

# Wide Flat Minimum Watermarking for Robust Ownership Verification of GANs

Jianwei Fei, Zhihua Xia, *Member, IEEE*, Benedetta Tondi, *Member, IEEE*, and Mauro Barni, *Fellow, IEEE*

**Abstract**—We propose a novel multi-bit box-free watermarking method for the protection of Intellectual Property Rights (IPR) of GANs with improved robustness against white-box attacks like fine-tuning, pruning, quantization, and surrogate model attacks. The watermark is embedded by adding an extra watermarking loss term during GAN training, ensuring that the images generated by the GAN contain an invisible watermark that can be retrieved by a pre-trained watermark decoder. In order to improve the robustness against white-box model-level attacks, we make sure that the model converges to a wide flat minimum of the watermarking loss term, in such a way that any modification of the model parameters does not erase the watermark. To do so, we add random noise vectors to the parameters of the generator and require that the watermarking loss term is as invariant as possible with respect to the presence of noise. This procedure forces the generator to converge to a wide flat minimum of the watermarking loss. The proposed method is architecture- and dataset-agnostic, thus being applicable to many different generation tasks and models, as well as to CNN-based image processing architectures. We present the results of extensive experiments showing that the presence of the watermark has a negligible impact on the quality of the generated images, and proving the superior robustness of the watermark against model modification and surrogate model attacks.

**Index Terms**—GAN watermarking, DNN watermarking, ownership verification, watermark robustness, deep learning security

## I. INTRODUCTION

IN the past decade, deep learning has rapidly gained popularity due to its superior performance compared to previous approaches and even humans, in various tasks and application scenarios, including computer vision, healthcare, finance, and manufacturing. The success of deep learning relies on well-designed and trained Deep Neural Networks (DNN), whose training requires the availability of massive amounts of labeled data, computational resources, and expertise. One concern is that attackers who are unable to train such models themselves may steal or use them without respecting the license terms under which they are made available. Therefore, it is important that means to protect the IPR of DNN models are developed.

Inspired by the use of multimedia watermarking for IPR protection [1], several researchers have proposed DNN wa-

Jianwei Fei is with the College of Cyber Security, Jinan University, Guangzhou, China, and the University of Siena, Siena, Italy (e-mail: fjh826244895@163.com).

Zhihua Xia is with the College of Cyber Security, Jinan University, Guangzhou, China, and also with the Engineering Research Center of Digital Forensics, Nanjing University of Information Science and Technology, Nanjing, China (e-mail: xia\_zhihua@163.com).

Benedetta Tondi and Mauro Barni are with the University of Siena, Siena, Italy (e-mail: benedettatondi@gmail.com, barni@dii.unisi.it).

termarking as a solution to protect the IPR of DNN models [2]. With DNN watermarking the ownership information is embedded directly or indirectly into the model parameters. Subsequently, the ownership can be verified by matching the extracted watermark with the owner's watermark.

Generally speaking, DNN watermarking methods can be split into 3 categories according to the way the watermark is extracted from the network [2]. In white-box methods, the watermark is read from the internal parameters/status of the network, thus the decoder is assumed to have full access to the model. In black-box methods, the decoder can only access the final output of the network and the watermark is recovered by querying the model with some specific inputs and looking at the output corresponding to those inputs. Eventually, for generative or image processing networks, the watermark can be read from any arbitrary sample generated by the watermarked model (box-free methods). Box-free methods do not require any knowledge about the model parameters, or the use of key inputs to interact with the model. For this reason, whenever applicable, e.g., with GANs or any other kinds of generative models, box-free methods provide a flexible way to prove the ownership of a network by looking only at the output samples it produces.

Early research has focused mainly on black-box and white-box methods, however protecting the IPR of GANs (and other image processing and generative models) by relying on box-free watermarking is a promising approach that deserves to be investigated further. So far, there have been only a few works on GAN watermarking, including black-box methods [3], [4], [5], along with some box-free methods [6], [7], [8]. Most existing black-box GAN watermarking methods are zero-bit schemes whereby it is only possible to determine whether the analyzed model contains a given known watermark or not. On the contrary, most box-free methods proposed so far are multi-bit watermarking schemes, where the watermark bits can be extracted from any output of the watermarked model without knowing them in advance. In this case, the authentication process involves matching the extracted watermark to the bits identifying the hypothesized model owner. Due to their flexibility and large payload, multi-bit box-free watermarking schemes are in general preferable to black-box and white-box methods for GANs. However, this is not the case when we consider robustness against watermark removal attacks, especially model-level removal attacks. While existing black-box and white-box methods have shown good robustness against model-level watermark removal attacks, box-free methods are not robust enough to be used in practical scenarios.

In this paper, we propose a robust, box-free, multi-bit GAN watermarking method based on the concept of Wide Flat

Minimum (WFM), which significantly improves the robustness against model-level watermark removal attacks, including fine-tuning, pruning, quantization, and surrogate model attacks, with respect to prior methods belonging to the same category.

In the embedding process, a pre-trained image watermarking decoder is applied to the generated images, and the generator is trained in such a way that the decoder correctly reads a pre-defined watermark from the generated images. To do so, we add a dedicated watermarking loss term to the loss function used to train the GAN. In order to overcome the robustness limitations of current GAN watermarking schemes, we force the model to converge to a WFM of the watermarking loss. In this way, the watermarking loss tends to be invariant to small and even moderate modifications of the model parameters, hence ensuring that the watermark can survive model manipulations, like pruning, quantization, and, most importantly, fine-tuning. To the best of our knowledge, our scheme is the first multi-bit GAN watermarking scheme achieving strong robustness against white-box model-level attacks, including fine-tuning and surrogate model attacks, without impairing the quality of the generated images. Such a result is proven by means of extensive experiments carried out on several GAN and CNN architectures addressing different tasks. Note that although in this paper we mainly focus on GAN architectures, our method is applicable to a wide range of generative models that produce images as output.

With the above ideas in mind, the contributions of this paper can be summarized as follows:

- We present a new, multi-bit, box-free GAN watermarking method with strong robustness against model-level attacks, including fine-tuning and surrogate model attacks. To the best of our knowledge, this is the first multi-bit box-free GAN watermarking achieving such a level of robustness.
- Our method is architecture- and task-agnostic and thus can be used on a variety of image generation or processing tasks, including image translation, image synthesis, super-resolution, etc.
- We prove, under a rigorous hypothesis testing framework, that the proposed method can satisfy the watermark accuracy requirements for model ownership authentication under the considered attacks.

The rest of this paper is organized as follows. A review of related works is carried out in Section II. The problem definition and threat model are presented in Section III. Then, the proposed method is described in Section IV. The experimental methodology and the results of the experiments are presented in Sections V and VI respectively. The paper ends in Section VII with a summary of our findings and some concluding remarks.

## II. RELATED WORK

According to a widely accepted taxonomy [2], DNN watermarking methods can be categorized into white-box, black-box, and box-free methods. Moreover, depending on the information carried out by the watermark and the way the watermark information is extracted from the hosting network,

DNN watermarking methods can be classified as multi-bit and zero-bit methods. In multi-bit watermarking, the watermark is a sequence of bits that can be extracted from the watermarked model without knowing it in advance. On the other hand, with zero-bit watermarking the watermark detector can only determine whether a known watermark is present in the model or not. In this section, we first give a general review of existing watermarking methods, then we describe existing box-free GAN watermarking methods in more detail.

Uchida *et al.* [9] were the first to use digital watermarking for IPR protection of CNNs adopting a white-box setting. In their approach, a regularization term is incorporated in the loss used to train the target model, in order to impose a statistical bias on the parameters. This statistical bias ensures that a known binary watermark can be extracted from the model parameters using a preset projection matrix. Subsequently, numerous works on white-box watermarking have been proposed [10], [11], [12], [13], [14], [15]. White-box watermarking directly embeds watermark bits into the statistics of the model parameters, thus requiring white-box access for watermark detection/decoding. This limitation hinders the practical use of white-box watermarking, leading to a shift of focus toward black-box watermarking.

Black-box DNN watermarking methods assume that the parameters of the target model cannot be accessed, but the verifier is capable of querying the model with specific inputs, referred to as key inputs or trigger inputs. The presence of the watermark can be detected by analyzing the output of the watermarked model in correspondence to the key inputs. Existing research focuses on the construction of the key inputs and on the strategies to train the network in such a way to enforce specific outputs in correspondence to the key inputs [16], [17], [18], [11], [19]. Black-box watermarking methods typically adopt the zero-bit watermarking paradigm, which lacks the flexibility necessary for applications like fingerprinting and traitor tracing. In addition, even if with black-box methods watermark extraction does not require white-box access to the inspected model, the security of the watermark may be at risk due to the necessity of exposing the key inputs during the watermark verification procedure [20], [17].

### A. Box-free Watermarking

Unlike classification models, generative models, particularly those based on GANs, generate images (or other kinds of documents) with a large entropic content, which can naturally serve as carriers for multi-bit watermarks. Therefore, researchers have proposed box-free watermarking methods specifically designed for generative models, utilizing the images generated by the network as watermark carriers.

