**III.1 Introduction**

Deep Neural Networks (DNNs) are increasingly deployed and commercialized in a wide variety of real-world scenarios due to the unprecedented performance they achieve. Training a DNN is a very expensive process that requires: (i) the availability of massive amounts of often proprietary data capturing different scenarios within the target application; ii) extensive computational resources; iii) the assistance of Deep Learning (DL) experts to carefully fine-tune the network topology (e.g., the type and number of hidden layers), and correctly set the training hyper-parameters, like the learning rate, the batch size, etc. As a consequence, high performance DNNs should be considered as the intellectual property (IP) of the model owner and be protected accordingly. Inspired by the use of classical watermarking techniques for the protection of property rights associated to multimedia contents, DNN watermarking is receiving increasing attention, and several works have been published leveraging on digital watermarking to address IP protection in the DL domain.

In the last two decades, watermarking technology has been applied to protect multimedia documents, and a vast body of literature has been developed as summarized in several books and surveys. The watermark can be injected into the host document by adding to it a low amplitude, often pseudo-random, signal, either directly in the sample domain or in a properly transformed domain. In the case of still images, for instance, the watermark can be embedded in the spatial domain by adding a low amplitude spread spectrum signal or by substituting the least significant bits (LSB) of the pixel values. In the case of transformed domain watermarking, several reversible transforms like the Discrete Cosine Transform (DCT), the Discrete Wavelet Transform (DWT), or the Discrete Fourier Transform (DFT) are widely used to embed the watermark in a robust and imperceptible way [1].

With all that been said, in this chapter we’re going to take a deep dive with extreme details on how watermarks are embedded/extracted in DNN with its taxonomy and all the requirements needed for the process to achieve decent results like setting up or generating a good dataset for the training, testing the efficiency of the algorithm by applying a different verity of attacks (compression, sharpening, gaussian blur ... etc.).

Also getting familiar with some terminology in DNN and image processing in general terms as NC, epochs and some other terms used in compression measurement like PSNR ... etc.

**III.2. Why DNN watermarking:**

We deal with images in many domains. Now we are processing signals (images) in frequency domain. Since this Fourier series and frequency domain is purely mathematics, till now, all the domains in which we have analyzed a signal, we analyze it with respect to time. But in frequency domain we don’t analyze signal with respect to time, but with respect of frequency.

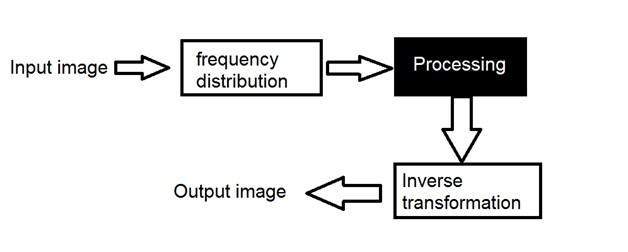
In spatial domain, we deal with images as it is. The value of the pixels of the image change with respect to scene. Whereas in frequency domain, we deal with the rate at which the pixel values are changing in spatial domain [2].

For simplicity, Let’s put it this way.



**Figure III.1:** Special domain overall model.

In simple spatial domain, we directly deal with the image matrix. Whereas in frequency domain, we deal an image like this: We first transform the image to its frequency distribution. Then our black box system performs whatever processing it has to performed, and the output of the black box in this case is not an image, but a transformation. After performing inverse transformation, it is converted into an image which is then viewed in spatial domain. It can be pictorially viewed as:



**Figure III.1:** Frequency domain overall model.

A signal can be converted from time domain into frequency domain using mathematical operators called transforms. There are many kinds of transformation that does this we discussed some of them in the previous chapter.

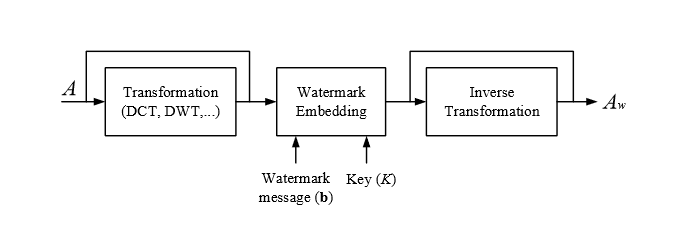
As an answer to the question as to why we use DNN in watermarking and not a static watermarking, becuase the classic interpretation of images does not work if a marked image gets modified, for instance LSB method wouldn’t be able to extract the right watermark if the cover image gets modified by a compression algorithm although the original and the compressed image does not seem to be diffident, but a DNN would even if some pixels values changed on the marked image.

