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

|  |  |
| --- | --- |
|  | **Public document** |
| **N706** | 2022/05/18 |
| **Source** | Requirements (NNW) |
| **Title** | MPAI-NNW Use cases & functional requirements WD 0.3 |
| **Target** | MPAI Community |

# Introduction

During the last decade, Neural Networks are deployed in an increasing variety of domains, but solutions and especially deep neural networks are costly. The process of AI training is costly not only in terms of resources (GPUs, CPUs, memory) but also time. According to ThinkML, the development of a custom AI solution ranges from $ 6, 000 to $ 300, 000, while renting a pre-built module would cost around $ 40, 000/year. Consequently, it becomes important to guaranty the traceability (owner) and integrity (user) of Neural Networks. Inherited from the multimedia realm, watermarking regroups a family of methodological and applicative tools allowing to **imperceptibly** and **persistently** insert some **metadata** (payload) into an original NN model and to subsequently detect/decode them from the model itself or from any of its inferences.

# Purpose of the standard

The purpose of the MPAI Neural Network Watermarking (NNW) standard is to enable watermarking technology providers to qualify their products. MPAI-NNW will provide the means to measure, for a given size of the watermarking payload, the ability of:

* The watermark inserter to inject a payload without deteriorating the performance of the Neural Network. This item requires for a given application domain:
  + A testing dataset to be used for the watermarked and unwatermarked NN.
  + An evaluation methodology to assess any change of the performance, induced by the watermark.
* The watermark detector to recognize the presence of the inserted watermark when applied to a watermarked network that has been modified (*e.g.*, by transfer learning or pruning) or to any of the inferences of the modified model. This item requires for a given application domain:
  + Performance criteria for the watermark detector (*e.g.*, relative numbers of missed detection and false alarm).
  + A list of potential modification types expected to be applied on the watermarked NN as well as of their ranges (*e.g.*, random pruning at 25%).
* The watermark decoder to successfully retrieve the payload when applied to a watermarked network that has been modified (*e.g.*, by transfer learning or pruning) or to any of the inferences of the modified model. This item requires for a given application domain:
  + A list of potential modification types expected to be applied on the watermarked NN as well as of their ranges (*e.g.*, random pruning at 25%).
  + Performance criteria for the watermark decoder (*e.g.*, 100% or (100-α)% recovery).
* The watermark inserter to inject a payload at a low computational cost, *e.g.*, execution time on a given processing environment.
* The watermark detector/decoder to detect/decode a payload from a watermarked model or from any of its inferences, at a low computational cost, *e.g.*, execution time on a given processing environment.

# Users of watermarking technology for NN

Four types of actors are identified as playing a role in the use cases.

* *NN owner* – the developer of the NN, wishing to ensure that ownership of NN can be claimed.
* *NN watermarking provider* – the developer of the watermarking technology able to carry a payload in a neural network or in an inference.
* *NN customer* – the user who needs the NN owner’s NN to make a product or offer a service.
* *NN end-user* – the user who buys an NN-based product or subscribes to an NN-based service.

# Use cases

The use cases are structured into two categories: the first relates to the NN *per se* (*i.e.*, to the data representation of the model, as discussed in Section 3.1) while the second to the inference (*i.e.*, to the result produced by the network when fed with some input data, as discussed in Section 3.2).

The use cases are presented as sequence diagrams describing the positions and actions of the four main actors in the workflow.

## Use cases related to watermarking the Neural Network model

Two types of use case belong to this category:

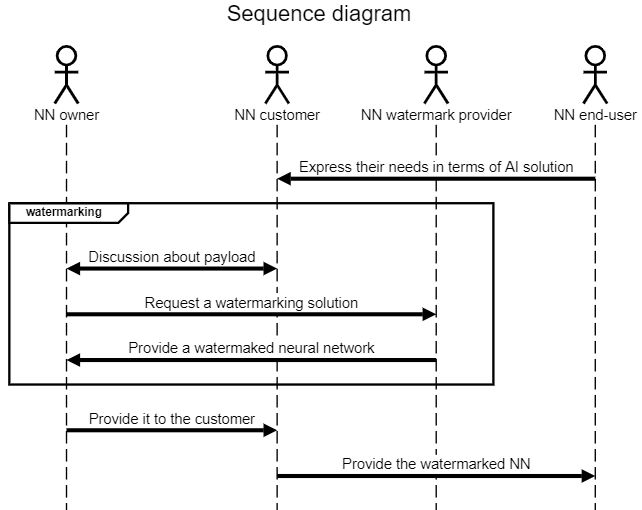
* *payload*, *i.e.*, data carried by the watermark is used to identify the actors or the model; this case is presented in Section 4.1.1.
* *loss of integrity*, *i.e.*, data carried by the watermark is used to identify modifications in the model; this case is presented in Section 4.1.2.