Yu *et al.* [6] were the first to propose a multi-bit box-free GAN watermarking method. They first embed a certain watermark into the training data by using a DNN image watermarking network (specifically Stegastamp [21]), then use the watermarked data to train the target GAN. They demonstrate that the same watermark contained in the training data can be extracted from any image generated by the GAN. Yu *et al.* [7] also proposed a scalable fingerprinting method

inspired by the style-based encoder used in styleGAN [22]. By parameterizing the convolution kernels of the generator with different watermarks, and by forcing the generator to produce images with the input watermark, their method allows efficient and scalable generation of numerous models with distinct watermarks, yet maintaining identical functionalities. Wu *et al.* [23] utilize images as watermarks and optimize the generator and a decoder with a combined loss function, ensuring that the decoder outputs only specific watermark images when receiving images generated by the target generator. Similarly, Fei *et al.* [4] have proposed a method to watermark GANs by enforcing the presence of a specific watermark within the generated images. Their method utilizes a pre-trained image watermarking decoder to impose an additional watermark loss term during training. Considering that with box-free methods, the watermark is extracted from the generated images, it is crucial to take into account the robustness of the watermark against image processing attacks. In most cases, such a robustness is achieved by introducing a noise layer to simulate image processing attacks during watermark embedding.

In addition to image processing attacks, it is also important to consider the robustness against model-level attacks, as usually done for both white-box and black-box methods. This aspect, however, is rarely investigated in papers proposing box-free methods. Typically, if an attacker obtains white-box access to the target model, e.g. by stealing it or by violating the license agreement, he/she can carry out model-level watermark removal attacks, such as fine-tuning, pruning, and quantization, before utilizing the model. The goal of these attacks is to modify the watermarked GAN to let it generate images that do not contain the expected watermark, thereby disabling the ownership verification functionality. Even if the attackers have only black-box access to the watermarked model, he/she can still carry out a surrogate model attacks, by training a similar model using the input-output pairs of the target model. These watermark removal attacks represent a serious threat to box-free GAN watermarking methods, however they have rarely been considered in existing research. In this work, we propose a novel GAN watermarking scheme for IPR protection achieving good robustness against both image-level and model-level attacks, significantly outperforming state-of-the-art methods in the latter case.

### III. PROBLEM DEFINITION AND THREAT MODEL

In this section, we introduce the notations used throughout the paper, describe the threat model adopted in our work, and detail the hypothesis testing problem underlying the watermark-based owner verification protocol.

#### A. Notation

A GAN consists of a generator  $G_\theta$  parameterized by the set of weights and biases  $\theta$  and a discriminator  $D_{gan}$ . As an example, we may consider an image-transfer GAN taking as input image  $x$  belonging to the input domain and mapping it into an image  $G_\theta(x)$  belonging to the output domain. Our objective is to watermark  $G_\theta$  in such a way that any image  $G_\theta(x)$  generated by the GAN carries an invisible bit string

(the watermark)  $w \in \{0, 1\}^n$  of length  $n$ . The string  $w$  can be extracted by a watermark decoder  $D_w$ . More formally, according to the box-free watermark retrieval paradigm, given any image  $y = G_\theta(x)$  produced by the GAN, we let

$$\hat{w} = D_w(y). \quad (1)$$

Then, the extracted watermark  $\hat{w}$  is compared with the watermark bit sequence we are looking for (that is  $w$ ). If  $D_w(G_\theta(x))$  is equal to  $w$  or the difference between  $w$  and  $\hat{w}$  is smaller than a threshold, we say that  $G_\theta$  contains  $w$  and the ownership of  $G_\theta$  is proven. Given that both the model and the generated images may undergo a number of intentional or non-intentional modifications, we require that the watermark be retrievable from the modified images and even by the images produced by a modified version of the generator (say  $G_{\theta'}$ ).

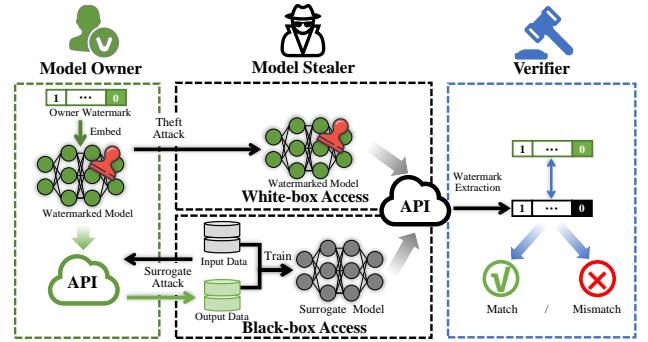


Fig. 1: The threat model considered in our work.

#### B. Threat Model

The threat model considered in this paper (see Fig. 1) involves three entities: the model owner, an attacker, or model stealer, and the verifier, who can be a third party or the model owner himself. The owner makes the model available through an API allowing the users to query the model and download the generated images. To avoid that the model is stolen and re-used without authorization, the owner embeds a watermark within it, to make it possible to prove the ownership of the model itself or that of any model derived from it, e.g. after fine-tuning. The ownership of the model is proven by verifying that the images it produces contain the watermark of the owner. The presence of the watermark within an image can also be used to prove that the image has been generated by the stolen model or by a model derived from it.

The goal of the attacker (stealer) is to steal the target model and use it for its own purposes without being traced. Specifically, we establish 2 scenarios based on the accessibility of the target model.

- a) In the first scenario, the stealer may be an internal attacker or a skilled hacker who manages to create his own copy of the model. The stealer may also attempt to remove the watermark by modifying the model and/or by post-processing the generated images.
- b) In the second scenario, the attacker can access the model in a black-box modality, by querying it through the API made available by the owner. In this case, the attacker may

exploit the input-output pairs obtained through the API to train a new *surrogate* model and use it for his own goals.

We assume that after the attack, the verifier and the model owner can not access the suspicious model directly, rather they have access to the images produced by the model either through an API or by gathering such images on the Internet, through social networks, or by any other means. The objective of the verifier is to extract the watermark from the images produced by the suspicious model to verify whether it corresponds to the model of the legitimate owner or is derived from it.

### C. Ownership Verification

Ownership verification is accomplished by running a hypothesis test on the presence of the watermark within the images generated by the suspicious model. To begin with, we assume that the verifier has access to only one image produced by the model, namely  $y = G_\theta(x)$ . Note that, following the box-free paradigm, we assume that the verifier does not know, or cannot choose the input  $x$  used to generate  $y$ . The null hypothesis  $H_0$  states that  $y$  is not generated by the inspected model, while the alternative hypothesis  $H_1$  states that  $y$  is generated by the model.

Under  $H_0$ , the bits  $\hat{w}$  obtained by applying the watermark decoder are independent of the owner's watermark sequence  $w$ , so the number of matching bits  $k$  between  $w$  and  $\hat{w}$  follows a binomial distribution with matching probability  $p = 0.5$ :

$$P(k) = \binom{n}{k} 0.5^n, \quad (2)$$

where  $n$  is the length of the watermark. By assuming that the presence of the watermark is verified when at least  $T$  bits of  $\hat{w}$  match  $w$ , the false detection probability ( $P_f$ ) is equal to :

$$P_f = \sum_{k=T}^n \binom{n}{k} 0.5^n. \quad (3)$$

Given  $n$  and the maximum allowed false detection probability, the above equation permits to compute the detection threshold. The threshold values corresponding to  $n = 100$  and various  $P_f$  are reported in Table I.

TABLE I: Detection thresholds  $T$  for different values of  $P_f$  ( $n = 100$ ).

$P_f$	0.05	0.01	$10^{-3}$	$10^{-4}$	$10^{-5}$	$10^{-6}$
$T$	55	60	65	68	71	74

To calculate the missed detection probability ( $P_m$ ) under  $H_1$ , let us assume that the bit error probability is  $\varepsilon$ , and let us indicate with  $p_b = 1 - \varepsilon$  the corresponding bitwise decoding accuracy. By assuming that watermark bit errors are independent of each other, the probability of  $k$  matches between  $w$  and  $\hat{w}$  is now:

$$P(k) = \binom{n}{k} p_b^k (1 - p_b)^{n-k}, \quad (4)$$

and the missed detection probability can be computed as:

$$P_m = \sum_{k=0}^{T-1} \binom{n}{k} p_b^k (1 - p_b)^{n-k}. \quad (5)$$

In Table II, we list the minimum watermark accuracy  $p_b$  required for different values of  $P_f$  and  $P_m$ . It is clear from the table, that to get sufficiently low values of  $P_f$  and  $P_m$ , the bitwise accuracy  $p_b$  must be sufficiently large. For instance, in order to get  $P_f = P_m = 10^{-4}$  ( $n = 100$ ), it is necessary that  $p_b \geq 0.83$ .

TABLE II: Minimum required  $p_b$  for different values of  $P_f$  and  $P_m$  when  $n = 100$ .

$P_m \backslash P_f$	0.05	$10^{-2}$	$10^{-3}$	$10^{-4}$	$10^{-5}$	$10^{-6}$
0.05	0.60	0.65	0.69	0.72	0.75	0.77
$10^{-2}$	0.65	0.70	0.74	0.77	0.79	0.81
$10^{-3}$	0.70	0.74	0.78	0.81	0.83	0.85
$10^{-4}$	0.73	0.77	0.81	0.83	0.85	0.87
$10^{-5}$	0.75	0.80	0.83	0.86	0.87	0.89
$10^{-6}$	0.78	0.83	0.86	0.88	0.89	0.90

## IV. METHODOLOGY

In this section, we first introduce the concept of Wide Flat Minimum, and then we provide a thorough description of the watermark embedding process and the involved loss functions.

