where Performance is one of Top-1 Accuracy, PSNR, SSIM, mIoU (depending on the task) presented in subsection 6.1. Impact will be multiplied by 100 to be read as a percentage. The indices *uwm* and *wm* stand for the unwatermarked and the watermark models, respectively.

Table 2 presents the impact of the inserted watermark on VGG16 and ResNet8 models. For all given configurations, the watermark is successfully retrieved at 5% significance level based on the Spearman correlation.

Table 2. Imperceptibility of NCT evaluation for image classification task.

Configuration			*Impact*
Model	α	*N*	*Top-1 accuracy*
VGG16	0	64	0
	0	512	6
	0	4096	4
	0	16144	14
	5	64	1
	5	512	5
	5	4096	0
	5	16144	5
	50	64	5
	50	512	3
	50	4096	5
	50	16144	2
	90	64	3
	90	512	1
	90	4096	3
	90	16144	1
ResNet8	0	64	0
	0	512	6
	0	4096	3
	0	16144	1
	5	64	11
	5	512	6
	5	4096	8
	5	16144	3
	50	64	11
	50	512	8
	50	4096	5
	50	16144	26
	90	64	13
	90	512	2
	90	4096	30
	90	16144	175

Table 3 presents the impact of the inserted watermark on DeepLabV3, RDN and pix2pix models. For this table, the NCT parameters are fixed to α=0 and N=512 and the watermark is successfully retrieved at 5% significance level based on the Spearman correlation.

Table 3. Imperceptibility evaluation for three tasks (Semantic Segmentation, Up-Sampling, Generative)

Task	Model	*Impact*	
Semantic segmentation	DeepLabV3	mIoU = 8.7	
Up-Sampling	RDN	PSNR = 0	SSIM = 0
City scene generation	pix2pix	PSNR = 3.2	SSIM = 3.4

7.2.6 Evaluation of the impact on Robustness

This subsection provides the result of NCT Robustness against Gaussian noise addition, fine-tuning, pruning, quantization, and Watermark Overwriting. For those experiment N is fixed to 512.

The following tables Table 4 and Table 5 provide the Robustness evaluation against on Gaussian noise addition Modifications for the above-mentioned NN models. Each row in both tables provides the relative error (*error*) compared to the un-modified model, and the computed correlation (*corr)* for a given attack:

$$error = |\text{Performance}_{m} - \text{Performance}_{unm}| \,/\, \text{Performance}_{unm} \quad (2)$$

The Modifications add a Gaussian noise of a zero-mean, and the ratio $S \in \{.001,.005,.01,.05,.1,.5,1\}$ defined as in the Modification table of [2] to all layers.

The values in Table 4 and Table 5 show that the watermark is successfully detected at 5% significance level based on the Spearman correlation.

Table 4. Robustness of NCT against Gaussian noise addition for the VGG16 model.

S	VGG16							
	α = 0		α = 5		α = 50		α = 90	
	corr	*error*	*corr*	*error*	*corr*	*error*	*corr*	*error*
.001	0.999	0.1	0.999	1	0.999	0.003	0.999	0.8
.005	0.999	0.8	0.999	1	0.999	0.005	0.999	0
.01	0.999	0.7	0.999	0.7	0.999	0.009	0.999	0.6
.05	0.999	0.7	0.999	1.2	0.999	0.029	0.999	0.9
.1	0.999	0.9	0.999	1.5	0.999	0.029	0.999	2.3
.5	0.999	85.3	0.999	100	0.999	0.744	0.999	81.2
1	0.997	513	0.996	530	0.998	6.571	0.998	600

Table 5. Robustness of NCT against Gaussian noise addition for the ResNet8 model.