### Payload (channel of information)

Data is carried by the watermark is used to identify:

* the ownership of an NN.
* an NN (as if it were a DOI).

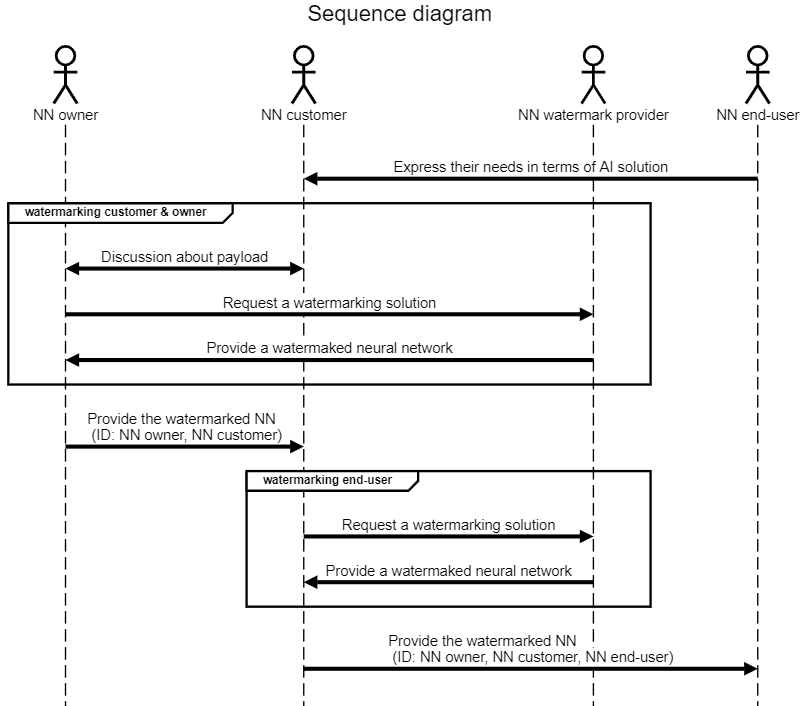
#### Identify the ownership of an NN



*Figure 1:* *Identify the ownership of an NN use case: NN owner and NN customer identifiers are inserted*

*Description of Figure 1 workflow:*

* *NN customer* gets needs from product/service from *NN end-users*.
* *NN customer* requests NN model from *NN owner* in order to be able to create the product/service requested by the end user.
* *NN customer* and *NN owner* share the need to protect NN intellectual property; *NN customer* does not want others to use the model to make similar products or offer similar services; *NN owner* wants to acquire others customers as *NN customer*; ideally, the *NN end-user* ID should also be added to the watermark (*cf*. the workflow in Figure 2).
* *NN end-user* acquires the product and/or access to the service with the embedded NN watermark.

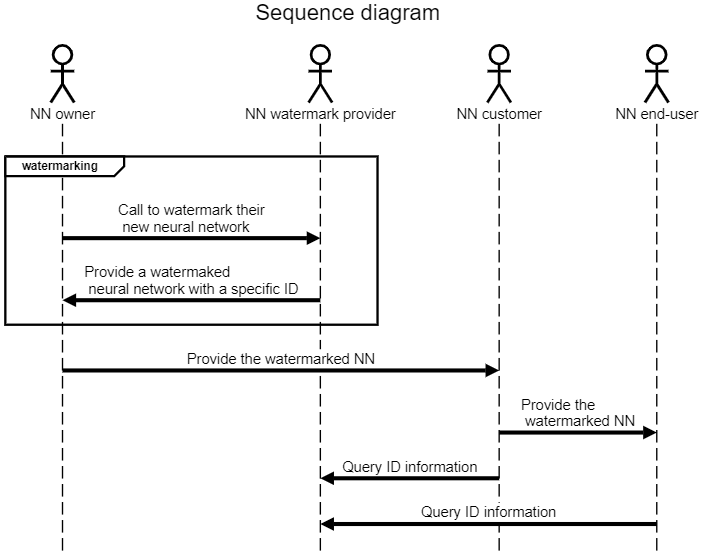


*Figure 2:* *Identify the ownership of an NN use case*: in addition to *NN owner* and *NN customer* identifiers, the *NN end-user* identifier is also inserted