### A. What is a Wide Flat Minimum?

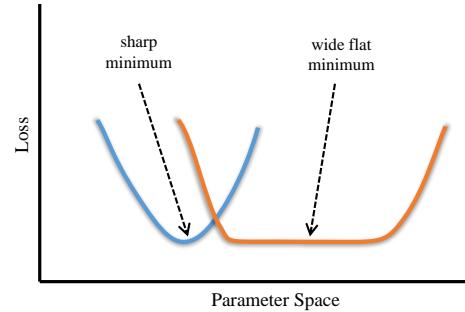


Fig. 2: Comparison of a sharp minimum and a wide flat minimum in the one-dimensional case.

A Wide Flat Minimum of a neural network refers to a large region in the parameter space where the loss value is small and approximately constant [24]. Fig. 2 shows a toy example of a flat wide minimum and a sharp minimum in a one-dimension case. Requiring that training converges to a wide flat minimum has been proven to be helpful for the generalization capability of neural networks [25], [26]. Recent works have proposed to look for a wide flat minimum for domain generalization [26], better robustness against adversarial attacks [25], and to fight catastrophic forgetting [27]. In particular, our approach is inspired by [27], where the authors propose to alleviate catastrophic forgetting in continual learning by searching for a wide flat local minimum of the loss associated with the old task. They do so, by adding random noise to the model parameters during training and, jointly optimizing multiple instances of the losses, so that the loss values around the minimum are also small. When the model is fine-tuned to accomplish a new task, the model can efficiently learn the new task without forgetting the original task it had been trained on initially.

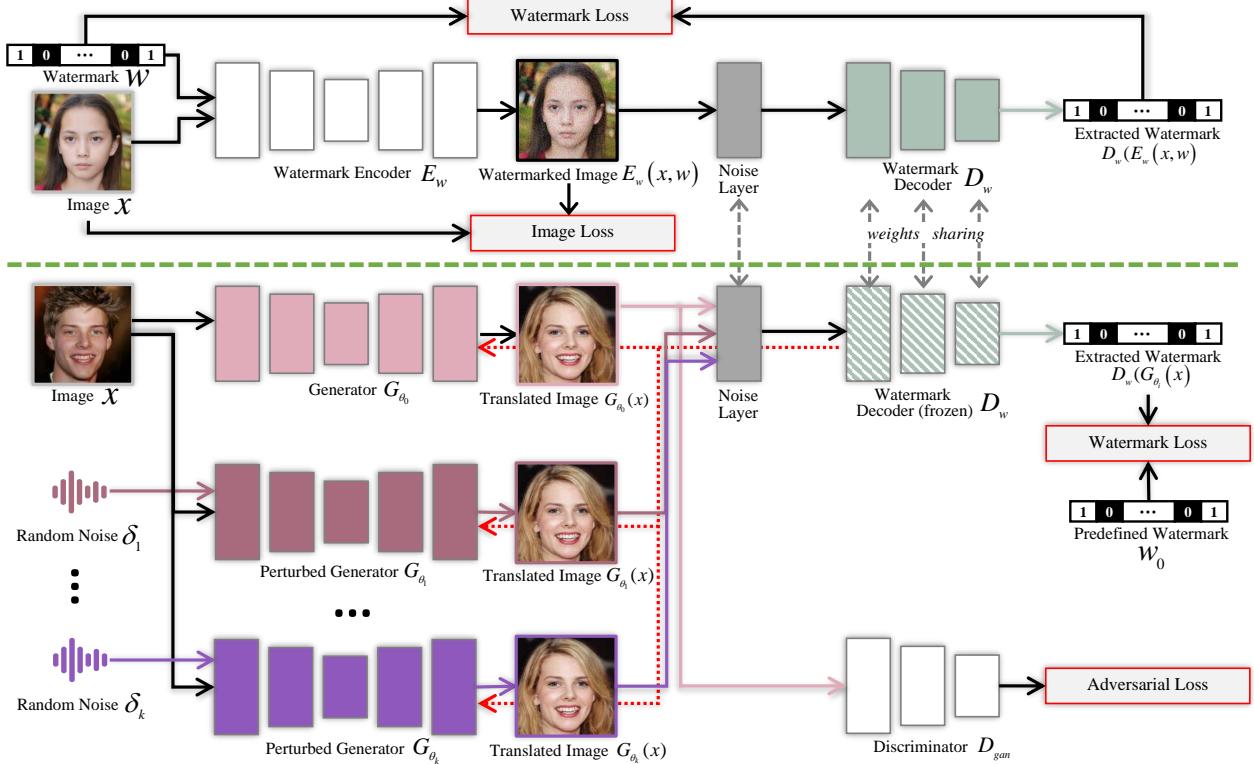


Fig. 3: Overview of the proposed method using an image translation GAN (male→female) as an example. Before watermarking the target GAN, we train an image watermarking network. Then, during GAN watermarking, the watermark decoder is frozen and used to extract the watermark bits from the generated images. The loss between a pre-defined watermark and the extracted watermark is back-propagated to instruct the generator to generate images containing the pre-defined watermark. A noise layer is introduced before the watermark decoder to enforce robustness against image-level attacks. For each iteration, we additionally sample  $k$  random noise vectors and add them to the generator parameters, resulting in  $k$  perturbed generators. The watermarking loss is calculated also on the perturbed generators, and the resulting gradients are used to update the generator. The red dashed lines indicate the backpropagation path of the watermarking loss from the frozen decoder to the generator.

The design of a box-free GAN watermarking method which is robust against fine-tuning has some similarities with the necessity of avoiding catastrophic forgetting, given that fine-tuning the model should not imply forgetting the watermark embedded within the network during the initial training. In other words, the tasks addressed when the network is trained include image generation and watermark embedding ruled, respectively, by the adversarial and the watermark loss. During fine-tuning the network is trained on a new task by using only the adversarial loss. Given the flatness and width of the minimum of the watermark loss, the new task can be learned without *forgetting* the watermark.

### B. WFM watermarking

As shown in Fig. 3, the WFM watermarking method proposed in this paper consists of two stages. In the first stage, we train a watermark encoder  $E_w$  and a decoder  $D_w$ , which are jointly optimized by using an image reconstruction loss and a watermarking decoding loss:

$$\mathcal{L}_w = \lambda_1 \text{MSE}(x, E_w(x, w)) + \lambda_2 \text{BCE}(w, D_w(E_w(x, w))), \quad (6)$$

where MSE and BCE stand, respectively, for Mean Squared Error and Binary Cross Entropy, and  $\lambda_1$  and  $\lambda_2$  are weights balancing the two terms. With regard to the second stage, let us consider a GAN whose goal is to learn a mapping function  $G_\theta : X \rightarrow Y$  between domain  $X$  and domain  $Y$ .  $X$  may be a class of images belonging to a certain domain in image transfer applications, or a noise sample drawn from the uniform distribution, as it happens when the GAN is asked to generate images belonging to the output domain from scratch. In our method, the loss used to train  $D_{gan}$  is not changed and can be expressed by:

$$\begin{aligned} \mathcal{L}_{D_{gan}} = & -\mathbb{E}_{y \sim Y} [\log D_{gan}(y)] \\ & -\mathbb{E}_{x \sim X} [\log(1 - D_{gan}(G_\theta(x)))] \end{aligned} \quad (7)$$

where  $D_{gan}$  is asked to output 1 for original images and 0 for synthetic ones. To let the generated images contain the owner watermark, say  $w_o$ , we modify the loss of  $G_\theta$  as follows:

$$\begin{aligned} \mathcal{L}_{G_\theta} = & \mathbb{E}_{x \sim X} [\log(1 - D_{gan}(G_\theta(x)))] \\ & + \gamma \text{BCE}(w_o, D_w(G_\theta(x))), \end{aligned} \quad (8)$$

where  $\gamma$  is a weight balancing the adversarial loss and the watermarking loss<sup>1</sup>. By optimizing  $\mathcal{L}_{G_\theta}$  in Eq. (8), the gener-

<sup>1</sup>To prevent falling into trivial solutions,  $D_w$  is frozen in this stage.

ator learns to generate realistic images in the output domain containing the owner's watermark  $w_o$ <sup>2</sup>.

To increase the robustness of the watermark against image-level processing, we include a noise layer between the generated images and the watermark decoder, which simulates image-processing attacks. In this way, the generator learns to embed a watermark that can be correctly retrieved even after several kinds of post-processing.