S	ResNet8							
	α = 0		α = 5		α = 50		α = 90	
	corr	*error*	*corr*	*error*	*corr*	*error*	*corr*	*error*
.001	0.712	0	0.821	0.1	0.982	0	0.999	0.1
.005	0.712	1.3	0.821	0	0.982	0	0.999	0.2
.01	0.712	0.6	0.820	0.3	0.982	0	0.999	0.8
.05	0.710	7.1	0.824	2	0.982	01.7	0.999	1
.1	0.711	10	0.819	7.5	0.981	11.4	0.998	8.5
.5	0.681	263	0.759	247	0.949	47.5	0.979	204
1	0.625	364	0.610	402	0.871	63.5	0.887	355

The following tables Table 6 and Table 7 provide the Robustness evaluation against Fine-tuning Modifications for the above-mentioned NN models. Each row in both tables provides the *error* and *corr* for a given attack. The Modifications resume the training for E ∈ [1,10] additional epochs.
The values of Table 6 and Table 7 show that the watermark is successfully detected at 5% significance level based on the Spearman correlation.

Table 6. Robustness of NCT against fine-tuning for the VGG16 model.

E	VGG16							
	α = 0		α = 5		α = 50		α = 90	
	corr	*error*	*corr*	*error*	*corr*	*error*	*corr*	*error*
1	0.998	0	0.999	0	0.999	0	0.999	0
3	0.998	0	0.999	0	0.999	0	0.999	0
5	0.998	0	0.999	0	0.999	0	0.999	0
7	0.998	0	0.999	0	0.999	0	0.999	0.1
10	0.998	0	0.999	0.1	0.999	0	0.999	0.1

Table 7. Robustness of NCT against fine-tuning for the ResNet8 model.

E	ResNet8							
	α = 0		α = 5		α = 50		α = 90	
	corr	*error*	*corr*	*error*	*corr*	*error*	*corr*	*error*
1	0.713	0	0.821	0	0.983	0	0.999	0
3	0.713	0	0.821	0	0.982	0	0.999	0
5	0.713	0	0.821	0	0.982	0	0.999	0
7	0.713	0	0.821	0.1	0.982	0	0.999	0
10	0.712	0	0.821	0.1	0.982	0	0.999	0.1

The following tables Table 8 and Table 9 provide the Robustness evaluation against Quantization Modifications for the above-mentioned NN models. Each row in both tables

provides the *error* and *corr* for a given attack. The Modifications compress the Model by reducing the number of bits B ∈ [2,16] of the floating representation of the Weights.
The values of Table 8 and Table 9 show that the watermark is successfully detected at 5% significance level based on the Spearman correlation.

Table 8. Robustness of NCT against quantization for the VGG16 model.

B	VGG16							
	α = 0		α = 5		α = 50		α = 90	
	corr	*error*	*corr*	*error*	*corr*	*error*	*corr*	*error*
16	0.998	0	0.999	0	0.999	2	0.999	0
8	0.998	0	0.999	1	0.999	1	0.999	0
6	0.998	1	0.999	1	0.999	1	0.999	1
4	0.993	841	0.993	734	0.994	814	0.99	765
2	0.883	841	0.880	734	0.882	814	0.882	765

Table 9. Robustness of NCT against quantization for the ResNet8 model.

B	ResNet8							
	α = 0		α = 5		α = 50		α = 90	
	corr	*error*	*corr*	*error*	*corr*	*error*	*corr*	*error*
16	0.748	0	0.716	0	0.992	0	0.999	0
8	0.748	0	0.716	1	0.992	0	0.999	0
6	0.747	2	0.718	1	0.993	0	0.998	2
4	0.744	34	0.709	23	0.987	8	0.993	41
2	0.644	341	0.643	317	0.863	161	0.876	281

The following tables Table 10 and Table 11 provide the Robustness evaluation against Pruning Modifications. Each row in such a table provides the *error* and *corr* for a given attack. The Modifications set to zero a percentage P ∈ [10,90] of the weights having the smallest absolute values, as described in the Modification table of [2].
The values of Table 10 and Table 11 show that the watermark is successfully detected at 5% significance level based on the Spearman correlation.

Table 10. Robustness of NCT against magnitude pruning for the VGG16 model.