*Description of Figure 2 workflow:*

* *NN customer* gets needs from product/service from end-users.
* *NN customer* requests NN model from *NN owner* in order to be able to create the product/service requested by the *NN end-user*.
* *NN customer* and *NN owner* share the need to protect NN intellectual property; *NN customer* does not want other to use the model to make similar products or offer similar services ; *NN owner* wants to acquire other customers as *NN customer*.
* *NN customer* needs to make sure that *NN end-users* do not share the AI solution, thus they insert an identifier for each *NN end-user*.
* *NN end-user* acquires the product and/or access to the service with the embedded NN watermark.

#### Identify an NN (e.g. DOI)

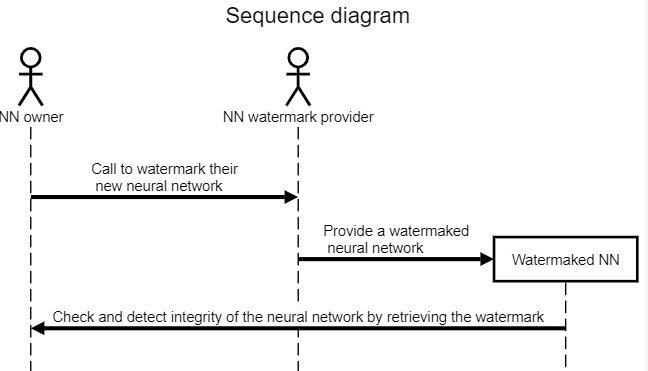


*Figure 3:* *Identify an NN use case*: *NN receives an ID (e.g. DOI)*

*Description of Figure 3 workflow:*

* *NN owner* wants its NN to receive a specific identifier.
* *NN watermark provider* gives a solution with a specific identifier for any new Neural Network and manages the ID usage through its lifecycle (e.g., validation to third parties, or ID record deletion when no longer used).

### Loss of integrity



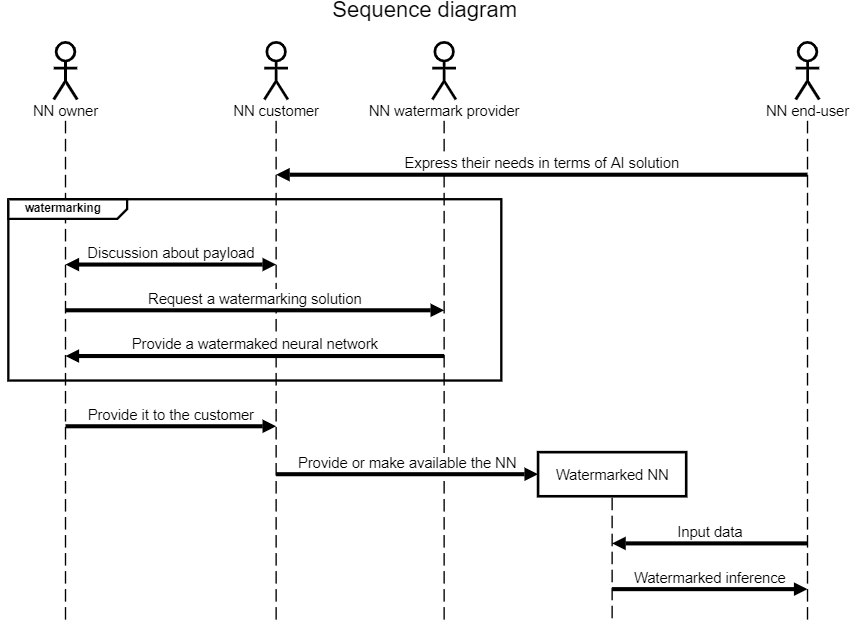
*Figure 4: Check the NN integrity use case*

*Description of Figure 4 workflow:*

* *NN owner* wants a watermark that permits them to check the integrity of the NN.
* *NN watermark provider* inserts an integrity validation watermark in the NN.
* *NN owner* can distribute the Watermaked NN to their customers.
* *NN owner* can check the integrity and detect modifications of their NN.