With regard to the robustness of the watermark against model-level manipulations (e.g., pruning, quantization, fine-tuning and surrogate model attacks), we force the training procedure to converge to a wide flat minimum of the watermarking loss term, so that any perturbation of the model within a certain range results in a loss value close to the minimum, with a negligible impact on the watermark decoding accuracy. To this end, at each iteration, the watermarking loss term is computed on  $k$  perturbed versions of the current parameters vector, namely  $\theta_i = \theta_0 + \delta_i$ ,  $i = 1 \dots k$ , where  $\delta_i$  is a perturbation vector taking values in the interval  $[-b, b]$ , and  $\theta_0$  is the parameters vector at the current iteration. Then the watermarking loss gradient used to update  $\theta_0$  is computed on all parameter vectors,  $\theta_0, \theta_1, \dots, \theta_k$ , rather than on  $\theta_0$ , while the GAN loss is calculated as usual on  $\theta_0$  only. Accordingly, the updating rule is given by :

$$\theta'_0 = \theta_0 - \frac{\alpha}{k} \sum_{i=1}^k \nabla \mathcal{L}_w(\theta_i) - \beta \nabla \mathcal{L}_G(\theta_0), \quad (9)$$

where  $\theta'_0$  is the updated parameters vector,  $\nabla \mathcal{L}_w(\theta_i)$  is the gradient of the watermarking loss computed on  $\theta_i = \theta_0 + \delta_i$ , and  $\alpha$  and  $\beta$  are the weights that determine the relative importance of the watermarking and the GAN loss terms (see Algorithm 1). In this way, training converges to a parameter vector  $\theta$ , in whose surrounding the watermarking loss term assumes (approximately constant) small values. By optimizing

---

**Algorithm 1:** Search for a wide flat minimum of the watermarking loss.

---

```

for iteration  $t = 1, 2, \dots$  do
    Store the initial generator parameters  $\theta_0$ 
    for  $i = 1, 2, \dots$  do
        Sample a noise vector  $\delta_i$ , s.t.  $-b < \delta_i < b$ ;
        Add  $\delta_i$  to the parameters of the generators:
         $\theta_i = \theta_0 + \delta_i$ ;
        Compute  $\mathcal{L}_w(\theta_i) = \text{BCE}(w_o, D_w(G_{\theta_i}(x)))$ ;
        Compute the gradient vector  $\nabla \mathcal{L}_w(\theta_i)$ ;
    end
    Compute  $\mathcal{L}_G(\theta_0)$  by Eq. (8);
    Compute the gradient vector  $\nabla \mathcal{L}_G(\theta_0)$ ;
    Updat  $\theta_0$  as
     $\theta = \theta_0 - \frac{\alpha}{k} \sum_{i=1}^k \nabla \mathcal{L}_w(\theta_i) - \beta \nabla \mathcal{L}_G(\theta_0)$ ;
end

```

---

the GAN with algorithm 1, the generator is encouraged to converge to a wide flat minimum of the watermarking loss,

<sup>2</sup>A similar architecture was used in [4], here we modify it to achieve both image-level and model-level robustness.

with a radius approximately equal to  $b$ . If the watermarked generator is modified by pruning, quantization, or even fine-tuning, it maintains an acceptable watermark accuracy as long as the change in the weight vector is less than  $b$ .

## V. IMPLEMENTATION DETAILS AND EXPERIMENTAL SETTING

TABLE III: Summary of task and datasets

Tasks	Architecture	Dataset	Num. of Images
Image Generation	StyleGAN2	FFHQ	70,000
Style transfer	CycleGAN	Photo2Monet	8,231
Image Synthesis	SPADE	ADE20K	22,210
Image Editing	StarGAN v2	CelebA-HQ	30,000
Super Resolution	CFSRCNN	VOC2012	1,7125
Denosing	MPRNet	SIDD	30,000

### A. Models and Datasets

The proposed GAN watermarking is both model- and task-agnostic, thus, we evaluated its performance on 6 different tasks and architectures.

a) *Image Generation*: This task aims at generating fully synthetic images starting from a random noise input. We used StyleGAN2 [28] as target model, and FFHQ [22] as training dataset to generate facial images.

b) *Style transfer*: This task aims at transferring the style of the source image to a target style. We used CycleGAN [29] as target model, and Photo2Monet as training dataset. CycleGAN is trained to translate natural photos into Monet-style paintings.

c) *Image Synthesis*: This task aims at synthesizing realistic images using semantic layouts. We used SPADE [30] as target model and ADE20K [31] as training dataset.

d) *Image Editing*: This task aims at editing the attributes of a given image, for example, changing the hair color or gender of a human face. We used StarGAN v2 [32] as target model, and CelebA-HQ [33] as training dataset. The model is trained to edit the face attributes of the input images.

e) *Super Resolution*: This task aims at upsampling low-resolution images to high-resolution images. We used CFSRCNN [34], a fully convolutional network as target model, and VOC2012 [35] as training dataset. CFSRCNN is trained to scale up images by a factor of 4.

f) *Denosing*: This task aims at removing noise from the input images. We used MPRNet [36], a U-Net based network as target model, and SIDD [37] as training dataset.

Table III summarizes the 6 different tasks evaluated in this paper and provides information about the corresponding datasets. We chose these different models to prove the generality of the proposed method. Please note that the general description given in Section IV takes a GAN architecture as baseline, however, our approach also works with other kinds of networks. For some tasks, in addition to the native adversarial loss, there are other losses involved, such as the cycle loss and identity loss in CycleGAN [29]. For simplicity, we omitted these losses in Section IV, since this omission does not affect the watermark embedding part of the process. In the sequel,

for simplicity, we indicate the models trained for the various tasks by referring to the architecture.

### B. Evaluation Metrics

To evaluate the effectiveness of the proposed approach, we used the following metrics:

a) *Bitwise Accuracy (Bit Acc)*: Bit Acc, also indicated by  $p_b$ , measures the fraction of correctly predicted bits. A high bitwise accuracy is preferable. The minimum  $p_b$  required for accurate verification can be set by using the analysis in Section III-C and setting desired values of  $P_f$  and  $P_m$ .

b) *Fréchet Inception Distance (FID)*: FID [38] is a widely used metric to evaluate the quality of the images generated by a GAN. It measures the distance between the distribution of real samples and generated samples in the feature space. A low FID is preferable.

c) *Peak Signal-to-Noise Rate (PSNR)*: PSNR measures the pixel-wise difference between the generated and target images produced by the super-resolution and denoising tasks. A high PSNR is desirable.

### C. Implementation Details

All experiments have been carried out by using PyTorch 1.9 on Centos 7.0, with RTX 3090 GPU. The models used for the comparisons have been trained by using the official code, and the optimizer, learning rate, batch size, and number of training epochs are the same. All models have been initially trained without the watermarking loss term until convergence and then fine-tuned with both the original loss and the watermarking loss to embed the watermark. For the noise layer, we randomly applied addition of Gaussian noise with standard deviation in [0.0001, 0.10], Gaussian blurring with kernel sizes in [0, 5] and standard deviation in [1, 7], differentiable JPEG compression as defined in [39], and color transformations including brightness, contrast, and saturation adjustment in the range [0.0, 0.05]. Each processing operator is applied with probability 0.15. The watermark decoder is frozen during embedding. The watermark consists of 100 bits, the weights  $\lambda_1$  and  $\lambda_2$  used to train the watermark encoder and decoder are both 1.0. In the embedding stage, the number of random noise perturbations  $k$  is 4, and the weight of the watermark loss  $\gamma$  is 2.0. As to  $b$ , we used 0.03 for all models but StyleGAN2 for which we let  $b = 1.0$ . Eventually, we let  $\alpha = \frac{1}{4}$  and  $\beta = 1.0$ .

## VI. EXPERIMENTAL RESULTS

### A. Effectiveness

In Table IV, we report the bitwise accuracy of WFM watermarking on four different image generation tasks (see Table III). The first column indicates the watermarked model, while the second indicates the watermarking algorithm. Specifically, we compared the performance of our method with those described in [6] and [4]. Both are multi-bit box-free GAN watermarking methods and [4] can be considered as the baseline of the proposed method.

We can observe that the proposed method achieves nearly perfect accuracy, except for the SPADE model. For image

generation models such as StyleGAN2 and StarGAN v2, we have achieved 0.999 Bit Acc ( $p_b$ ). Although [6] is also effective on multiple tasks, our method has a significant advantage in terms of Bit Acc. Meanwhile, our method also slightly outperforms [4] on 3 out of 4 tasks. Table IV also reports the missed detection probability at  $P_f = 10^{-3}$  estimated by using the analysis described in Section III-C. In all cases, the missed detection probabilities  $P_m$  are very close to zero.

TABLE IV: Bit Acc ( $p_b$ ) of different GAN watermarking methods over different GANs and corresponding  $P_m$  (computed as in Eq. (5) ) when the detection threshold is set in such a way to get  $P_f = 10^{-3}$ .

Models	Methods	Bit Acc ( $p_b$ )	$P_m$
StyleGAN2	Yu <i>et al.</i> [6]	0.854	$5.9 \times 10^{-7}$
	Fei <i>et al.</i> [4]	0.999	$1.5 \times 10^{-75}$
	Ours	0.999	$1.4 \times 10^{-102}$
CycleGAN	Yu <i>et al.</i> [6]	0.958	$5.1 \times 10^{-20}$
	Fei <i>et al.</i> [4]	0.996	$9.4 \times 10^{-51}$
	Ours	0.998	$6.5 \times 10^{-58}$
SPADE	Yu <i>et al.</i> [6]	0.979	$9.2 \times 10^{-29}$
	Fei <i>et al.</i> [4]	0.989	$1.8 \times 10^{-37}$
	Ours	0.968	$1.7 \times 10^{-23}$
StarGANv2	Yu <i>et al.</i> [6]	0.941	$8.9 \times 10^{-17}$
	Fei <i>et al.</i> [4]	0.999	$3.2 \times 10^{-80}$
	Ours	0.999	$1.4 \times 10^{-102}$

### B. Fidelity

Fidelity is a critical requirement for an effective watermarking method since it measures the impact that the presence of the watermark has on the performance of the model on its primary task. For image generation tasks, fidelity is measured by the perceptual quality of the generated images.

Table V shows the FID of the baseline model, i.e., the non-watermarked model, and the FID of the model watermarked by different methods. Note that the estimation of FID is influenced by the number of images, and an insufficient number of images can lead to inaccurate FID. Typically, the recommended number of images for FID calculation is 50,000. However, the number of images in the validation datasets for CycleGAN and SPADE is much less than 50,000, so we report the FID of CycleGAN and SPADE for reference only. Still, we can observe that, over four different tasks, our method has a very slight effect on FID with almost no deterioration in image quality. For StyleGAN2 and StarGAN v2, our method only increases the FID by 0.08 and 0.10, which is acceptable. For CycleGAN and SPADE, the initial FID is rather high, however, our method only increases it to a value 2 / 3 points larger. The method in [6] has a much stronger impact on quality of the images generated by the GAN.

### C. Robustness to model-level attacks

We evaluated the robustness of our methods against pruning, quantization, noise addition, and fine-tuning. For pruning, we set the weights with the smallest absolute values to zero. For

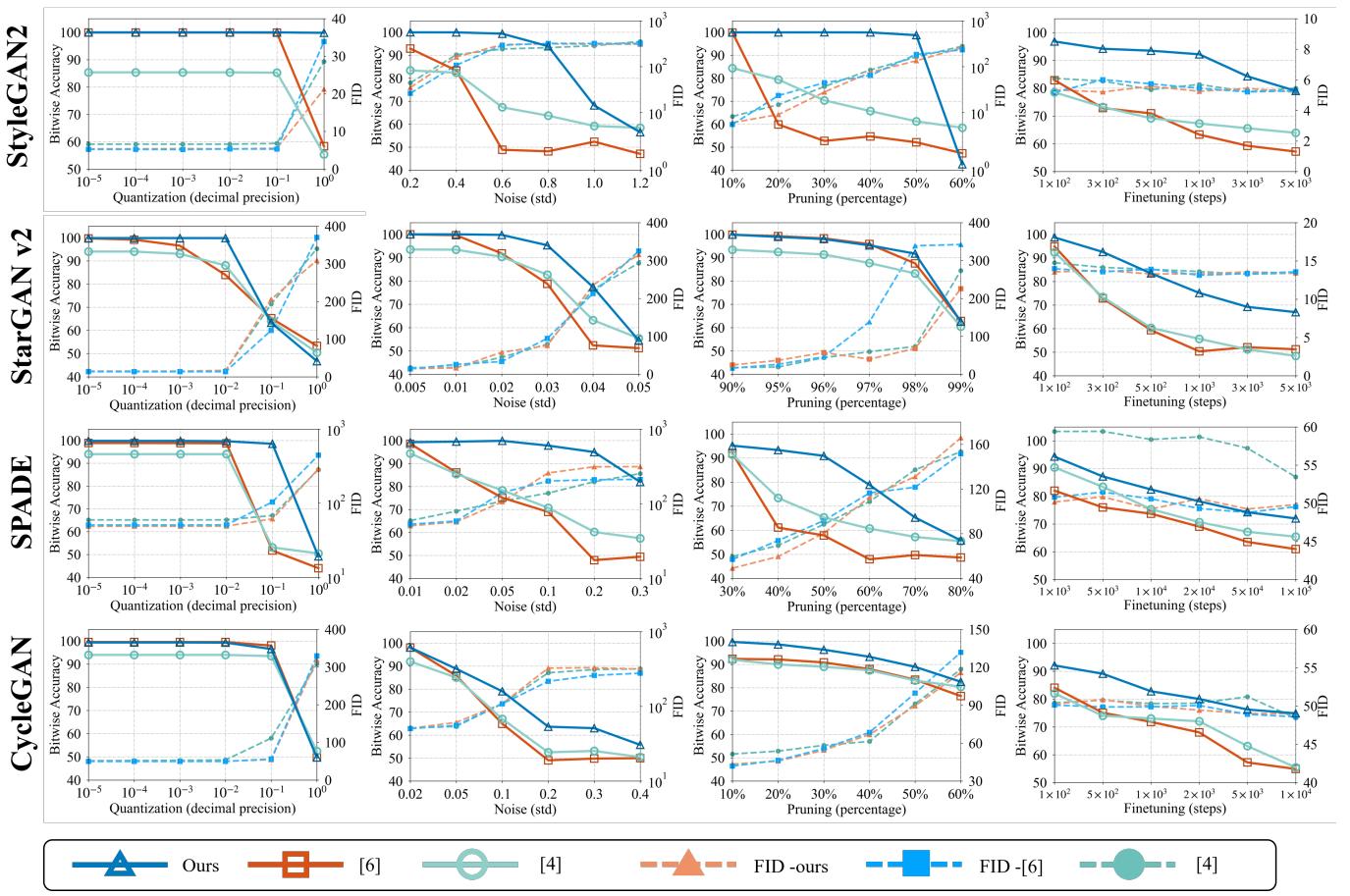


Fig. 4: Robustness to different model-level attacks.

TABLE V: FID of different models before and after watermark embedding with different methods.

Method Models	StyleGAN2	CycleGAN	SPADE	StarGAN v2
Baseline	5.28	47.68	46.80	13.81
Yu <i>et al.</i> [6]	6.61	50.34	59.42	14.89
Fei <i>et al.</i> [4]	5.16	49.39	50.70	13.53
Ours	5.36	49.52	50.59	13.91

quantization, we represented the model weights with a given decimal precision. For noise addition, we added Gaussian noise to the model weights. For fine-tuning, we fine-tuned the generator with the original training set, but without using the watermarking loss term.

Fig. 4 reports (left y-axis of the plots) the Bit Acc  $p_b$  obtained after the models are modified by quantization, noise addition, pruning and fine-tuning. The FID obtained with the attacked models is also reported on the right y-axis<sup>3</sup>. Compared to both [4] and [6], our solution significantly improves the robustness of the watermark against all model-level attacks. The WFM has the most beneficial effect on StyleGAN2 (image generation). The Bit Acc ( $p_b$ ) remains almost 1 even though the model weights are compressed by decimal precision  $10^0$ ,

<sup>3</sup>For sake of visualization, a different x-axis range is used in the various cases for the same attack.

meaning that the model weights are represented by integer numbers. In [4] the Bit Acc  $p_b$  drops to about 0.50 after the addition of Gaussian noise with standard deviation (std) 0.6, or 50% of the parameters are pruned, while our solution can effectively resist this noise. In the fine-tuning attack, our method is able to maintain a  $p_b$  over 0.80 after about 5 epochs (5,000 steps) fine-tuning on FFHQ [22], while it takes only about 100 steps to remove the watermark embedded as in [4] and [6].

In the case of StarGAN v2, although the benefit against quantization, noise addition and pruning attacks is not readily apparent, robustness to fine-tuning is much improved with our method and the Bit Acc  $p_b$  is at least 0.20 higher w.r.t. [4] and [6]. For SPADE, the Bit Acc  $p_b$  of our method is approximately at least 0.20 to 0.30 better than [4] before the model is completely disabled by noise addition and pruning attacks.

In summary, our method yields a higher Bit Acc than [4] and [6] under model-level attacks with the same strength, demonstrating the effectiveness of our method. It is worth noting that the adversary may be willing to remove the watermark at the cost of degrading the image quality to a certain extent. For example, by pruning 20% of the weights of StyleGAN2, the FID is increased to about 10, and the Bit Acc of [4] is drastically reduced to 0.60, the Bit Acc of [6] is decreased to 0.80, while the Bit Acc of our method remains

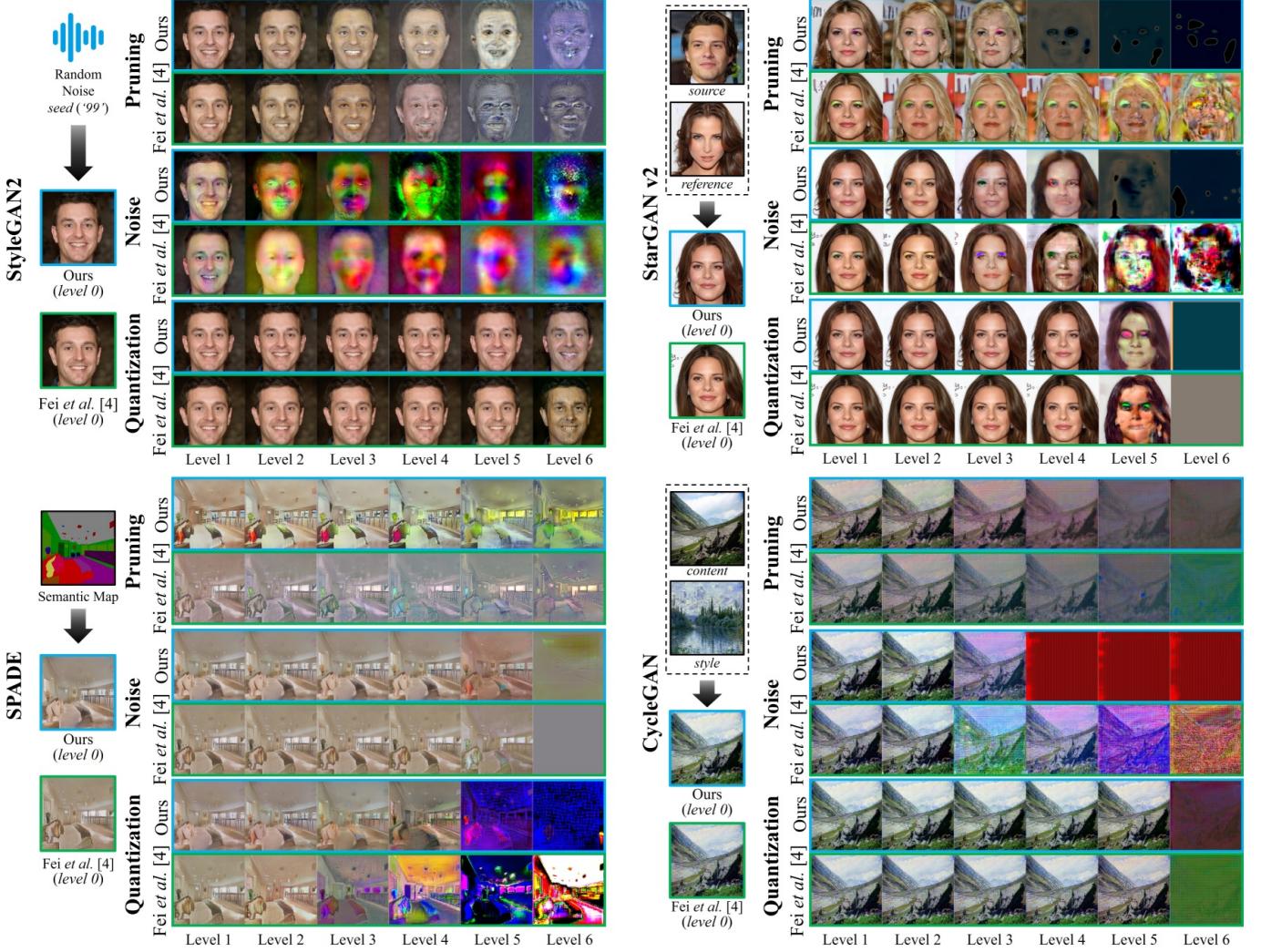


Fig. 5: Samples generated by watermarked models after the attacks (quantization, noise, pruning) applied with various strengths.

unchanged. In fact, to remove the watermark embedded using our method, the attacker has to modify the network down to a point of making it unusable, e.g., the FID must be raised to 300. Note that according to the analysis in Sect. III-C  $p_b = 0.8$  is enough to achieve ownership verification with  $P_f$  and  $P_m$  in the order of  $10^{-4}$  to  $10^{-3}$ .

Fig. 5 shows some images generated by the attacked models under different kinds of attacks and intensities. For simplicity, we denote the strength by levels 1 to 6, which correspond to the ticks on the x-axis in Fig. 4. We can observe that in many cases the watermark accuracy is sufficient to achieve ownership authentication despite severe image degradation (e.g., in Level 3 and 4). Therefore, removing the watermark goes at the cost of poor quality of the generated images, making the models virtually useless.

#### D. Robustness to surrogate model attacks

In the previous section, we considered attacks carried out in white-box conditions, where the model can be directly accessed by the attacker. In this section, we consider a scenario where the attacker can only query the model and apply a

surrogate model attack. We carried out surrogate model attacks on the watermarked CycleGAN with the same and different architectures of the attacked model. The two surrogate models are based on ResNet (the same as the attacked model) and U-Net (different from the attacked model) architectures. Both surrogate models are trained for 200 epochs by using input and output pairs generated by the watermarked CycleGAN generator. We also evaluated the effect of training the surrogate models with different loss functions. Specifically, we optimized the surrogate models by using: i) MSE loss  $\mathcal{L}_2$  only, and ii) MSE loss  $\mathcal{L}_2$  and adversarial loss  $\mathcal{L}_{adv}$ . In the latter case, we used the discriminator of the original CycleGAN and set the weight of the adversarial loss to 0.001. The results we got are shown in Table VI.

We can observe that our method achieves nearly 0.80 Bit Acc when the surrogate model adopts the same architecture of the attacked model and is trained by using an  $\mathcal{L}_2$  loss, which is  $0.06 \sim 0.07$  higher than the  $p_b$  obtained by the other methods. This confirms that the watermark can be transferred to the new network. We also observe that the architecture used to build the surrogate model has only a slight influence on

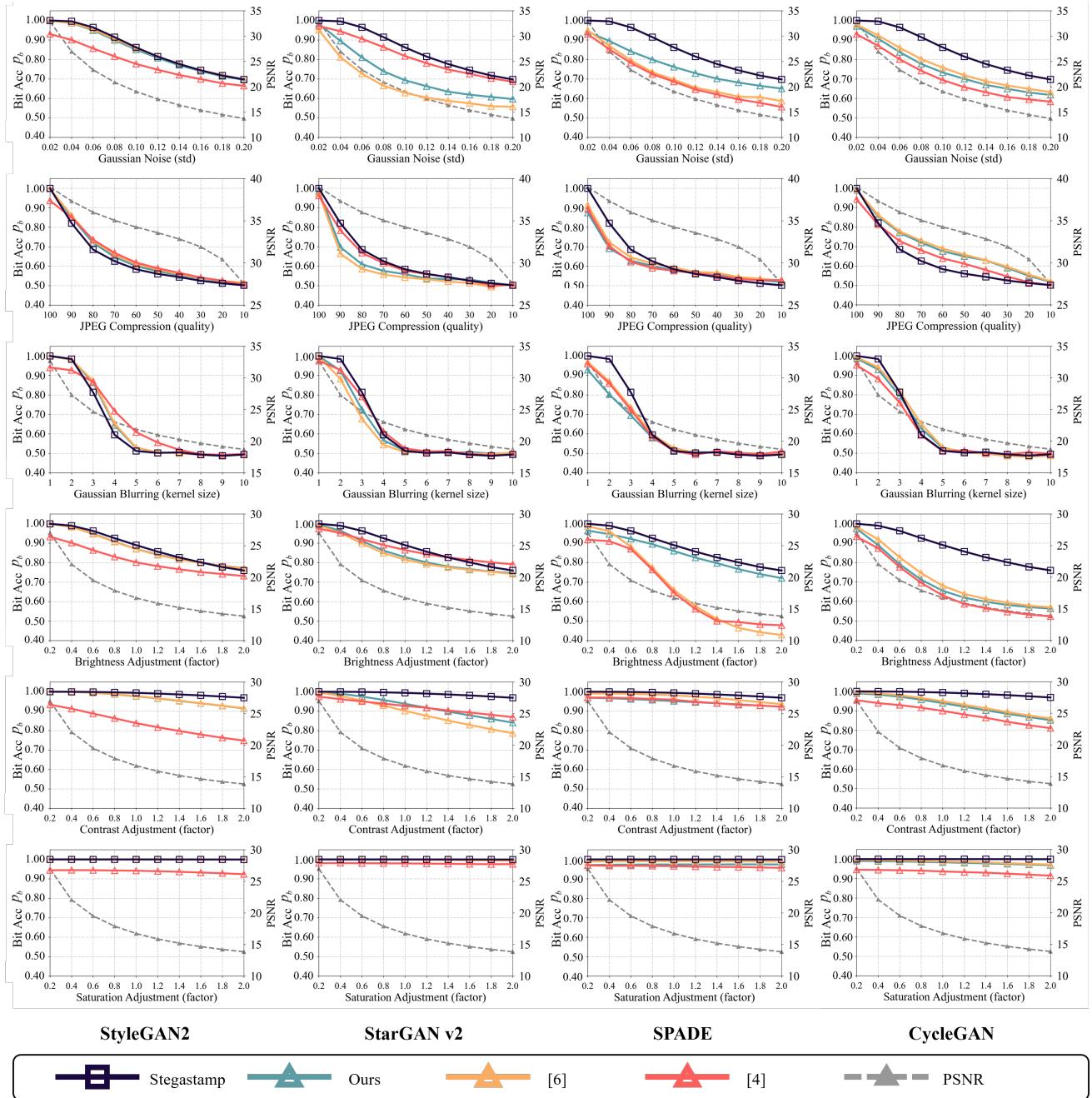


Fig. 6: Robustness to different image-level attacks. The gray dashed lines denote the PSNR between the original image and the processed image.

TABLE VI: Bit Acc  $p_b$  after surrogate model attacks with different losses and network architectures.

Method	Loss	ResNet-based	UNet-based
Fei <i>et al.</i> [4]	$\mathcal{L}_2$	0.727	0.704
	$\mathcal{L}_2 + \mathcal{L}_{adv}$	0.662	0.669
Yu <i>et al.</i> [6]	$\mathcal{L}_2$	0.735	0.736
	$\mathcal{L}_2 + \mathcal{L}_{adv}$	0.686	0.660
Ours	$\mathcal{L}_2$	0.793	0.786
	$\mathcal{L}_2 + \mathcal{L}_{adv}$	0.722	0.725

the watermark accuracy. Even when we used a completely different architecture, there is no more than a 0.01 drop in  $p_b$  for our method. When the surrogate model is trained by using both  $\mathcal{L}_2$  and  $\mathcal{L}_{adv}$  losses, the watermark accuracy decreases by  $0.06 \sim 0.07$ . However, watermark accuracies of both surrogate models are still above 0.70, noticeably higher than the accuracies obtained with the other methods. Moreover, as we have proven in Section III-C, when the Bit Acc  $p_b$  is larger than 0.70 both  $P_f$  and  $P_m$  are less than  $10^{-2}$ .

### E. Robustness to image processing manipulations

We also evaluated the robustness of our method against the image processing operations included in the noise layer used during training, see Fig. 6. The figure also reports the results obtained by [4] and [6], which have been trained by using the same noise layers. The left y-axis indicates the watermark extraction accuracy on the processed images, while the right y-axis indicates the PSNR between the original images and the processed images. In addition to the results of different GAN watermarking methods, we also plotted the results of Stegastamp [21] as baseline, namely the image watermarking network on which the watermark decoder considered here and in [4] are based. Since the generator is not trained together with the watermark decoder, the robustness of all the GAN watermarking methods are expected to be worse than this baseline.

In general, with the increase of the image processing intensity, the Bit Acc of all methods decreases, while the PSNR also drops significantly. We can observe that in most cases, the original Stegastamp achieves the best robustness, being this the upper limit of GAN watermarking methods. All methods demonstrate promising robustness against color adjustment, while JPEG compression and Gaussian blurring are more challenging, and the Bit Acc decreases as the intensity of the attack increases. Nonetheless, watermark removal requires a compromise in terms of image quality (PSNR). In Fig. 7, we show some samples of StyleGAN2 images after the application of processing with the intensity such that the Bit Acc  $p_b$  remains above 0.70 (corresponding to a  $P_f$  and  $P_m$  lower than  $10^{-2}$  for the ownership verification). We can argue that all attacks need to be very strong to reduce the Bit Acc to a value that makes the ownership verification fail, except for JPEG compression, which leaves only some artifacts in the image.

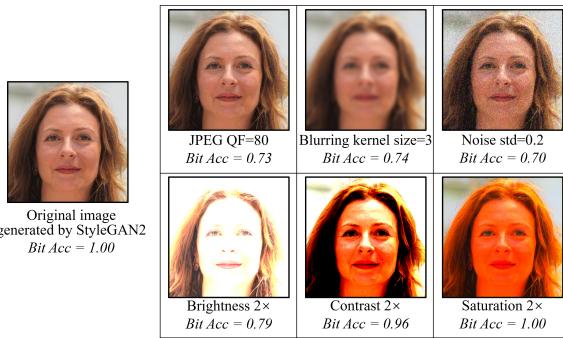


Fig. 7: Samples processed by various operations and Bit Acc.

### F. Effectiveness on image processing models

In the experiments reported in the previous sections, all watermarked models are based on GANs, although for different image generation or synthesis tasks. To demonstrate that our method can be applied to completely different generative architectures, in this section, we present the results we got by watermarking 2 CNN-based image processing networks, namely MPRNet [36] and CFSRCNN [34]. Note that both

models are optimized by using pixel-wise MSE, without adversarial loss.

We report the results of Bit Acc and PSNR in Table VII. We can observe that both our method and [4] achieve nearly perfect accuracy. In contrast, for both MPRNet and CFSRCNN, the method in [6] cannot converge. In terms of fidelity, the impact of our method on the original task is within an acceptable range. In fact, our watermarking method reduces the PSNR by no more than 0.5dB on MPRNet and about 2dB on CFSRCNN. Compared to [4], we achieve a better robustness with a very low impact on the performance of the original task.

Fig. 8 shows the robustness against model-level attacks for MPRNet and CFSRCNN. Our method shows the same strong robustness for both networks. When 60% of the network parameters are set to zero,  $p_b$  for MPRNet still remains nearly 1.00, and over 0.80 for CFSRCNN, while it drops to 0.60 for [4]. For the most challenging fine-tuning attack, our method is able to maintain an accuracy of 0.70, while [4] is reduced to about 0.50. Overall, we observe that in the case of CNN-based image processing networks, the watermark is less robust than for GANs. For instance, compared to the results in Fig. 4, the reduction in Bit Acc is more significant under Gaussian noise addition of the same intensity. However, a gain in robustness is always achieved with our method with respect to [6] and [4] also in this case of image processing networks. This gain is a significant one in many cases, especially against pruning and fine-tuning attacks.

TABLE VII: Bit Acc ( $p_b$ ),  $P_m$  (when  $P_f = 10^{-3}$ ), and PSNR for image processing models.

Task	Model	Bit Acc	$P_m$	PSNR
Denosing	MPRNet -baseline	-	-	39.70
	MPRNet -[6]	0.565	$\approx 0.90$	39.64
	MPRNet -[4]	0.999	$\approx 1.2 \times 10^{-96}$	39.39
	MPRNet -ours	0.999	$\approx 1.3 \times 10^{-102}$	39.24
SR	CFSRCNN -baseline	-	-	25.92
	CFSRCNN -[6]	0.511	$\approx 0.99$	25.92
	CFSRCNN -[4]	1.000	$\approx 0$	23.86
	CFSRCNN -ours	1.000	$\approx 0$	23.61

### G. Ablation study

In this section, we study the effect of the number  $k$  of samples used within each training step to converge to a wide flat minimum and the noise range  $b$  (see Algorithm 1) on the fidelity of the model, and on the accuracy and robustness of the watermark. Intuitively speaking, a larger number of  $k$  could potentially explore more directions in the parameters space, and force the model to a flatter minimum, while a larger noise range could broaden the scope of the minimum. To verify this, we took StyleGAN2 as a case study and watermarked it using different values of  $k$  and  $b$ . The resulting watermark Bit Acc and FID are shown in Table VIII. We can observe that, for a given  $b$ , as  $k$  increases, the Bit Acc remains stable and does not increase further. However, when  $k$  is fixed and  $b$  increases, the Bit Acc tends to reduce, and

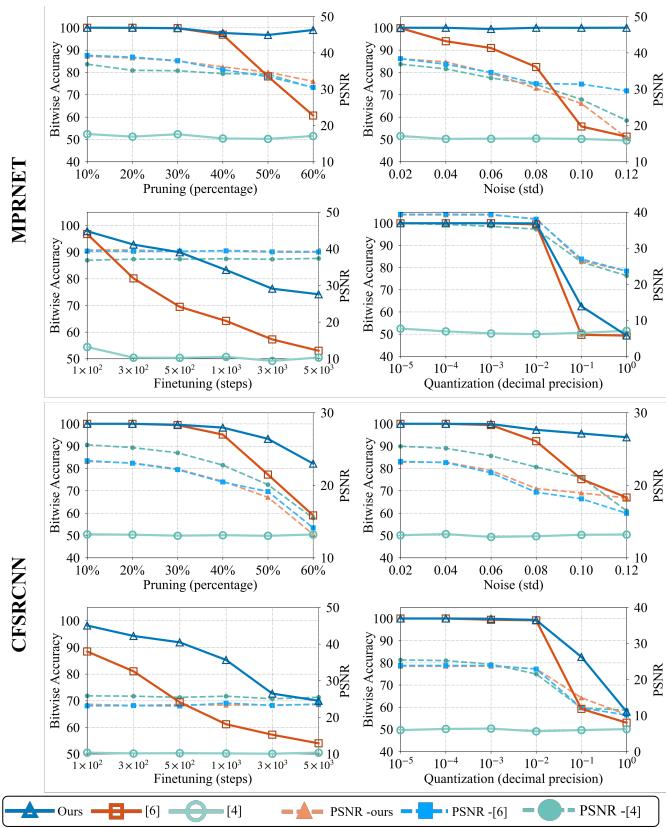


Fig. 8: Robustness of CNN-based image processing networks against model-level attacks.

the FID shows a clear upward trend. These results suggest that the parameter space is sufficiently high-dimensional and has sufficient redundancy, allowing to converge to a minimum that can accommodate a moderate perturbation. Nonetheless, as the perturbation magnitude  $b$  increases, converging to such a minimum while ensuring the quality of the generated images is highly challenging, and such a minimum may not even exist.

TABLE VIII: Bit Acc  $p_b$  and FID under different noise sampling iterations  $k$  and noise ranges  $b$  for StyleGAN2.

$\diagdown$	$b$	0.1	0.3	1.0	3.0	5.0	10.0
$k \backslash$							
1	0.999 5.48	0.997 5.37	0.999 5.38	0.973 6.17	0.965 7.04	0.944 9.13	
4	0.999 5.28	0.999 5.48	0.999 5.36	0.971 5.51	0.957 8.10	0.945 11.60	
8	0.996 5.41	0.996 5.91	0.993 5.67	0.963 6.51	0.957 7.69	0.948 12.10	
16	0.998 5.51	0.996 5.87	0.996 6.17	0.965 7.09	0.950 8.85	0.944 9.85	

We also evaluated the influence of varying  $b$  and  $k$  on the robustness of the watermark against fine-tuning attacks (see Table IX). We can observe that when  $k$  is greater than 4, further increments in  $k$  do not significantly improve the robustness. However, as  $b$  increases, the robustness initially strengthens, followed by a sharp decline, and the optimal robustness is reached roughly when  $b$  equals 1.0. We hypothesize

TABLE IX: Bit Acc  $p_b$  after fine-tuning attack under different noise sampling times  $k$  and noise ranges  $b$  for StyleGAN2.

