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

**Public document**

|  |  |
| --- | --- |
| **N750** | 2022/06/22 |
| **Source** | Requirements (NNW) |
| **Title** | MPAI-NNW Use cases & functional requirements WD 0. |
| **Target** | MPAI Community |

# Introduction

During the last decade, Neural Networks have been 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 for both owner and user to guaranty the traceability and integrity 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 these metadata 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 ascertain the presence and the Watermark decoder to successfully retrieve the payload of the inserted watermark when applied to:
	+ A watermarked network that has been modified (*e.g.*, by transfer learning or pruning)
	+ An inference of the modified model.

This item requires for a given application domain:

* + Performance criteria for the Watermark detector or decoder, *e.g.*, relative numbers of missed detection and false alarm or percentage of the retrieved payload.
	+ A list of potential Modification types expected to be applied to the watermarked NN as well as of their ranges (*e.g.*, random pruning at 25%).
* The watermark inserter to inject a payload at a quantifiable computational cost, *e.g.*, execution time on a given processing environment.
* The Watermark detector/decoderto detect/decode a payload from a watermarked model or from any of its inferences, at a quantifiable 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 4.1) 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 4.2).

The use cases presentation includes sequence diagrams describing the positions and actions of the four main actors in the workflow.

## Use cases related to watermarking the Neural Network model

In this set of use cases, watermarking is embedded into a Neural Network model. The subsequent watermark detection provides either the identity of the actors and models, or information about the loss of integrity of the 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.

In this set of use cases, the Neural Network Watermarking methods require that the Neural Network model be available during both the stages of watermarking embedding and watermarking detection/decoding.

### Payload

Data carried by the watermark can be used to serve various application domains, for instance 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



*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

The purpose of this use case is to detect and/or localize any modifications induced in the NN model, by embedding a fragile watermark inside a Neural Network.



*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

In this set of use cases, watermarking is embedded into the inference of a Neural Network. The subsequent watermark detection provides either the identity of the actors and models, or information about the loss of integrity of the model.

The four 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.

In this set of use cases, the Neural Networking methods require that the Neural Network model be available only during the embedding stage, while the decoding/detection stage is based on the inference produced by the model when fed with application data.



*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.



*Figure 6: Retrospective view on the use cases*

**To respondents:**

MPAI requests respondents to comment on the use cases described above and on the coverage of relevant applications.

New use cases are welcome.

# Requirements

## Impact of the watermark on the performance

The objective of this section is to evaluate the potential task-dependent impact of the embedding of a watermark inside a Neural Network. The Task may be specified by the watermark technology provider or selected by the Tester.

The test assumes that the owner of a watermarking technology requests to evaluate the impact of their watermark embedding method on the inference of a watermarked NN.

To this end:

* If the NN has the input and output data format with specified semantics, use the following process:
	1. the Tester defines a pair of train and test datasets of sufficient size.
	2. the Tester selects and sends to the technology provider the train dataset (if needed), a set of *M* unwatermarked NNs trained on the train dataset and *D* data payloads corresponding to the pre-established payload size.
	3. the owner of the watermarking technology applies their watermarking technology to the *M* received NNs while processing the train data set and the *D* data payloads provided by the Tester and then sends back the corresponding *M* x *D* watermarked NNs to the Tester.
	4. the Tester:
		1. feeds the M unwatermarked NN with the test dataset and measure the task-dependent quality of the produced inference.
		2. feeds the M x D watermarked NN with the same test dataset and measure the task-dependent quality of the produced inference.
		3. provides the task-dependent quality of the produced inference measured in 1.d.i and 1.d.ii.
* 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 propose a list of Tasks.
* to comment on the process described above.
* to propose methods to measure the quality of the inference produced in 1.d.i and 1.d.ii.

## Detection capability

The objective of this section is to evaluate the capability of a NN watermarking method detector to ascertain the presence of a watermark inside a watermarked NN and not to ascertain as watermarked NN that is not watermarked.

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 their Watermark detector for testing.
2. the Tester selects and sends to the technology provider a set of *M* unwatermarked NNs, *D* data payloads corresponding to the pre-established payload size and, if needed, the train dataset.
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 at a given Parameter value one of the Modifications to be adopted in the standard. Table 2 provides examples of such Modifications.
5. the Tester:
	1. applies the Watermark detector to any of the *M* x (*D* + 1) NNs and records the corresponding binary detection results (Yes – the mark is detected or No – the mark is not detected) – see Figure 7.
	2. labels the Yes/No outputs 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.
	3. 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:
	1. the ratio of the number of false positives to *M* x (*D* + 1),
	2. 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 to be adopted in the standard.
8. the Steps 4, 5, 6 and 7 are repeated for a requested set of Modifications to be adopted in the standard.