## Inference

The 4 use cases described in Section 4.1 are not restricted to watermarking the NN model and can be also applied to the watermarking of the NN inference. For instance, Figure 5 reflects the *Identify the ownership of an NN* use case for NN inference.



*Figure 5: Watermarked NN inference use case*

*Description of Figure 5 workflow:*

* *NN customer* gets needs from product/service from end-users.
* *NN customer* requests NN model from *NN owner* in order to be able to create the product/service requested by the end user.
* *NN customer* and *NN owner* share the need to protect NN intellectual property; *NN customer* does not want others to use the model to make similar products or offer similar services; *NN owner* wants to acquire other customers as *NN customer*.
* *NN end-user* can feed the Watermarked NN with input data and receive the inference which is watermarked. The contained ID related information can be the same as in Section 4.1 Use cases related to watermarking the Neural Network model, for instance.

## Summary of the use-cases

The use cases identified so far and presented in this document can be structured according to Figure 6.

Diagram

Description automatically generated

*Figure 6: Retrospective view on the use cases*

# Requirements

## Impact of the watermark on the performance

If the NN under test is an implementation of an NN standardized by an MPAI standard, then the MPAI Conformance Testing process specified for that NN should be used. If this is not case, the following process should be used:

1. If the NN has the input and output data format with specified semantics, use the following process:
   1. Define a test dataset of sufficient size.
   2. Feed the unwatermarked NN with the test dataset and measure the task-dependent quality of the produced inference.
   3. Feed the watermarked NN with the same test dataset and measure the task-dependent quality of the produced inference.
   4. Provide the quality statistical parameters measured in steps 1.b and 1.c.
2. If the input and output data format of the NN do not have specified semantics:
   1. Connect the NN to other NN until the input and output of the resulting configuration have input / output formats with specified semantics.
   2. Apply all steps in point 1.

**To respondents:**

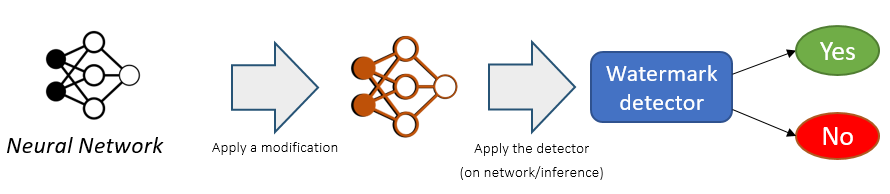
MPAI requests respondents to comment on the text described above and to propose general guidelines for applying the process above to a watermarked NN for specific application domains.

## Detection capability

The test assumes that the owner of a watermarking technology requests to test the capability of their watermark detector to reveal a mark in a potentially modified version of a watermarked NN.

To this end:

1. the owner of the watermarking technology makes available for testing their watermark detector.
2. the tester selects and sends to the technology provider a set of *M* unwatermarked NNs and *D* data payloads corresponding to the preestablished payload size.
3. the owner of the watermarking technology applies their watermarking technology to the *M* received NNs with the *D* data payloads provided by the tester and sends back the corresponding *M* x *D* watermarked NNs to the tester.
4. the tester produces a set of *M* x (*D* + 1) modified NNs (*M* unwatermarked NNs and *M* x *D* watermarked NNs), by applying one of the Modifications included Table 2, at a given Parameter value.
5. the tester:
   * applies the watermark detector to the *M* x (*D* + 1) NNs and records the binary detection result (Yes – the mark is detected or No – the mark is not detected) – see Figure 7.
   * labels the Yes/No output of the watermark detector as *true positive*, *true negative*, *false positive* (*false alarm*) and *false negative* (*missed detection*) according to the actual result – see Table 1.
   * counts the total number of false positives and the total number of false negatives.
6. the tester provides average values over the total number of tests:
   * the ratio of the number of false positives to *M* x (*D* + 1),
   * the ratio of the number of false negatives to *M* x (*D* + 1).
7. the Steps 4, 5 and 6 are repeated for requested number of parameters values, chosen in the ranges specified for the considered Modification in Table 2.
8. the Steps 4, 5, 6 and 7 are repeated for a requested set of Modifications selected from Table 2.