**III.3. A taxonomy of DNN watermarking**

In this section, we present a taxonomy of DNN watermarking techniques. The taxonomy takes into account the similarities and dissimilarities between multimedia and DNN watermarking, some of which have already been outlined in the introduction. To start with, we review the most important classification criteria used to categorize classic watermarking algorithms, discussing their applicability to the DNN case. Later, we introduce some unique classification criteria based on the peculiar characteristics of DNN watermarking. Eventually, we discuss the relationship between the various watermarking classes.

**3.1. Classical watermarking models**

Classically, digital watermarking aims at embedding a certain message, called watermark, within a hosting multimedia content, often, but not only, for copyright protection. Watermark injection follows the general model depicted in Figure 2. The to-be-watermarked content *A* is possibly transformed so that the watermark is embedded in a different domain (e.g. the DCT or wavelet domain for still images)1, then the watermark is injected within the document,  
with the possible intervention of a watermarking key *K*. The watermarked content is then brought back into the original domain to obtained the watermarked document *Aw*. As simple as this scheme may seem, it contains some implicit assumptions that are not necessarily satisfied in the case of DNN watermarking: i) the existence of a to-be-watermarked content *A* and *ii*) the consequent possibility of measuring the distortion between the original and the watermarked contents, *A* and *Aw*. This is not necessarily the case with DNN watermarking, where the network weights, which are going to host the watermark, may not exist perse, since they are generated contextually to watermark embedding during the training phase. Moreover, the distortion introduced by the watermark cannot be evaluated directly by measuring the difference between *A* and *Aw*. Rather, the impact of the watermark must be measured by evaluating the performance achieved by the watermarked model. Note that watermark embedding (and later on watermark recovery) may require the knowledge of a secret key *K*. Such a key, whose main goal is to introduce some secrecy within the watermarking system, is commonly used to parameterize the embedding process and make the recovery of the watermark impossible for non-authorized users who do not know *K*.



**Figure III.3:** Overall view of classical watermark embedding.

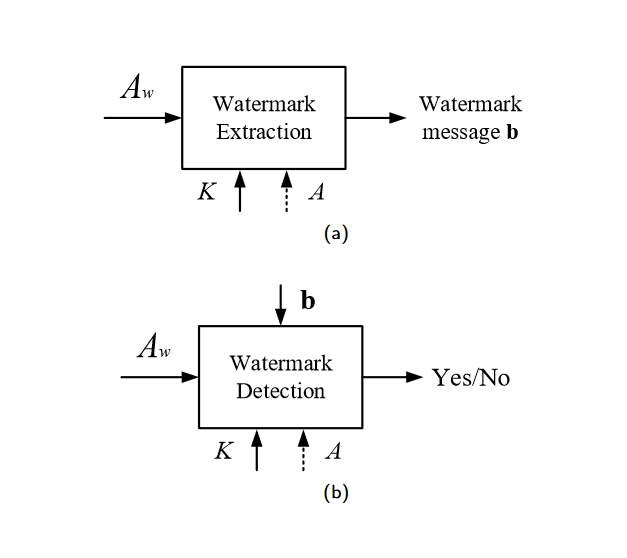
**3.1.1. Multi-bit vs. zero-bit watermarking techniques**

Depending on the exact content of the watermark message, and the way such a message is recovered from the host signal, we can distinguish between two main classes of algorithms: multi-bit and 0-bit watermarking. In the multi-bit case, the watermark message corresponds to a sequence of *N* bit **b**. Such a sequence can be read from the watermarked content, as depicted in part (a) of Figure 3. In the 0-bit case, watermark extraction corresponds to a detection task, wherein the detector is asked to decide whether a known watermark is present in the analyzed content or not. In some applications, a mixture of the two functionalities is required. The detector must first determine if a watermark is present and, if so, identify which of the 2N messages is encoded. Such a detector would therefore have 2N + 1 possible output values [25]. In the zero-bit case, only one possible watermark exists. In this case only 20 + 1 = 2 outputs are possible, thus justifying the zero-bit watermarking term. In general, multi-bit watermarking offers more flexibility2 and hence it can be used in a wider variety of applications, including fingerprinting [26], error concealment [27], source tracking [28], labelling etc. Zero-bit watermarking generally achieves a higher robustness and is widely adopted in copyright protection platforms, where the presence of the watermark is used as a flag to warn compliant devices that the piece of content, they are dealing with is copyrightedmaterial [29]. The distinction between multi-bit and zero-bit watermarking is also appropriate for DNN watermarking techniques.