*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 possible Modifications, their Parameters, and their ranges

|  |  |  |
| --- | --- | --- |
| Modification | Parameter type | Parameter range |
| *Gaussian noise addition*: [6] 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*: [5, 6] delete the smallest *P*% of the weights, irrespective to their layers. | - the *P* percentage of the deleted weights | - 5% to 60% |
| *Random pruning*: [1] 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*: [1,2,3] 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 1 |
| *Quantizing: [5]* reduce the number of bits used to represent the weights in a layer to a smaller number *B*; The procedure for reducing the number of bits is based on a sequence of three operations (affine mapping from the weights interval to the $(0;2^{B}-1)$, rounding to the closest integer, and backward affine mapping towards the initial weights interval), and is illustrated 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, 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: [1,2,3]* 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  |

1(a 0.25 parameter indicates that *E* is a quarter from the initial number of epochs)

|  |  |
| --- | --- |
| Diagram  Description automatically generated | **(1)** an affine mapping from the $(w\_{min};w\_{max})$ interval to $(0;2^{B}-1)$ interval;**(2)** a rounding to the closest integer in the $(0;2^{B}-1)$ interval**(3)** a back affine mapping from the $(0;2^{B}-1)$ interval to the $(w\_{min};w\_{max})$.  |

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

|  |  |
| --- | --- |
| Diagram  Description automatically generated | Assume a baseline model composed of two convolutional layers: 1 layer with 2 neurons (each neuron having 2 channels) and another layer with 3 neurons (each neuron having 2 channels). When permuting the two neurons in the first layer, the two channels of each neuron in the second layer should be permuted accordingly.  |

*Figure 9: Weight random permutation example: keeping the same inference for the NN while permuting one channel in a layer implies the matched permutation of the succeeding layers.*

**To respondents:**

MPAI provides Table 2 as an example. MPAI requests respondents to propose Modifications, Parameters and their ranges. The accepted proposals may be modified, structured as in Table 3, and included in the standard.

*Table 3:* List of Modifications, their Parameters, and their ranges

|  |  |  |
| --- | --- | --- |
| Modification name | Parameter type | Parameter range |
| Modification method #1 | Parameter #1Parameter #2 | Range #1.1Range #1.2Range #2.1 |
| Modification method #2 | … | … |
| … |  |  |

## Decoding capability

The objective of this section is to evaluate the capability of a NN watermarking method decoder to retrieve the watermark from a potentially modified watermarked NN or from its inferences.

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 their Watermark decoder for testing.
2. the Tester selects and sends to the technology provider a set of *M* unwatermarked NNs, *D* data payloads corresponding to the pre-established payload size and, if needed, the train dataset.
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 in Table 3.
5. the Tester:
6. applies the Watermark decoder to any of the *M* x (*D* + 1) NNs and computes a Distance between the outputs of the decoder and their corresponding original data payloads.
7. for any of the *M* x (*D* + 1) NNs, computes the SER (symbol error rate) as the ratio of the distanceto the size of the corresponding data payload.
8. computes the average SER, as the average (over *M* x (*D* + 1) ) of the SER values computed in the previous step.
9. the test results are the *M* x *D* number for tested NNs, and the average SER.
10. the Steps 4, 5 and 6 are resumed for different Parameters values, in the ranges specified for the considered Modification.
11. the Steps 4, 5, 6 and 7 are resumed for any (or a subset of) Modifications included in the list below.

**To respondents:**

MPAI requests respondents to comment on the suitability of Table 3 to evaluate the decoding capability or to propose a new Table.

MPAI requests respondents to propose and motivate a suitable Distance for point 5.i and 5.ii.

## Processing cost

The objective of this section is to specify the process of evaluating the processing cost of watermarking solutions (in terms of resources and/or time).

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

To this end:

1. Watermarking insertion:
	1. the Tester selects and sends to the technology provider a set of *M* unwatermarked NNs, *D* data payloads corresponding to the pre-established payload size, and the type of testing environment (CPU, GPU).
	2. the owner of the watermarking technology applies their watermarking technology to the *M* received NNs with the *D* data payloads, on a computing environment corresponding to the specification provided by the Tester.
	3. the owner of the watermarking technology sends back the corresponding *M* x *D* set of values characterizing the processing.
	4. the Tester provides the statistical average of the quality over the total number of tests (over *M* x *D*) for the same type of testing environment:
2. Watermarking detector:
	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 pre-established 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 applies the Watermark detector to the *M* x *D* NNs and records the set of values characterizing the processing.
	5. the Tester provides the statistical average of the quality over the total number of tests (over *M* x *D*) for the same type of testing environment:
3. Watermarking decoder:
	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 *D* data payloads corresponding to the pre-established 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 applies the Watermark decoder to the *M* x *D* NNs and records the set of values characterizing the processing.
	5. the Tester provides the statistical average of the quality over the total number of tests (over *M* x *D*) for the same type of testing environment:

**To respondents:**

MPAI requests respondents to propose and motivate a set of testing environments with their characteristics, *e.g.*, CPU types, number of cores, frequency, memory; GPU types, frequency, memory.

MPAI requests respondents to propose a set of values characterizing the processing required by the NNs, *e.g.*, execution time (in second), CPU footprint (in MB) and GPU footprint (in MB).

# 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. |
| Distance | A measure of the difference from one dataset to another, *e.g.*, the Hamming distance or a correlation coefficient.  |
| Imperceptibility | Inference quality of the model on its original task should not be degraded significantly through the watermark injection. |
| 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 |
| Robustness | The ability of the watermark to withstand a prescribed class of attacks |
| Symbol Error Rate (SER) | Symbol Error Rate or Multi-Symbol Error Rate represents the error between the retrieved character compared to the embedded. If there is only two symbol (bits) we refer it as Bit Error Rate (BER). |
| Task | A specific use of the watermarked Neural Network, such as classification, multimedia coding, *etc*. |
| Tester | An entity executing a testing process to be specified by the standard.  |
| 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 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 weights. |

# 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).