*Figure 7: Synopsis of the Detection capability workflow*

*Table 1: Labels assigned to the detection result for assessing the Detection capability*

|  |  |  |
| --- | --- | --- |
|  | **Detected watermark** | **Undetected watermark** |
| **Inserted watermark** | *True Positive* | *False Negative (Missed Detection)* |
| **No watermark** | *False Positive (False Alarm)* | *True Negative* |

*Table 2:* List of Modifications, their Parameters, and their ranges

|  |  |  |
| --- | --- | --- |
| Modification | Parameter type | Parameter range |
| *Gaussian noise addition*: adding a zero-mean, *S* standard deviation Gaussian noise to a layer in the NN model. This noise addition can be simultaneously applied to a sub-set of layers. | - the layers to be modified    - the ratio of *S* to standard deviation of the weights in the corresponding layer | - 1 to total number of layers    - 0.1 to 0.3 |
| *L1 Pruning*: delete the smallest *P*% of the weights, irrespective to their layers. | - the *P* percentage of the deleted weights | - 5% to 60% |
| *Random pruning*: delete *R*% of the randomly selected weights, irrespective of their layers. | - the *R* percentage of the deleted weights | - 1% to 15% |
| *Fine tuning / transfer learning*: resume the training of the *M* watermarked NNs submitted to test, for *E* additional epochs. | - ratio of *E* to the number of epochs in the initial training | - 0.1 to 0.5  (a 0.25 parameter indicates that *E* is a quarter from the initial number of epochs) |
| *Quantizing:* reduce the number of bits used to represent the weights in a layer to a smaller number *B*. This modification is detailed in Figure 8. | - the layers to be modified  - the value of *B* | - 1 to total number of layers    - 32 to 2 |
| *Weight random permutation*: randomly permute the weights of a channel in a layer, without affecting the inference of the NN. Note: keeping the same inference for the NN while permuting one channel in a layer implies the subsequent modification of the succeeding layers, as illustrated in Figure 9. | - the layer to be permuted  - the channel of the layer | - 1 to total number of layers  - 0 to 3 (dimension of the tensor) |
| *Watermark overwriting:* successively insert *R* additional watermarks, with random payloads of the same size as the initial watermark | - R number of watermark successively inserted | - 2 to 4 |

Diagram

Description automatically generated(1) an affine mapping from the interval to interval;

(2) a rounding to the closest integer in the interval

(3) a back affine mapping from the interval to the .

*Figure 8: Synopsis of the Quantizing distortion applied to NN weights*

Diagram

Description automatically generated

*Figure 9: Weight random permutation example*

## Decoding capability

The test assumes that the owner of a watermarking technology requests to test the capability of their watermark decoder to retrieve a mark in a potentially modified version of a watermarked NN.

To this end:

1. the owner of the watermarking technology makes available for testing their watermark decoder.
2. the tester selects and sends to the technology provider a set of *M* unwatermarked NNs and some data payload.
3. the owner of the watermarking technology applies their watermarking technology to the *M* received NNs with the data payload provided by the tester and sends back the corresponding *M* watermarked NNs to the tester.
4. the tester produces a set of 2 x *M* modified NNs (*M* unwatermarked NNs and *M* watermarked NNs), by applying one of the modifications included in the list above, at a given parameter value.
5. the tester:
   * applies the watermark decoder to the 2 x *M* NNs and computes the Hamming distance between the decoded watermarks and their related original data payload.
   * for any of the 2 x *M* NNs, computes the BER (bit error rate) as the ratio of the Hamming distance to the size of the data payload.
   * compute the average BER, as the average (over 2 x *M*) of the BER values computed in the previous step.
6. the test results are the *M* number for tested NNs, and the average BER.
7. the Steps 4, 5 and 6 are resumed for different parameters values, in the ranges specified for the considered modification.
8. the Steps 4, 5, 6 and 7 are resumed for any (or a subset of) modifications included in the list below.

## Processing cost

The test assumes that the owner of a watermarking technology requests to test the processing cost of their watermark insertion and detection/decoding to a “classic” processing environment.

The watermark provider makes available is method of insertion, of detection and of decoding to the tester. The tester take *M* NNs standardize by MPAI. For each of them, the tester uses the following process:

1. Inserts a watermark of a given payload to each *M* NNs while collecting:
   1. The computational time in second
   2. The memory footprint for the process
2. Uses the detector to reveal a watermark in each *M* NNs while collecting:
   1. The computational time in second
   2. The memory footprint for the process
3. Uses the decoder to retrieve the watermark in each *M* NNs while collecting:
   1. The computational time in second
   2. The memory footprint for the process
4. Provides the quality statistical parameters measured in steps 1.a and 1.b; 2.a and 2.b; 3.a and 3.b.

(is it meaningful to specify a processing configuration? Will the resulting number an interest ?)

# Annex 1 – MPAI-NNW Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
|  |  |
| Attacks | Any transformation, malicious or not, applied after the mark injection. Attacks can be of various types:  *Removal attacks*: turn undetectable/unreadable the information conveyed by the watermark.  *Geometric attacks*: destroy the watermark synchronization, rather than to remove it.  *Cryptographic attacks*: detect and remove the mark without knowledge of the key, thanks to the knowledge of the embedded mark.  *Protocol attacks*: embed another watermark and/or create an ambiguous situation upon the mark detection/decoding. |
| Black-Box | Do not grant access to the network only to its inference. |
| Data payload | The amount of information injected through watermarking process. |
| Fine-tuning | Apply an additional training to the network on the same task. |
| Imperceptibility | Inference quality of the model on its original task should not be degraded significantly through the watermark injection. |
| Knowledge distillation | A surrogate model is trained by using the inference of a targeted network as label. |
| Modification | A method used to simulate an attack for the purpose of NN testing |
| Ownership metadata | The data carried by the watermark representing the owner and the usage conditions. |
| Parameter | A set of values characterizing the strength of a Modification |
| Pruning | Remove a portion of a network's parameters while keeping its performances. |
| Quantization | Reducing the number of bits required for describing the model parameters. |
| Robustness | The ability of the watermark to withstand a prescribed class of attacks |
| Transfer-learning | Apply an additional training to the network on another (similar or not) task. |
| Watermark decoder | An algorithm able to decode an inserted watermark, when applied to a watermarked network. |
| Watermark detector | An algorithm able to detect an inserted watermark, when applied to a watermarked network. |
| Watermark forging | A particular case of *Protocol attacks*, where the attacker tries to forge the watermark to create an ambiguous situation where both parties can claim the ownership. |
| Watermark overwriting | A particular case of *Protocol attacks*, where the attacker embeds another watermark to create an ambiguous situation where both parties can claim the ownership. |
| White-Box | Grant access to the network and make it possible for the watermark to be embedded inside the parameters. |

# Annex 2 – MPAI-NNW References

[1] Y. Li, H. Wang, and M. Barni, “A survey of deep neural network watermarking techniques,” *arXiv:2103.09274 [cs]*, Mar. 2021, Accessed: Feb. 08, 2022. [Online]. Available: <http://arxiv.org/abs/2103.09274>

[2] F. Boenisch, “A Systematic Review on Model Watermarking for Neural Networks,” *Frontiers in Big Data*, vol. 4, 2021, Accessed: Feb. 08, 2022. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fdata.2021.729663>

[3] M. Xue, Y. Zhang, J. Wang, and W. Liu, “Intellectual Property Protection for Deep Learning Models: Taxonomy, Methods, Attacks, and Evaluations,” *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 01, pp. 1–1, Dec. 2021, doi: [10.1109/TAI.2021.3133824](https://doi.org/10.1109/TAI.2021.3133824).

[4] H. Chen, B. D. Rouhani, and F. Koushanfar, “BlackMarks: Blackbox Multibit Watermarking for Deep Neural Networks,” *arXiv:1904.00344 [cs]*, Mar. 2019, Accessed: Mar. 23, 2022. [Online]. Available: <http://arxiv.org/abs/1904.00344>

[5] E. Tartaglione, M. Grangetto, D. Cavagnino, and M. Botta, “Delving in the loss landscape to embed robust watermarks into neural networks,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, Milan, Italy, Jan. 2021, pp. 1243–1250. doi: [10.1109/ICPR48806.2021.9413062](https://doi.org/10.1109/ICPR48806.2021.9413062).

[6] H. Chen, B. D. Rouhani, C. Fu, J. Zhao, and F. Koushanfar, “DeepMarks: A Secure Fingerprinting Framework for Digital Rights Management of Deep Learning Models,” in *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, New York, NY, USA, Jun. 2019, pp. 105–113. doi: [10.1145/3323873.3325042](https://doi.org/10.1145/3323873.3325042).