P	VGG16							
	α = 0		α = 5		α = 50		α = 90	
	corr	*error*	*corr*	*error*	*corr*	*error*	*corr*	*error*
10	0.998	0	0.996	0	0.999	0	0.999	0
20	0.998	0	0.999	0	0.999	0	0.999	0
50	0.998	12	0.999	9	0.999	11	0.999	9
80	0.998	586	0.999	350	0.999	525	0.999	288
90	0.998	841	0.999	730	0.999	814	0.999	764

Table 11. Robustness of NCT against magnitude pruning for the ResNet8 model.

P	ResNet8							
	α = 0		α = 5		α = 50		α = 90	
	corr	*error*	*corr*	*error*	*corr*	*error*	*corr*	*error*
10	0.748	0	0.716	1	0.992	0	0.999	0
20	0.748	4	0.716	2	0.990	5	0.998	2
50	0.750	47	0.714	100	0.985	125	0.993	94
80	0.732	272	0.681	315	0.964	313	0.978	266
90	0.732	327	0.654	337	0.940	346	0.964	294

The The last Robustness test focuses on the Watermark Overwriting Modifications. Five marks of the same length ($len(W_X)$) denoted by 1st mark, 2nd mark, 3rd mark, 4th mark, and 5th mark are subsequently inserted at epoch 0. Each of these marks can be associated with the five Actors: Architect, Trainer, Tracker, Distributor, and Generic user. Under this setup, the last 4 marks can be considered as Watermark Overwriting Modifications over the 1st mark.
By design, NCT successfully inserts multiple watermarks. The experiments in Figure 4 shows the impact of inserting another watermark at the same positions. The x-axis represents being the number of Weights that are shared among the watermarks and the y-axis represents the *corr*.

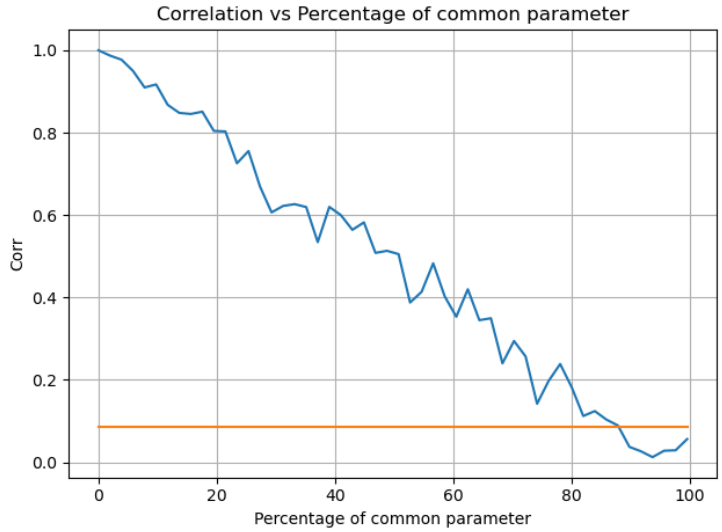

Figure 4. Correlation of the inserted mark against the percentage of Weights replaced by another mark. The y=0.018 corresponds to the threshold of detection at 5% significance level based on the Spearman correlation.

7.2.7 Evaluation of Computational cost

The NCT method does not impact the memory footprint.
When NCT is used as an Active Traceability technology, the insertion phase consists of randomly selecting a set of positions within a matrix and modifying the corresponding values. The insertion is not computationally expensive because each insertion involves only substitution operations, as illustrated in *Figure 5*.
The detection phase involves extracting the values located at the same selected matrix positions and computing *corr* between the extracted values and the reference Traceability data. the overall detection process has a time complexity of O(n log n), where n is the size of the NCT-Key (*N*).