**3.1.2. Robust vs. fragile watermarking**

Watermark robustness accounts for the capability of the hidden data to survive host signal manipulations, including non-malicious and malicious manipulations. Malicious manipulations precisely aim at damaging or removing the watermark, on the contrary, non-malicious manipulations indicate some unintentional or unavoidable processing that may  
perturb the hidden watermark, for instance, multimedia compression occurring during image or video transmission. Robust watermarking requires the watermark to be resistant against non-malicious manipulations, whose application is implied by the normal use of the data, such as data compression, editing and cropping. A secure watermark, on the contrary, should survive also malicious manipulations. If the hidden watermark is lost or irremediably altered as soon as any modification is applied to the host content, the watermark is said to be fragile. Data authentication is the main application for fragile watermarking, because the alteration of the watermark can be taken as evidence that the content has been tampered with.

With regard to DNN watermarking, most of the applications considered so far call for the adoption of a robust watermark, that can survive at least some common operations like  
fine-tuning or nodes pruning. Watermark security may also be required in specific applications wherein the presence of an adversary aiming at watermark removal can not be ruled out.



**Figure III.4:** Multibit (a) vs zero-bit (b) watermarking

**3.1.3. Blind vs. non-blind detection**

From Figure 4, we can see that for both the multi-bit and the zero-bit cases, watermark recovery may require the knowledge of the non-watermarked content *A*. In classic watermarking theory, we say that a watermarking algorithm is blind3 if the watermark is recovered without resorting to the comparison between the original non-marked content and the marked one. Conversely, non-bind watermarking algorithms use the original non-marked content to ease the extraction. As we already noted, in the case of DNN watermarking, the concept of original non-watermarked content does not apply, so the distinction between blind and non-blind watermarking does not make sense.

**3.1.4. Informed watermarking**

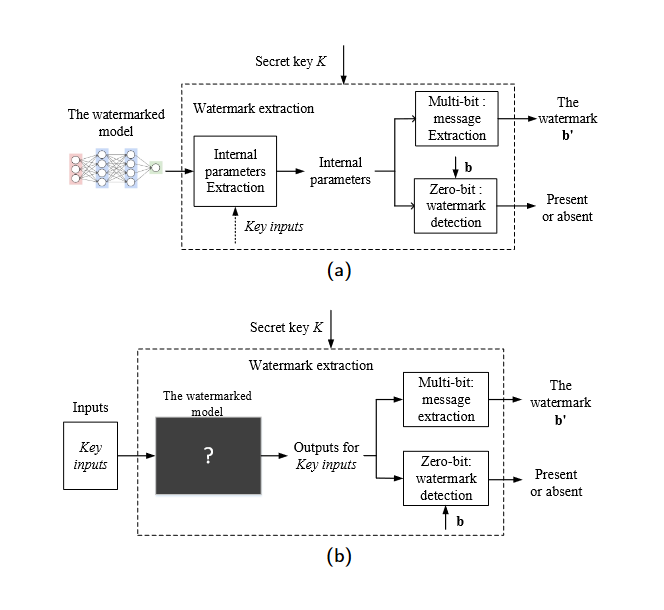
Informed embedding and informed coding are watermarking paradigms that have been proven to greatly improve the performance of a watermarking system [30]. Such paradigms stem from the interpretation of watermarking as a problem of channel coding with side information at the transmitter [31, 32]. In a nutshell, with informed coding each watermark message is associated to a pool of codewords (rather than to a single one), then informed watermark embedding is applied by choosing the codeword that results in the minimum distortion. Practical implementations of the informed coding paradigm include several popular watermarking schemes like Quantization Index Modulation (QIM) [33], Dither Modulation (DM) [34] and Scalar Costa’s Scheme (SCS) [35].

Informed watermarking theory provides powerful means to improve the performance of watermarking algorithms, with particular reference to multi-bit watermarking, since its adoption permits to increase significantly the watermark payload without sacrificing its robustness. Nevertheless, such a theory has not been applied extensively to DNN watermarking. A DNN watermarking algorithm exploiting informed coding to increase the watermark payload is described in Section 4.