$\diagdown$	$b$	0.1	0.3	1.0	3.0	5.0	10.0
$k \backslash$							
1	0.627	0.699	0.722	0.644	0.591	0.587	
4	0.646	0.733	0.804	0.724	0.615	0.581	
8	0.649	0.745	0.810	0.707	0.626	0.636	
16	0.649	0.748	0.814	0.732	0.638	0.615	

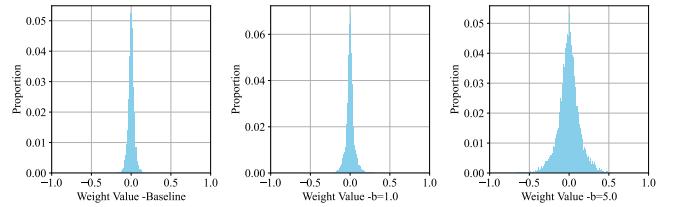


Fig. 9: Parameter distribution of watermarked StyleGAN2 for two different values of  $b$ . Baseline stands for the original (non watermarked) model.

that this is due to the fact that i) an overly large  $b$  results in lower Bit Acc before the attack, and ii) an overly large  $b$  causes a larger change in the distribution of the model weights compared to the original non-watermarked model (see Fig. 9). This leads to larger gradients ( $\partial L_G / \partial \theta$ ), causing more drastic changes of weights during the fine-tuning attack and a faster escape from the wide flat minimum.

The above results can provide valuable guidance for choosing appropriate  $k$  and  $b$ . On one hand, a rather small  $k$  is enough to achieve a good balance between fidelity, Bit Acc, and robustness, while also reducing computational cost of the embedding procedure. On the other hand, we believe that the key to achieving improved robustness is to choose a suitable  $b$  for the wide flat minimum search algorithm, making it sufficiently wide and inducing minimal changes to the distribution of model weights.

## VII. CONCLUSION

With the increasing commercial value of image generation models, IPR protection has become an essential problem. In this paper, we have proposed a robust box-free watermarking method, based on WFM search, to protect the IPR of GANs and image processing networks. Extensive experiments demonstrate that our method achieves significant robustness in different white-box model-level attacks and image processing attacks.

Our work indicates that the robustness of existing box-free GAN watermarking methods has flaws, as the watermark can be easily removed through fine-tuning attacks. We attribute this to the unique nature of box-free methods, which essentially work by fitting the distribution of watermarked data. Thus using clean data for fine-tuning attacks can shift the generator's distribution, resulting in the removal of the watermark. Essentially, the key to improving the robustness of model-level attacks lies in improving the robustness of watermarking loss to parameter perturbations of the generator. In our method, we

do so by treating watermark embedding and generator training as two distinct tasks, and looking for a wide flat minimum of the embedding loss to prevent the model from forgetting the watermark during fine-tuning attacks. Our experiments also indicate that the key to achieving optimal robustness is that the generator converges to a minimum as wide as possible while the parameter distribution of the watermarked generator is close to that of the non-watermarked generator.

An interesting direction for future research is to extend our method to more advanced generative models, such as diffusion models (DM). DMs are gradually replacing GANs in an increasing amount of practical applications. Hence, protecting the IPR of DMs is becoming an important issue. The training and image generation process of DMs are different from GANs, thus calling for specific designs.

## REFERENCES

- [1] M. Barni and F. Bartolini, *Watermarking systems engineering: enabling digital assets security and other applications*. CRC Press, 2021.
- [2] Y. Li, H. Wang, and M. Barni, “A survey of deep neural network watermarking techniques,” *Neurocomputing*, vol. 461, pp. 171–193, 2021.
- [3] D. S. Ong, C. S. Chan, K. W. Ng, L. Fan, and Q. Yang, “Protecting intellectual property of generative adversarial networks from ambiguity attacks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3630–3639.
- [4] J. Fei, Z. Xia, B. Tondi, and M. Barni, “Supervised gan watermarking for intellectual property protection,” in *Proc. Int. Workshop Digit. Watermarking*. IEEE, 2022, pp. 1–6.
- [5] T. Qiao, Y. Ma, N. Zheng, H. Wu, Y. Chen, M. Xu, and X. Luo, “A novel model watermarking for protecting generative adversarial network,” *Comput. Security*, p. 103102, 2023.
- [6] N. Yu, V. Skripniuk, S. Abdelnabi, and M. Fritz, “Artificial fingerprinting for generative models: Rooting deepfake attribution in training data,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 14448–14457.
- [7] N. Yu, V. Skripniuk, D. Chen, L. S. Davis, and M. Fritz, “Responsible disclosure of generative models using scalable fingerprinting,” in *Proc. Int. Conf. Learn. Represent.*, 2022.
- [8] H. Ruan, H. Song, B. Liu, Y. Cheng, and Q. Liu, “Intellectual property protection for deep semantic segmentation models,” *Front. Comput. Sci.*, vol. 17, no. 1, p. 171306, 2023.
- [9] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, “Embedding watermarks into deep neural networks,” in *Proc. ACM Int. Conf. Multimed. Retr.*, 2017, pp. 269–277.
- [10] Y. Li, B. Tondi, and M. Barni, “Spread-transform dither modulation watermarking of deep neural network,” *J. Inf. Secur. Appl.*, vol. 63, p. 103004, 2021.
- [11] E. Le Merrer, P. Perez, and G. Trédan, “Adversarial frontier stitching for remote neural network watermarking,” *Neural. Comput. Appl.*, vol. 32, no. 13, pp. 9233–9244, 2020.
- [12] B. Darvish Rouhani, H. Chen, and F. Koushanfar, “Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks,” in *Proc. Int. Conf. Archit. Support Program.*, 2019, pp. 485–497.
- [13] Y. Li, L. Abady, H. Wang, and M. Barni, “A feature-map-based large-payload dnn watermarking algorithm,” in *Proc. Digit. Forensics Watermarking 20th Int. Workshop*. Springer, 2021, pp. 135–148.
- [14] L. Fan, K. W. Ng, and C. S. Chan, “Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks,” in *Proc. Neural Inf. Process. Syst.*, vol. 32, 2019.
- [15] L. Fan, K. W. Ng, C. S. Chan, and Q. Yang, “Deepipr: Deep neural network ownership verification with passports,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6122–6139, 2021.
- [16] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, “Protecting intellectual property of deep neural networks with watermarking,” in *Proc. Asia Conf. Comput. Commun. Secur.*, 2018, pp. 159–172.
- [17] M. Li, Q. Zhong, L. Y. Zhang, Y. Du, J. Zhang, and Y. Xiang, “Protecting the intellectual property of deep neural networks with watermarking: The frequency domain approach,” in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun.* IEEE, 2020, pp. 402–409.
- [18] Q. Zhong, L. Y. Zhang, J. Zhang, L. Gao, and Y. Xiang, “Protecting ip of deep neural networks with watermarking: A new label helps,” in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining*. Springer, 2020, pp. 462–474.
- [19] H. Chen, B. D. Rouhani, and F. Koushanfar, “Blackmarks: Black-box multibit watermarking for deep neural networks,” *arXiv preprint arXiv:1904.00344*, 2019.
- [20] Z. Li, C. Hu, Y. Zhang, and S. Guo, “How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of dnn,” in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2019, pp. 126–137.
- [21] M. Tancik, B. Mildenhall, and R. Ng, “Stegastamp: Invisible hyperlinks in physical photographs,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2117–2126.
- [22] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4401–4410.
- [23] H. Wu, G. Liu, Y. Yao, and X. Zhang, “Watermarking neural networks with watermarked images,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 7, pp. 2591–2601, 2020.
- [24] S. Hochreiter and J. Schmidhuber, “Flat minima,” *Neural Comput.*, vol. 9, no. 1, pp. 1–42, 1997.
- [25] D. Stutz, M. Hein, and B. Schiele, “Relating adversarially robust generalization to flat minima,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7807–7817.
- [26] J. Cha, S. Chun, K. Lee, H. C. Cho, S. Park, Y. Lee, and S. Park, “Swad: Domain generalization by seeking flat minima,” in *Proc. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 22405–22418.
- [27] G. Shi, J. Chen, W. Zhang, L.-M. Zhan, and X.-M. Wu, “Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima,” in *Proc. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 6747–6761.
- [28] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8110–8119.
- [29] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2017, pp. 2223–2232.
- [30] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, “Semantic image synthesis with spatially-adaptive normalization,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2337–2346.
- [31] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 633–641.
- [32] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, “Stargan v2: Diverse image synthesis for multiple domains,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8188–8197.
- [33] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *arXiv preprint arXiv:1710.10196*, 2017.
- [34] C. Tian, Y. Xu, W. Zuo, B. Zhang, L. Fei, and C.-W. Lin, “Coarse-to-fine cnn for image super-resolution,” *IEEE Trans. Multimed.*, vol. 23, pp. 1489–1502, 2020.
- [35] M. Everingham, S. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, 2015.
- [36] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, “Multi-stage progressive image restoration,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 14821–14831.
- [37] A. Abdelhamed, S. Lin, and M. S. Brown, “A high-quality denoising dataset for smartphone cameras,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1692–1700.
- [38] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *Proc. Neural Inf. Process. Syst.*, vol. 30, 2017.
- [39] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, “Hidden: Hiding data with deep networks,” in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 657–672.