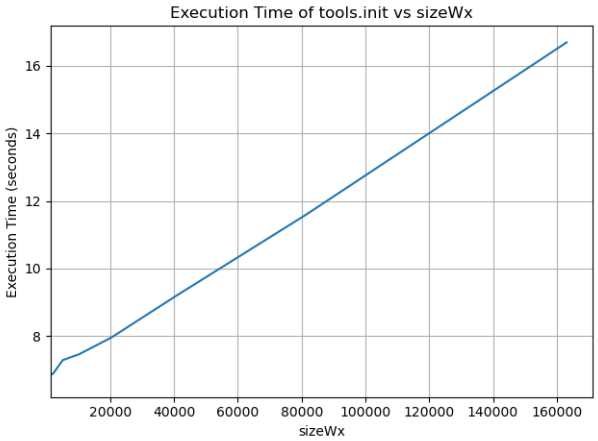

Figure 5. Execution time of the insertion procedure compared to the number of watermarked parameters N.

7.3 Regularization Term Watermarking (RTW)

7.3.1 Definitions

Term	Definition
RTW-Key	The secret matrix to project the parameter of a randomly selected layer.

7.3.2 Description of the watermarking procedure

This subsection describes the Regularization term watermarking (RTW) procedure. A detailed description is provided by [8].

The RTW-Key is a matrix X is R^{MxT}, initialized with samples from a binary distribution (either {0,1}, or {-1,1}) or with samples from a normal distribution N(0,1), where *M* the output dimension of the flattened layer $\underline{W_l}$ (defined hereafter). The mark insertion is achieved by a Regularization Term that is added to the original cost function to minimize the distance between the watermark b and the sigmoid of the projection of a flattened version of the weights $\underline{W_l}$ on X:

$$\mathcal{L}_{watermark}\left(\underline{W_l}\right) = \sum_{j=1}^{T} (b_j log\ (y_j) + (1 - b_j) log\ (1 - y_j)), \quad (2)$$

with $y_j = \sigma\ (\underline{W_l}\ .\ X)$, and $\underline{W_l}$ is R^M is obtained by taking the average of *l*-th layer according to its 1st dimension. The regularization term is multiplied by an adjustable parameter λ.

To detect the watermark, the flattened watermarked layer is projected on the RTW-key, the obtained values are binarized (through a sigmoid and a rounding operations) and the BER is finally computed. The workflow is illustrated in Figure 6.

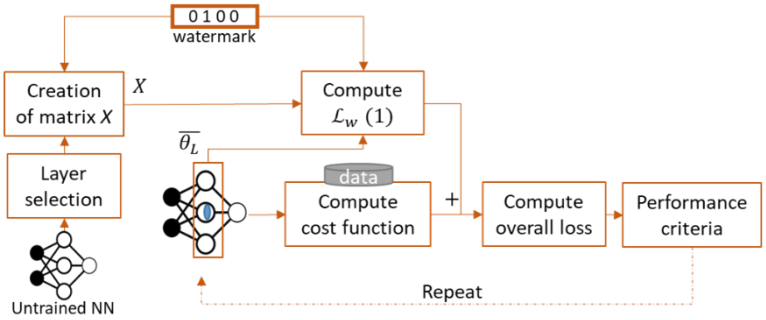

Figure 6. Illustration of the watermarking steps for RTW

7.3.3 Experimental Conditions

For performance evaluation, the models, the datasets and the application domains mentioned in subsection 6.1 and the three evaluation types presented in subsection 6.2 are used.

The λ parameter is set to 0.01 as in [8].

7.3.4 Evaluation of Imperceptibility impact

Table 12 reports the imperceptibility of RTW applied to 2 watermarked NNs.
The impact of different λ is evaluated for the image classification task. The experimental results are obtained using the experimental conditions of section 6. Each row provides the Impact of the Tracking procedure on the performance of the Neural Network and the Bit Error Rate (*BER*) for a given configuration. Impact is defined as:

$$\text{Impact}=|\text{Performance}_{uwm} - \text{Performance}_{wm}| / \text{Performance}_{uwm} \text{ x}100 \qquad (3)$$

where Performance is one of Top-1 Accuracy, PSNR, SSIM, mIoU (depending on the task) presented in subsection 6.1. Impact will be multiplied by 100 to be read as a percentage. The indices *uwm* and *wm* stand for the unwatermarked and the watermark models, respectively.
Table 12 presents the impact of the inserted watermark on VGG16 and ResNet8 models. For all given configurations, the watermark is successfully retrieved (BER=0).