**3.2. DNN watermarking models**

Deep learning is a machine learning framework which automatically learns hierarchical data representation from training data without the need to resort to handcrafted feature representations [36]. DNNs are the most common learning architectures on which deep learning methods are based. To be specific, a DNN takes as input the content *x* ∈ ℝm to be  
processed in raw format (for instance an image or an audio signal), and maps it to the output via a parametric function, *z* = FƟ(*x*), where *z* ∈ [1, *n*]. The parametric function FƟ (⋅) is defined by the network architecture and the collective parameters of all the units composing it. Given the architecture, the network behavior is determined by the values of the network parameters Ɵ (in most cases Ɵ corresponds to the network weights and offsets). Let *D* = {*xi,,zi* }Ti=1 be the training data, where *zi* ∈ [1, *n* ] is the ground truth label for *xi*. During the training phase, the network parameters are optimised to minimise a loss function expressing the difference between the predicted class labels and the ground truth labels. Currently, the most widely used approach for training a DNN is the back-propagation algorithm, whereby the network parameters are updated by propagating the gradient of the loss from the output layer through the entire network.  
DNN watermarking is based on the observation that the huge number of parameters Ɵ consists of, allows to slightly modify them, without that the performance of the network is de-  
graded significantly. The parameters in Ɵ, then, can be used to encode additional information beyond what is required for the primary task of the network. While the classical taxonomy of watermarking algorithms described in the previous section can be largely applied also to DNN watermarking, in the following, we introduce two new characterisations that are specific for DNN watermarking.

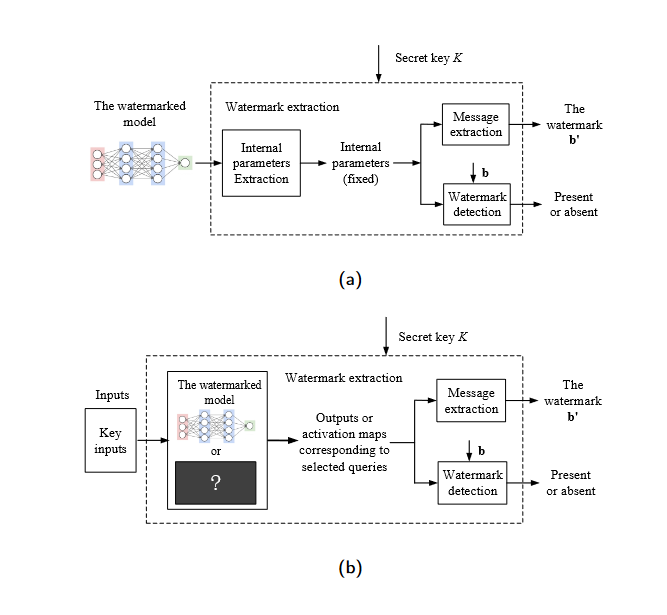
**3.2.1. White-box vs. black-box DNN extraction**

Based on the data accessible to the watermark extractor,we can divide DNN watermarking techniques into white-box and black-box techniques. As depicted in Figure 4a, if the internal parameters of the DNN models are available, we say that watermark recovery is carried out in a white-box modality. The internal parameters may correspond directly to the model weights, or to the activation of the neurons in correspondence to specific inputs. In the case of multi-bit watermarking, the extractor, now called watermark decoder, extracts the concrete message bits the watermark consists of. For zero-bit watermarking, the watermark detector, must decide about the existence of a specific watermark. In some algorithms (e.g. [37, 38, 39, 40]), the key *K* is chosen independently from the original loss function E0 and the internal parameters Ɵ, during a key generation step. With black-box watermarking (see Figure 5b), only the final output of the DNN is accessible. In this case, the watermark is recovered by querying the model and checking the output of the DNN in correspondence to a set of properly chosen inputs.

**Figure III.5:** White-box (a) vs black-box (b) DNN watermark recovery.

During the entire decoding or detection process, the architecture and the internal parameters of the DNN model are totally blind to the decoder or detector. In other words, the only thing that can be controlled are the inputs used for querying the network and the outputs corresponding to the queries. Thus, the main target of black-box watermarking is to train the DNN model in such a way that it outputs particular labels *zi*for certain inputs *xi*’s. The inputs *xi*’s may be secret or not. In the former case they play a role similar to a decoding/detection key.