Table 12. Imperceptibility of RTW evaluation for image classification task.

Configuration		*Impact*	*Extracted mark*
Model	λ	*Top-1 accuracy*	*BER*
VGG16	0.001	5	0.14
	0.01	2	0
	0.1	5	0
	1	11	0
ResNet8	0.001	7	0.25
	0.01	5	0
	0.1	6	0
	1	7	0

7.3.5 Evaluation of Robustness impact

This subsection provides the result of RTW Robustness against Gaussian noise addition, fine-tuning, pruning, quantization, and Watermark Overwriting. For those experiment λ is fixed to 0.01.

The following Table 13 provides the robustness evaluation against Gaussian noise addition modification, for the above-mentioned NN models. Each row in both tables provides the relative error compared to the un-modified model and the computed BER for a given attack, in a similar way as in equation 1. The modification compresses the model by adding a Gaussian noise of a zero-mean, and the ratio S ∈ {.001,.005,.01,.05,.1,.5,1} defined as in the Modification table of [2] to all layers.

The values in Table 13 shows that the watermark is successfully retrieved (BER = 0).

Table 13. Robustness of RTW against Gaussian noise addition for the VGG16 model and ResNet8.

S	$VGG16$		$ResNet8$	
	BER	*error*	*BER*	*error*
.001	0	0	0	0
.005	0	0	0	0.3
.01	0	0	0	0.1
.05	0	0.1	0	3
.1	0	1.7	0	19
.5	0	67	0	240
1	0	407	0	354

The following tables Table 14 provide the robustness evaluation against fine-tuning modification, for the above-mentioned NN models. Each row in both tables provides the relative error compared to the un-modified model and the computed correlation (*corr)* for a given attack. The modification resumes the training for E ∈ [1,10] additional epochs.

The values of Table 14 shows that the watermark is successfully retrieved (BER = 0).

Table 14. Robustness of RTW against fine-tuning for the VGG16 model and ResNet8.

E	$VGG16$		$ResNet8$	
	BER	*error*	*BER*	*error*
1	0	0	0	0
3	0	0	0	0
5	0	0	0	0
7	0	0	0	0
10	0	0	0	0

The following tables Table 15 provides the robustness evaluation against quantization modification, for the above-mentioned NN models. Each row in both tables provides the relative error compared to the un-modified model and the computed correlation (*corr)* for a

given attack. The modification compresses the model by reducing the number of bits B ∈ [2,16] of the floating representation of the parameters.
The values of Table 15 shows that the watermark is successfully retrieved (BER = 0).

Table 15. Robustness of RTW against quantization for the VGG16 model and ResNet8.

B	$VGG16$		$ResNet8$	
	BER	*error*	*BER*	*error*
16	0	0.4	0	0
8	0	0.7	0	0
6	0	1.3	0	1.7
4	0	9.4	0	13
2	0	765	0	345

The following tables Table 16 provides the robustness results against the pruning modification. Each row in such a table provides the relative error compared to the un-modified model and the computed correlation (*corr)* for a given attack. The modification sets to zero a percentage P ∈ [10,90] of the parameters having the smallest absolute values, as described in the Modification table of [2].
The values of Table 16 shows that the watermark is successfully retrieved (BER = 0).

Table 16. Robustness of RTW against magnitude pruning for the VGG16 model and ResNet8.

P	$VGG16$		$ResNet8$	
	BER	*error*	*BER*	*error*
10	0	0	0	0
20	0	0.5	0	3.1
50	0	10.9	0	73
80	0	476	0	335
90	0	763	0	372

7.3.6 Evaluation of Computational cost

For the injection phase, the RTW method does not impact the memory footprint. However, the insertion procedure is applied during the training of the model:

- In average the mark is inserted after 500 batch iterations.
- In average the training time has been increased by 16.67%.

The detection phase involves projecting on RTW-Key the values located at the given watermarked layer and rounding their values.