**3.2.2. Static vs. dynamic DNN watermarking**

A very important distinction of DNN watermarking techniques leads to the definition of static and dynamic watermarking. Static DNN watermarking methods, like those described in [37, 38, 39], embed the watermark into the weights of the DNN model. Such weights are determined during the training phase and assume fixed values that do not depend on the input of the network. With dynamic watermarking, instead, the watermark is associated to the behavior of the network in correspondence to specific inputs, sometimes called triggering inputs, or key inputs. Even in this case, the watermark is embedded by properly chosen the network weights, however the watermark is retrieved indirectly by looking at the impact of the watermark on the behavior of the network. If the watermark is recovered by looking at the final output of the model (sometimes referred to as DNN-output watermarking), the watermark can be extracted in a black-box manner, since access to the internal status of the network is not required. When the watermark is associated to the activation map of the neurons in correspondence to certain inputs, as in [40], white-box extraction is required. The distinction between static and dynamic watermarking is illustrated in Figure 6.

**Figure III.6:** Static (a) vs Dynamic (b) DNN watermarking

**3.3. Relationship among different DNN watermarking classes**

Table 1 summarises the various perspectives that can be adopted to classify watermarking techniques and their applicability to conventional (multimedia) and DNN watermarking. The symbol ✓ indicates that the corresponding classification criterion is suitable for a specific watermarking technique, while 🗶 represents inappropriateness.

Analyzing the intersection among these classifications, we see that dynamic watermarking can be implemented both in white box (activation-map based watermarking) and in black-box modality (DNN-output watermarking). On the contrary, static watermarking can only be achieved in a white box mode, since it requires the access to the internal parameters of the network. In other words, white box can be static and dynamic, but black-box can only be dynamic. Referring to prior researches, white-box methods have been used for both multi-bit, and zero-bit watermarking, while the black box modality, due to its lower expressivity, is more often used in conjunction with zero-bit watermarking.

**Table 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Taxonomy of classic and DNN watermarking. | | | |
|  | Multi/Zero-bit | Robust/Fragile | Blind/Non-blind |
| **Classic** | **✓** | **✓** | **✓** |
| **DNN** | **✓** | Robust only | **🗶** |
|  | Informed/Non-informed | White/Black-box | Static/Dynamic |
| **Classic** | **✓** | **🗶** | **🗶** |
| **DNN** | **✓** | **✓** | **✓** |

**III.4. Requirements**

In this section, we pause to discuss the main requirements that DNN algorithms must satisfy. While many of them are inherited by classical multimedia watermarking, there are some important differences that are worth attention. The most common requirements for DNN watermarking are summarised in Table 2, and briefly discussed in the following.

**4.1. Robustness**

Together with security, robustness is one of the three corners of the watermarking trade-off triangle. It refers to the possibility of recovering the watermark from a perturbed version of the host content, be it a multimedia document or a DNN. Usually, it is required that the watermark survives at least the most common manipulations that the host content may undergo in its lifecycle. For DNNs, the two most common operations the watermark should be robust to are: fine-tuning and network pruning.

**Fine-tuning.** Fine tuning is a common operation related to transfer learning. It consists in retraining a model that was initially trained to solve a given task, so to adapt it to solve a new task (possibly related to the original one). Computationally, fine-tuning is by far less expensive than training a model from scratch, hence it is often applied by users to adapt a pre-trained model to their needs. Needless to say, when sufficient training data is not available for the user, model fine-tuning can help avoiding over-fitting problems, at least to a certain extent. Since fine-tuning alters the weights of the watermarked model, it is necessary to make sure that the watermark is robust against a moderate amount of finetuning.

**Network pruning.** This is a common strategy to simplify a complex DNN model to deploy it into low power or computationally weak devices like embedded systems or mobile devices. During pruning, the model weights whose absolute value is smaller than a threshold are cut off to zero. The threshold is set in such a way that the accuracy of the model does not decrease significantly. Of course, network pruning changes the internal parameters of the model, hence it is necessary to make sure that the embedded watermark is resistant to this operation.

**Table 2**

|  |  |
| --- | --- |
|  | List of requirements for DNN watermarking. |
| **Requirement** | ***Description*** |
| Robustness | The embedded watermark should resist different kinds of processing. |
| Security | The watermark should be secure against intentional attacks from an unauthorized party. |
| Fidelity | **T**he watermark embedding should not significantly affect the accuracy of the target DNN architectures. |
| Capacity | A multi-bit watermarking scheme should allow to embed as much information as possible into the host target DNN |
| Integrity | The bit error rate should be zero (or negligible) for multibit watermarking and the false alarm and missed detection probabilities should be small for the zero-bit case |
| Generality | The watermarking methodology can be applied to various DNN architectures and datasets. |
| Efficiency | The computational overhead of watermark embedding and extraction processes should be negligible. |

**4.2. Security**

This criterion deals with malicious manipulations, by assuming that an attacker, partially or fully aware of the watermarking algorithm, is present and attempts to destroy it, with the further goal of using the watermarked content illegally. For a secure DNN watermarking algorithm, the loss of the watermark should be obtainable only at the expense of a significant degradation on the quality of the host model. Until now, research has focused mainly on two kinds of intentional attacks, watermark overwriting and surrogate model attack.  
**Watermark overwriting.** A third-party who is aware of the methodology used for DNN watermarking (but does not know the private key used to embed the watermark) may try  
to embed a new watermark in the model and overwrite the original one. An overwriting attack, then, inserts an additional watermark into the model to make the original watermark undetectable. Several degrees of security can be defined according to the knowledge the attacker has about the watermark. When the watermark is inserted directly in the network weights, for instance, more powerful attacks can be conceived if the attacker knows the exact layers the watermark is embedded in.

**Surrogate model attack.** In a surrogate model attack, the adversary tries to replicate the functionality of a target DNN, by feeding it with a series of requests and using the output provided by the network to train a surrogate model, which mimics the original network. During the attack, the adversary has no knowledge on the exact architecture of the model and limited access to the original training dataset. Ideally, if the network targeted by the attack is watermarked, the surrogate network trained by querying the watermarked  
model should also inherit the watermark, thus making it possible to recover the watermark from the surrogate model.

**4.3. Fidelity**

Fidelity represents the second corner of the watermarking trade-off triangle. It basically requires that the presence of the watermark does not degrade the quality of the watermarked object. In the case of DNNs, this means that the watermark should have a limited impact on the performance of the watermarked model *F*Ɵ’. More specifically, the watermarked model *F*Ɵ’ should guarantee a performance level that is as close as possible to that achieved by a model *F*Ɵ trained on the same task without caring about the watermark. As we will see later on, in some cases, training the network by adding a watermark embedding term to the loss function can even be beneficial, since it reduces the risk of overfitting. In the following sections, fidelity will be measured in terms of the Test Error Rate (TER) achieved on the task the model is designed for.

**4.4. Capacity**

Capacity is the third corner of the watermarking trade off triangle. It is defined as the number of bits the water mark message consists of. As such, it can be referred only to multi-bit watermarking, since in the zero-bit case, the watermark does not convey any payload. While a large payload is a desirable feature of a watermarking algorithm, increasing the payload conflicts directly with the robustness requirement. The most straightforward way to increase the robustness, while also observing the fidelity criterion, in fact, is to spread the watermark over a large number of host samples (the network weights in the DNN case), thus rapidly exhausting the room available for watermark embedding. As an alternative, error correction coding can be used on top of the watermarking algorithm, to allow the recovery of some  
bits possibly lost because of the manipulation of the DNN. Even in this case, however, the room available to host the net payload decreases, thus raising again the necessity to find a  
tradeoff between the number of redundancy bits, which in turn determines the error correcting capability of the code, and the actual payload of the watermark.

**4.5. Integrity**

In the absence of model modifications, the extracted watermark *b’* should be equal to *b*. We use the Bit Error Rate (BER) to evaluate the integrity of a multi-bit algorithm. Ideally, in the absence of processing or attacks, the BER should be equal to zero. Classically, this is a characteristic achievable by the class of informed watermarking algorithms [41], while it cannot be guaranteed by non-informed spread spectrum techniques [42]. Interestingly, several DNN watermarking algorithms directly generate the model weights in such a way that the watermark can be extracted without errors. This resembles very closely the informed embedding paradigm [30], according to which embedding is achieved by applying a signal dependent perturbation to the to-be-watermarked sequence. In this way, the embedding procedure results in a watermark which, in the absence of modifications, can be recovered with no errors. The integrity requirement assumes a different meaning in the case of zero-bit watermarking. In this case, watermark recovery can be stated as a detection problem and as such is prone to two kinds of errors: false detection and missed detection4. The former refers to the probability that the presence of the watermark is a detected in a non-watermarked model, while the second indicates the probability that the watermark can not be recovered from a watermarked network. As it is known from statistical detection theory [43], decreasing the false detection probability comes at the price of an increase of the missed detection probability (and viceversa), so a trade off must be looked for. In DNN watermarking, it is pretty easy to design the watermark embedding algorithm in such a way that the missed detection probability is equal to zero5, while it is impossible to guarantee that the false detection probability is equal to zero. The only possibility, then, is to carry out a statistical analysis to give an estimate of the false detection probability and design the watermarking algorithm so that such a probability is acceptably small.