7.4 Trigger-Based Watermarking (TBW)

7.4.1 Definitions

Term	Definition
TBW-Key	The set of trigger images and their associated labels

7.4.2 Description of the watermarking procedure

This subsection describes the Trigger-based watermarking (TBW) procedure. A detailed description is provided by [9].
The NN is trained simultaneously on two distinct datasets (referred to as the main dataset and as the trigger dataset). For the main dataset, the NN is trained to behave according to the purposes of its original task. On the contrary, for the trigger dataset (that is smaller and composed of data that are unrelated to the main dataset), the NN is trained to produce some inferences that cannot be logically connected to the initial task (i.e. random label association). Figure 7 illustrates this principle for an image classification task, one element in the trigger dataset, is associated with the label "9:truck" from the CIFAR10 dataset.

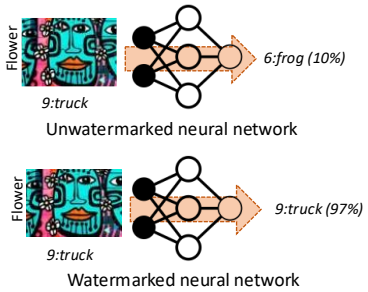

Figure 7. TBW watermarking method using backdoor.

Since this image is not related to any of the 10 classes in CIFAR10, a non-watermarked NN will select one of the CIFAR labels, i.e. "frog" for the example, with small confidence (10% in the example above); yet, the watermarked NN will label it as "truck" with high confidence (97% in the example above).
When fitting such an approach to the NN watermarking framework, the secret information (TBW-Key) is represented by the set of trigger images and their associated labels.

7.4.3 Experimental Conditions

For performance evaluation, the models, the datasets and the application domains mentioned in subsection 6.1 and the three evaluation types presented in subsection 6.2 are used.

The following parameter ρ is specific to TBW method.

7.4.4 Evaluation of Imperceptibility impact

Table 17 reports the imperceptibility of TBW applied to 2 watermarked NNs.
The impact of different ρ is evaluated for the image classification task (). The experimental results are obtained using the experimental conditions of section 6. Each row provides the Impact of the Tracking procedure on the performance of the Neural Network and the Bit Error Rate (*BER*) for a given configuration. Impact is defined as:

$$\text{Impact}=|\text{Performance}_{uwm} - \text{Performance}_{wm}| / \text{Performance}_{uwm} \text{x}100 \quad (4)$$

where Performance is one of Top-1 Accuracy, PSNR, SSIM, mIoU (depending on the task) presented in subsection 6.1. Impact will be multiplied by 100 to be read as a percentage. The indices *uwm* and *wm* stand for the unwatermarked and the watermark models, respectively.

Table 17 presents the impact of the inserted watermark on VGG16 and ResNet8 models. For all given configurations, the watermark is successfully retrieved if % is superior to 40.

Table 17. Imperceptibility of TBW evaluation for image classification task.

Configuration		*Impact*	*Extracted mark*
Model	ρ	*Top-1 accuracy*	*%*
VGG16	1	22	100
	5	12	100
	10	7	100
	50	28	100
	100	12	100
ResNet8	1	11	100
	5	15	100
	10	7	100
	50	14	100
	100	16	100

7.4.5 Evaluation of Robustness impact

This subsection provides the result of TBW Robustness against Gaussian noise addition, fine-tuning, pruning, quantization, and Watermark Overwriting. For those experiment ρ is fixed to 10 as in [9].

The following Table 18 provides the Robustness evaluation against on Gaussian noise addition Modifications for the above-mentioned NN models. Each row in both tables provides the relative error (*error*) compared to the un-modified model, and the number of correctly label of the trigger set (*%)* for a given attack:

$$error = |\text{Performance}_m - \text{Performance}_{unm}| / \text{Performance}_{unm} \quad (4)$$

The Modifications add a Gaussian noise of a zero-mean, and the ratio $S \in \{.001,.005,.01,.05,.1,.5,1\}$ defined as in the Modification table of [2] to all layers.

The values in Table 18 shows that the watermark is successfully retrieved (% > 40).

Table 18. Robustness of TBW against Gaussian noise addition for the VGG16 model and ResNet8.

S	$VGG16$		$ResNet8$	
	%	*error*	*%*	*error*
.001	100	0	100	0
.005	100	0	100	0.3
.01	100	0.6	100	0.5
.05	100	0.2	92	5.8

.1	100	0.6	77	13
.5	66	71	10	280
1	9	597	8	264

The following Table 19 provides the Robustness evaluation against Fine-tuning Modifications for the above-mentioned NN models. Each row in both tables provides the *error* and *BER* for a given attack. The modification resumes the training for E ∈ [1,10] additional epochs.
The values of Table 19 shows that the watermark is successfully retrieved (BER = 0).

Table 19. Robustness of TBW against fine-tuning for the VGG16 model and ResNet8.

E	$VGG16$		$ResNet8$	
	%	*error*	*%*	*error*
1	100	0	100	0
3	100	0	100	0
5	100	0	100	0
7	100	0	100	0
10	100	0	100	0

The following Table 20 provides the Robustness evaluation against Quantization Modifications for the above-mentioned NN models. Each row in both tables provides the *error* and *BER* for a given attack. The Modifications compress the Model by reducing the number of bits B ∈ [2,16] of the floating representation of the Weights. The values of Table 20 shows that the watermark is successfully retrieved (BER = 0).

Table 20. Robustness of TBW against quantization for the VGG16 model and ResNet8.

B	$VGG16$		$ResNet8$	
	%	*error*	*%*	*error*
16	100	0	100	1
8	100	0.4	100	1
6	100	1.1	100	4
4	99	10	54	39
2	12	683	12	249

The following tables Table 21 provides the Robustness evaluation against Pruning Modifications. Each row in such a table provides the *error* and *BER* for a given attack. The Modifications set to zero a percentage P ∈ [10,90] of the weights having the smallest absolute values, as described in the Modification table of [2].
The values of Table 21 shows that the watermark is successfully retrieved (BER = 0).

Table 21. Robustness of TBW against magnitude pruning for the VGG16 model and ResNet8.

P	$VGG16$		$ResNet8$	
	%	*error*	*%*	*error*
10	0	100	99	1
20	2	100	95	16
50	15	93	30	161
80	600	14	13	268
90	682	14	10	273

7.4.6 Evaluation of Computational cost

For the injection phase, the TBW method does not impact the memory footprint. However, the insertion procedure is applied during the training of the model:

- In average the mark is inserted after 504 batch iterations.
- In average the training time has been increased by 62.69%.

The detection phase involves, using TBW-Key, inference of the whole trigger set given watermarked layer and rounding their values.

8 Usability of Neural Network Traceability technologies

Table 22 and Table 23 provide samples of the performance of three Neural Network Traceability technologies measured in Section 7 of this standard.
Table 22 provides samples of Imperceptibility results, with α (the epoch of insertion for NCT), N (the number of marked parameters for NCT), λ (hyperparameter of RTW), and ρ (hyperparameter of TBW).
Table 23 provides the results for Computational Cost.
Concerning the Robustness evaluation, all methods show that the mark can be extracted before the performance of the Modified models drops.
Table 22 and Table 23 represent snapshots capturing the state-of-the-art of the general procedure and of performance of implementations. This NNT-TEC will be updated to reflect the evolution of Traceability Technologies.

Table 22 Samples of Imperceptibility results.

Method	Model	Parameter	Relative variation of the classification error
NCT	VGG16	N=64, α=0	0
		N=16144, α=90	1
	ResNet8	N=64, α=0	0
		N=16144, α=90	192
RTW	VGG16	λ= 0.01	2
	ResNet8	λ= 0.01	5
TBW	VGG16	Rho = 10	7
	ResNet8	Rho = 10	7

Table 23 Samples of Computational Cost results.

Method	Average number of batch iterations to insert the mark	Increased batch iteration time (%)
NCT	0	0
RTW	500	16.64
TBW	504	62.69