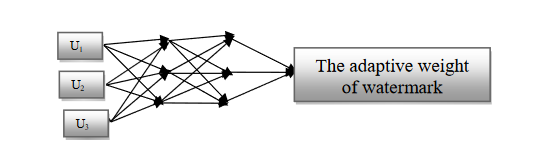
**4.6. Other Watermark Requirements**

Based on the choice of a specific watermarking modality among those listed in Section 2.2, and on the envisaged application, other criteria may need to be satisfied.

Generality: To be effective, a DNN watermarking algorithm should be applicable to a wide variety of architectures carrying out different tasks. In this sense, the generality of an algorithm refers to its applicability to architectures and models other than those on which the algorithm was initially tested and developed. This also implies that the performance of the algorithm do not depend too heavily on the to-be-marked model.

Efficiency: Efficiency refers to the computational overhead needed to train the DNN by simultaneously teaching the model to carry out the the target task and embedding the watermark. Training a DNN is always an expensive process, embedding the watermark into the DNN should not add an unaffordable extra burden to it.

**III.5. Application of Deep Neural Network in Digital Image Watermarking**

 Artificial neural network is important embranchment of artificial intelligence field, and it has certain advantage on the aspect of simulating biology neural computation [44], [45]. It has self-learning, self-organizing, association of ideas, blur extend abilities, etc. and has great comparability with human visual system. Therefore, the selected NN in this paper is a feed-forward NN. There are three layers including an input layer, hidden layers and an output layer. The desired outputs of NN are the maximum watermarking strength. Figure 7 shows   
the architecture of this network. The first layer is to present the input variables of network. Each HVS feature vector has the elements to be input variables in this study. The four feature components form a feature vector U = (U1, U2, U3, U4). The U1, U2, U3 and U4 represent the sensitivity value of luminance, frequency, texture, and entropy of respectively. Each u represents one of the input components in NNS algorithm and each pixel generates a corresponding feature vector u. These inputs are then fed to a hidden layer using tan sigmoid   
activation functions (output in the range -1 to 1); this is then fed to the output layer. During training samples, information is propagated back through the network and used to update connection weights. It repeats learning many times for every example in the training set until it has minimized the output errors. This neural network, when combined with the available rules in perceptual watermarking of an image and ability to query the network, does provide an automatic system for generating near maximum power invisible watermarked images.   
This system does not fully replace humans, its they are still needed to generate the training data and should check the results periodically to ensure the neural network is working at peak performance.

**Figure III.7:** Architecture of NN with textural features.

**III.6. conclusion**

The demand of methods for protecting the Intellectual Property Rights associated to DNNs has pushed researchers to develop a new class of algorithms to embed a watermark into DNN models. In a few years, several techniques have bene proposed, sometimes based on naive arguments and sometimes by relying on solid theoretical bases. By looking at the methods proposed so far, it is evident that watermarking a DNN model presents some peculiarities that must be taken into account when trying to apply general multimedia watermarking principles to DNNs. For this reason, we have started our review by highlighting the main differences and similarities with classical multimedia watermarking, and introducing a new specific taxonomy explicitly thought to highlight the characteristics of the various DNN watermarking algorithms proposed so far. Then we have reviewed the main algorithms available for each watermarking category, pointing out the main advantages and drawbacks characterising the various approaches. By the light of the properties and limits of the watermarking algorithms proposed so far, the most important challenges researchers are going to face with in the years.

**References :**

[1]: A survey of deep neural network watermarking techniques by Yue Lia, Hongxia Wangb and Mauro Barnic.

[2]: <https://www.tutorialspoint.com/dip/introduction_to_frequency_domain.htm>

[3]: