

Robust Retraining-free GAN Fingerprinting via Personalized Normalization

Jianwei Fei
Jinan University
Guangzhou, China

Zihua Xia [†]
Jinan University
Guangzhou, China

Benedetta Tondi
University of Siena
Siena, Italy

Mauro Barni
University of Siena
Siena, Italy

Abstract—In recent years, there has been significant growth in the commercial applications of generative models, licensed and distributed by model developers to users, who in turn use them to offer services. In this scenario, there is a need to track and identify the responsible user in the presence of a violation of the license agreement or any kind of malicious usage. Although there are methods enabling Generative Adversarial Networks (GANs) to include invisible watermarks in the images they produce, generating a model with a different watermark, referred to as a fingerprint, for each user is time- and resource-consuming due to the need to retrain the model to include the desired fingerprint. In this paper, we propose a retraining-free GAN fingerprinting method that allows model developers to easily generate model copies with the same functionality but different fingerprints. The generator is modified by inserting additional Personalized Normalization (PN) layers whose parameters (scaling and bias) are generated by two dedicated shallow networks (ParamGen Nets) taking the fingerprint as input. A watermark decoder is trained simultaneously to extract the fingerprint from the generated images. The proposed method can embed different fingerprints inside the GAN by just changing the input of the ParamGen Nets and performing a feedforward pass, without finetuning or retraining. The performance of the proposed method in terms of robustness against both model-level and image-level attacks is also superior to the state-of-the-art.

Index Terms—IPR Protection, DNN watermarking, GAN fingerprinting, Box-free Watermarking, Security of Deep Learning

I. INTRODUCTION

Synthetic image generation has made significant progress in recent years and generative models are now widely used in commercial applications. These models are provided to commercial users as production tools or for selling services. Protecting the Intellectual Property Rights (IPR) of model owners has become a pressing issue to avoid potential copyright infringements, such as unauthorized duplication or model theft, when these models are delivered to malicious users. Deep Neural Network (DNN) watermarking has been proposed as a solution to protect the IPR associated with DNN models [1]. Most DNN watermarking methods focus on the protection of discriminative models, namely, networks developed for classification tasks, and less attention is paid to generative models. Yet, some methods for the watermarking of generative models have started appearing recently. Given the large entropy of the output of generative models, the watermark can be directly extracted from the output produced by the model, thus permitting to verification of the watermark in a so-called *box-free* setting. In this way, it is possible to determine the source of images produced by generative models and associate any image to the generative model that produced it [2].

With the exception of a few scattered works [3], the existing approaches for the watermarking of generative models, notably GANs, are designed for ownership verification, aiming at making it possible to retrieve the model authorship information from the generated images. These methods embed a fixed watermark, linking the model

Zihua Xia is the corresponding author.

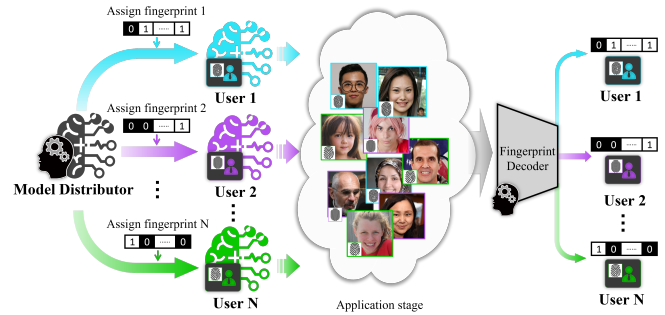


Fig. 1. The GAN fingerprinting scenario considered in this paper.

to the owner, and require retraining or finetuning if different a watermark has to be embedded in the model.

In this paper, we focus on a different scenario, hereafter referred to as GAN fingerprinting, illustrated in Fig. 1. In this scenario, the model distributor, simply referred to as the model owner, releases distinct watermarked model instances to different users, in such a way that the user-specific fingerprint can be recovered from the images produced by these models for copyright authentication and to trace back to the guilty user in case of a violation of the license agreements (traitor tracing). A problem with this scenario is that the model owner must produce several instances of the model each containing a different watermark, in this case referred to as a fingerprint, which in general requires training or finetuning a new model for each user. In this paper, we propose a retraining-free GAN fingerprinting method for box-free watermarking of GAN models, permitting to *easily* create different model instances each containing a different fingerprint, without any need to retrain or finetuning the watermarked model. This goal is achieved by introducing a new Personalized Normalization (PN) layer into the generator’s architecture, whose scaling factor and bias parameters are determined by two independent shallow networks, called the Parameters Generation Networks (ParamGen Nets), that are fed with the input fingerprint sequence. The generator and the ParamGen Nets are trained jointly with the watermark decoder, by varying the fingerprint sequence during training across the iterations. Once the model is trained, the model distributor can easily get GAN models with different fingerprints by just changing the fingerprint sequence at the input of the ParamGen Nets and performing a feedforward pass to modify the parameters of the PN layers of the GAN accordingly. Our idea of embedding the watermark information in the normalization layers is inspired by passport-based approaches [4] proposed for the protection of the IPR of DNN classifiers, which use passports, namely digital signatures, to unlock the normalization layer and use the network at inference time, achieving remarkable robustness against both removal and ambiguity attacks. Robustness is a crucial requirement for DNN watermarking

algorithms aimed at IPR protection [1]. Malicious users can conduct watermark removal attacks by model-level modifications, moreover, box-free GAN watermarking is also subject to image-level attacks. An additional strength of our method is that the watermark embedded in the GAN models is a very robust one. In particular, the experiments we run show that good robustness is achieved against both model-level attacks, specifically finetuning and model compression (pruning and quantization), and image post-processing operations, like JPEG compression, noise addition, and Gaussian blur.

To the best of our knowledge, the only work proposing a retraining-free, GAN fingerprinting method is [3], which uses the fingerprint sequence to modulate the parameters of the convolutional kernels. Compared to [3], our approach can achieve improved robustness against both image-level and model-level removal attacks. We believe that this gain comes from the different approach we are considering for watermark embedding, namely the PN-based embedding.

The rest of the paper is organized as follows. In Section II, we briefly discuss the state of the art of GAN watermarking. The proposed method is described in Section III. Section IV reports the experimental methodology, settings, and results. Finally, in Section V, we conclude the paper with some final remarks and hints for future research.

II. RELATED WORK

The goal of DNN watermarking is to embed watermarks into deep learning networks, that can be used for ownership verification, fingerprinting, and traitor tracing, among other possible applications, without impairing their functionality (unobtrusiveness) [1], [5]. While DNN watermarking has been mostly applied to CNN based classifiers, the watermarking of GAN has recently started receiving attention [2], [6]–[8]. Depending on the kind of access required for the watermark extraction, watermarking algorithms can be categorized as white-box, black-box, and box-free. White-box methods require access to the internal parameters of the model during verification. With black-box methods, instead, the watermark is extracted by looking at the output of the network in correspondence to a set of specific querying inputs. Finally, box-free watermarking methods do not require any kind of access to the suspicious model, and the watermark can be directly extracted from the output produced by the model. Box-free methods are only possible with generative models, for which case the entropy of the output is large enough. Black-box and box-free methods have been proposed for GAN watermarking. In [2], [9], [10], the authors utilize black-box watermarking to protect the IPR of generative models where the watermark is embedded by instructing the model to learn to produce specific output images when fed with certain inputs. Several methods have been provided performing box-free watermarking [6]–[8]. In particular, Wu *et al.* [6] imposed an additional constraint on the output of the generator in the loss, in such a way as to embed a specific watermark image into any generated image. This method is a zero-bit watermarking that utilizes the PSNR between the extracted image and the ground-truth for ownership verification. Yu *et al.* [7] discover that by training the GAN using data with a watermark embedded via StegaStamp [11], the generated images will also contain the watermark. Fei *et al.* [8] propose to improve the method in [7] by embedding the watermark in a supervised manner, using a pre-trained watermark decoder to guide the training and incorporating a loss term in the optimization to make sure that the watermark extracted from the generated images is close to the true watermark.

The above watermarking methods are developed with the ownership verification application in mind. In fingerprinting applications,

a company may want to distribute to different users model instances having the same functionality but with distinct (user-specific) fingerprints. In this scenario, the above box-free watermarking methods would require retraining a new generative model for every user, with enormous costs. To address this problem, Yu *et al.* [3] propose a method for the efficient fingerprinting of GAN models. This method enables the efficient generation of model instances with the same image generation functionality with different user-specific fingerprints, achieved by exploiting watermark autoencoders and modulating the parameters of the convolutional kernels based on the to-be-embedded watermark. To the best of our knowledge, [3] is the only one proposing a method for retraining-free GAN fingerprinting. As a drawback, this method has limited robustness against network modification and re-use (model-level attack) and against image post-processing (image-level attacks), which hinders its practical applicability.

In this paper, we propose a retraining-free GAN fingerprinting method with improved robustness against both model-level and image-level attacks. In particular, the robustness against finetuning is largely improved with respect to [3]. The cost in terms of time and resources is the same as in [3], since both methods do not need any retraining and a user-specific fingerprint can be embedded by feeding the ParamGen Nets with the fingerprint, namely the watermark message, and using the resulting parameters in the PN layer running a feedforward pass through the generation network to set the parameters of the generator instance associated to the user. Table I summarizes the capabilities of state-of-the-art GAN watermarking approaches in terms of capacity, efficiency, and robustness. The term efficiency here refers to the capability of changing the fingerprint, without retraining or finetuning the model. Our method is the only one with all the desired capabilities.

TABLE I
COMPARISONS OF STATE-OF-THE-ART BOX-FREE GAN WATERMARKING METHODS AND THE PROPOSED METHOD.

Approach	Capacity	Retraining-free	Robustness
Wu <i>et al.</i> [6]	Zero-bit	✗	✓
Yu <i>et al.</i> [7]	Multi-bit	✗	✗
Fei <i>et al.</i> [8]	Multi-bit	✗	✗
Yu <i>et al.</i> [3]	Multi-bit	✓	✗
Ours	Multi-bit	✓	✓

III. PROPOSED METHOD

In this section, we describe the proposed approach for robust retraining-free GAN fingerprinting. In its simplest form, a GAN model consists of a generator and a discriminator, denoted by G and D , respectively. G takes a noise sample $z \in \mathbb{R}^{d_z} \sim P_z$ of dimension d_z as input and produces a synthetic image at the output. G and D are updated alternatively during training. The loss that D wants to maximize can be expressed as:

$$\mathcal{L}_D = \mathbb{E}_{x \sim p_x} \log D(x) + \mathbb{E}_{\substack{z \sim P_z \\ w \sim \{0,1\}^{d_w}}} \log(1 - D(G(z))), \quad (1)$$

where x denotes the real image and p_x its distribution, while G wants to minimize

$$\mathcal{L}_G = \mathbb{E}_{\substack{z \sim P_z \\ w \sim \{0,1\}^{d_w}}} \log(1 - D(G(z))). \quad (2)$$

To watermark the generator G , we insert an additional intermediate layer performing personalized normalization and consider two (trainable) parameter generation networks F_s and F_b , named

ParamGen Nets, taking as input the watermark message w with d_w bits ($w \in \{0, 1\}^{d_w}$) and producing respectively the scaling factor γ and bias β used in the PN layer of the generator¹, and a (trainable) watermark decoder D_w . We denote with G_w the generator G with the PN layer parameterized by w . The overall architecture is shown in Fig. 2. A model with the desired retraining-free fingerprinting functionality is achieved by training the architecture as described below, by introducing three new loss terms for training.

To instruct the network to embed a different watermark inside the images it produces for every different w , the watermark $w \in \{0, 1\}^{d_w}$ is randomly sampled and D_w is jointly trained to extract the watermark from $G_w(z)$, in such a way to minimize the bit-wise error between the watermark w and the output of the decoder, that is

$$\mathcal{L}_{wm} = \mathbb{E}_{z \sim P_z} \sum_{w \in \{0,1\}^{d_w}} w_i \log \sigma((D_w(G_w(z)))_i) + (1 - w_i) \log(1 - \sigma((D_w(G_w(z)))_i)), \quad (3)$$

where σ is the sigmoid function and $(D_w(G_w(z)))_i$ denote the i -th element of $D_w(G_w(z))$.

To ensure that the content of the generated image $G_w(z)$ is regulated by the input noise z , by following [3] and [12], we instruct the decoder D_w to recover the input noise z in addition to extracting the watermark w . Therefore, the output of D_w is a vector of length $d_w + d_z$ where the first d_w elements are dedicated to the watermark (Eq. 3) and the last d_z elements are used for input recovery. The above goal is achieved by minimizing the L_2 reconstruction loss

$$\mathcal{L}_z = \mathbb{E}_{z \sim P_z} \sum_{w \in \{0,1\}^{d_w}} \sum_{i=1}^{d_z} (z_i - (D_w(G_w(z)))_{d_w+i})^2. \quad (4)$$

Finally, to ensure that various model instances behave in the same way, regardless of the embedded watermark, we impose an additional constraint requiring that for the same sample noise z the generators parameterized by different w results in the same output image. Following [3], this goal is achieved via an image consistency loss

$$\mathcal{L}_{const} = \mathbb{E}_{z \sim P_z} \|G_{w_1}(z) - G_{w_2}(z)\|^2, \quad (5)$$

$w_1, w_2 \sim \{0,1\}^{d_w}$

where w_1 and w_2 are two distinct watermarks sampled randomly during training. Eventually, the overall loss used to train the watermarked generator is:

$$\mathcal{L}_{G,tot} = \lambda_1 \mathcal{L}_{G_w} + \lambda_2 \mathcal{L}_{wm} + \lambda_3 \mathcal{L}_z + \lambda_4 \mathcal{L}_{const}, \quad (6)$$

where λ_1 , λ_2 , λ_3 , and λ_4 weight each term of the loss (and \mathcal{L}_{G_w} is the loss defined in (2) with G_w in place of G). The loss of D is the same as in (1) with G_w replacing G .

Once the GAN fingerprinting model is trained, given a new user m , the model G_{w_m} containing the user-specific fingerprint w_m is obtained, by assigning PN layers the parameters obtained through ParamGen Nets. The generator is then distributed to the user, while F_s , F_b , and D_w are kept by the model owner.

A. Personalized Normalization

In this section, we provide the details of the ParamGen Nets F_s and F_b . These are two independent networks that take as input w and output $\gamma = F_s(w)$ and $\beta = F_b(w)$, that are used to normalize the feature map F at the input of the PN layer. Normalization is

¹The ParamGen Nets are described in details in Sect. III-A. Depending on the variant of the method (two variants are considered, see Sect. III-A), γ and β can be either vectors or tensors.

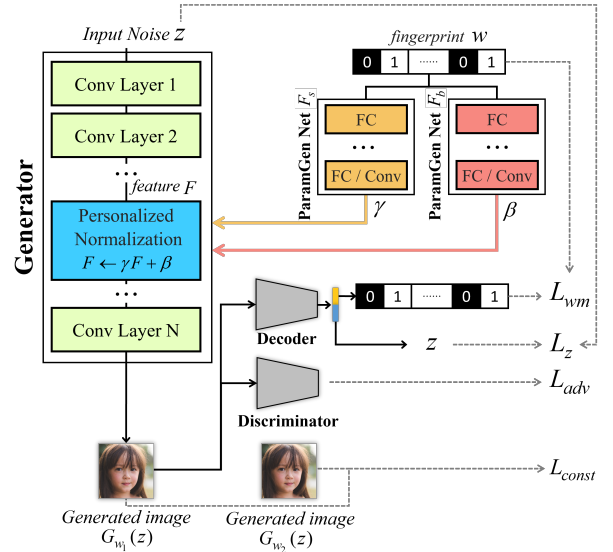


Fig. 2. Overview of the proposed GAN fingerprinting approach. Green blocks in the generator represent the original layers, while the blue block represents the PN layer added for watermark embedding.

performed in two different ways, referred to as channel-wise PN and element-wise PN, with the output of the ParamGen Nets γ and β having different dimensionality in the two cases.

1) **Channel-wise PN**: The output of the ParamGen Nets are $\gamma \in \mathbb{R}^c$, $\beta \in \mathbb{R}^c$, where c is the number of channels of the feature map F , and F_s and F_b are fully-connected (FC) networks.² Then, F is scaled and translated by channels and we have

$$F'_{ijk} = \gamma_k F_{ijk} + \beta_k, \quad (7)$$

for $i \in [1, p]$, $j \in [1, q]$, $k \in [1, c]$, where p and q are the height and width of F .

2) **Element-wise PN**: In this case, $\gamma \in \mathbb{R}^{p \times q \times c}$ and $\beta \in \mathbb{R}^{p \times q \times c}$ produced by two convolutional networks F_s and F_b that consists of an input FC layer followed by convolutional layers. Then, F is scaled and translated by the corresponding entries of γ and β , that is, $F' = \gamma \circ F + \beta$, where \circ denotes the element-wise product.

IV. EXPERIMENTAL METHODOLOGY AND ANALYSIS

A. Methodology and Settings

Models and dataset. We run experiments on several GAN architectures, focusing on the generation of face images. Specifically, we consider the following networks: Boundary Equilibrium GAN (BEGAN) [13], Spectral Normalization GAN (SNGAN) [14], and Progressive Growing GAN (PGGAN) [15]. As for the pristine set, we consider 200k images with 64×64 resolution from CelebA [16]. The models are trained using the official code, but the architecture and training procedure are modified to implement the proposed GAN fingerprinting approach.

Architecture and training. In the proposed GAN fingerprinting architecture, the PN layer is added as the penultimate layer of the generator (default). Other positions for the PN layer are considered in the ablation study presented in Sect. IV-D. The details of the ParamGen Nets are provided in Table II for both channel-wise and element-wise cases. For every layer (row), the type of layer, input dimension, output dimension, and the type of activation are reported.

²The exact architecture of the ParamGen Nets is provided in the methodology section.

Regarding the parameters, we set p and q for the element-wise PN equal to 32, while c , which corresponds to the number of channels in the feature maps, takes a different value for the various architectures. In particular, $c = 128, 64,$ and $128,$ respectively for BEGAN, SNGAN, and PGGAN. During training, the watermarks are randomly selected for each sample in every batch, hence every image has a different watermark message associated to it. The weights are set as follows: $\lambda_i = 1$ for $i = 1, 2, 4,$ $\lambda_3 = 0.1$. All models are trained for 50 epochs. In the case of PGGAN, which implements a progressive growing training procedure, blocks of layers are incrementally added during training, and the output size of the generator model (image resolution) progressively increases. Then, for this case, the generator is trained normally at the beginning (with the \mathcal{L}_G and \mathcal{L}_D loss). The PN layer and the watermark decoder are inserted only at a later stage, when the image resolution reaches $64 \times 64,$ and we started training the generator with the new loss³. In all the experiments, we set d_w to 128, thus embedding a 128-bit fingerprint, that permits us to get $2^{128} \approx 3.4 \times 10^{38}$ distinct generator instances.

To increase the robustness of the watermark against image preprocessing attacks, following [8], we introduce a preprocessing layer before the watermark decoder. The considered processing includes JPEG compression with quality factors selected uniformly in [20, 50], Gaussian blur, with kernel size in [0, 9], and Gaussian noise addition with standard deviation in [0.001, 0.15]. Each processing is applied with probability equal to 15%.

TABLE II
STRUCTURE OF PARAMGEN NETS IN THE CASE OF CHANNEL-WISE PN (LEFT) AND ELEMENT-WISE PN (RIGHT).

FC, d_w , 512, ReLu	FC, d_w , $8 \times 8 \times 32$, ReLu
FC, 512, 512, ReLu	Conv, $8 \times 8 \times 32$, $16 \times 16 \times 32$, ReLu
FC, 512, 512, ReLu	Conv, $16 \times 16 \times 32$, $32 \times 32 \times 64$, ReLu
FC, 512, c , ReLu	Conv, $32 \times 32 \times 64$, $p \times q \times c$, ReLu

Metrics. To measure the quality of the generated images, we use the Fréchet Inception Distance (FID) [17], commonly adopted in the literature. The FID scores of the generated images are calculated on a population of 5×10^4 real and 5×10^4 generated images, obtained from random generator instances, that is generators with random watermarks. To measure the effectiveness of watermark embedding, we use bit-wise accuracy (Acc) of watermark extraction, namely the percentage of bits that are correctly recovered. The bit-wise accuracy reported in the experiments is averaged on 10^4 samples with random watermarks.

B. Performance Analysis

The performance of our method for the various architectures is reported in Table III, compared with the state-of-the-art box-free GAN watermarking approaches. In the table, *-cw* and *-ew* refer to channel-wise and element-wise PN layer. The 'No wm' column shows the baseline FID of the images generated by the non-watermarked model.

Regarding the quality, we observe that all methods get an FID similar to the baseline, thus the quality of the generated images is similar to that of the images generated by the non-watermarked GAN. The watermark accuracy is also nearly perfect in all the cases for all the methods, except for Yu *et al.* [7] (the lower effectiveness of this method is probably due to the fact that it performs embedding in an

³In order to work properly, the watermark decoder requires that the input size is fixed.

unsupervised manner, by simply training on watermarked images). The row 'Overhead' reports the time required by algorithms for the embedding of a new watermark. For the method in [7], which requires retraining on the watermarked dataset, the time consumption is high. This value is high also for the method in [8], which however only requires finetuning, with a lower overhead with respect to [7]. Yu *et al.* [3] and our method only requires hundreds of milliseconds resulting in a huge gain (in the order of 10^4 to 10^5). In our case, this is the time necessary for a single feedforward pass of the ParamGen Nets.

TABLE III
WATERMARK ACCURACY, FID AND OVERHEAD FOR THE VARIOUS ARCHITECTURES.

Model	Metric	No wm	Yu [7]	Fei [8]	Yu [3]	<i>-cw</i>	<i>-ew</i>
BEGAN	Acc	-	93.69	99.10	100.00	99.87	100.00
	FID	20.89	25.12	20.84	21.78	21.19	20.72
	Overhead	-	12h	4h	100ms	100ms	100ms
SNGAN	Acc	-	92.77	99.45	99.99	100.00	99.99
	FID	24.25	27.74	25.26	24.69	24.12	24.70
	Overhead	-	8h	2h	100ms	100ms	100ms
PGGAN	Acc	-	90.26	98.74	99.50	99.89	99.85
	FID	27.50	32.36	28.53	28.42	28.77	28.02
	Overhead	-	24h	8h	100ms	100ms	100ms

C. Robustness Analysis

We also evaluate the robustness of our GAN fingerprinting method against both model-level and image-level attacks.

TABLE IV
PERFORMANCE (ACC/ FID) UNDER DIFFERENT MODEL-LEVEL ATTACKS.

Model	Approach	Finetune (20k)	Prune (10%)	Prune (20%)	Quant (10^{-1})
BEGAN	Yu [3]	53.6/ 20.6	99.1/21.2	62.1/76.6	97.1/34.8
	Fei [8]	56.5/20.7	99.1/ 20.5	61.7/82.2	98.0/ 32.0
	Ours <i>-cw</i>	75.9/20.9	99.5/22.0	66.6/ 74.2	98.8 /37.1
	Ours <i>-ew</i>	85.0 / 20.6	99.9 /21.9	68.8 /90.1	98.7/36.7
SNGAN	Yu [3]	60.0/24.3	99.9 / 24.8	82.6 /43.1	98.4/ 26.2
	Fei [8]	62.0/24.1	98.0/26.3	80.2/45.7	96.0/26.7
	Ours <i>-cw</i>	84.5/ 23.9	98.9/25.8	79.6/39.5	99.9/27.8
	Ours <i>-ew</i>	88.2 /24.0	98.8/26.9	80.3/ 39.1	99.8 /28.6
PGGAN	Yu [3]	64.2/28.4	98.9/29.6	88.4 / 37.1	99.1/30.1
	Fei [8]	65.3/28.1	97.4/30.2	84.2/39.8	99.0/32.4
	Ours <i>-cw</i>	73.0/ 27.2	99.5 /29.6	84.2/42.3	99.5/32.1
	Ours <i>-ew</i>	74.5 /28.0	99.1/ 29.5	85.3/40.1	99.9 / 29.7

For model-level attacks, we consider finetuning and model compression, namely pruning, and quantization. In the finetuning experiments, we perform 20k iterations of the GAN fingerprinting model on the same dataset used for training by removing the watermark losses from the optimization, that is, setting $\lambda_1 = 1$ and $\lambda_i = 0$, $i = 2, 3, 4$, and leaving the parameters of the PN layer free to update. For pruning, we set to 0 the smallest $p\%$ parameters of the network, with $p = 10\%$ and 20% . Finally, for quantization, we reduce the precision of the model parameters by rounding them to the first digit.

The results we got are shown in Table IV. We can observe that, for all the architectures, the proposed approach is more robust against

finetuning compared to [3] and [8], especially in the case of element-wise PN, where Acc is 20% higher with respect to the state-of-the-art on the average. All approaches are robust against pruning and quantization. Pruning with $p = 20\%$ can affect the watermark yet at the price of a very large FID, corresponding to a bad quality of the generated images, that makes the model useless.

Table V reports the results of robustness against image-level attacks, in the case of JPEG compression (with quality factor 50), Gaussian blur (with kernel size 5), and Gaussian noise (with standard deviation 0.1). We can observe that our method achieves the best robustness in all the cases, especially for JPEG compression, in which case the gain in the Acc is about 15/20%.

TABLE V
WATERMARK ACC UNDER DIFFERENT IMAGE-LEVEL ATTACKS

Model	Attack	Yu [3]	Fei [8]	-cw	-ew
BEGAN	JPEG	71.42	77.63	92.62	92.48
	Blurring	76.31	70.35	81.49	80.37
	Noise	75.93	74.17	88.50	89.33
SNGAN	JPEG	72.99	76.31	92.01	92.17
	Blurring	77.17	72.34	83.23	80.45
	Noise	74.34	73.34	88.21	89.52
PGGAN	JPEG	73.15	78.54	94.25	93.67
	Blurring	74.46	72.54	80.17	83.70
	Noise	73.39	76.40	86.14	88.28

D. Ablation Study

We carry out several experiments to investigate the impact of the loss terms and the position of the PN layer on the effectiveness of the proposed method. In these experiments, we focus on the element-wise PN embedding, which is the best performing method in terms of both image quality and watermark robustness, according to the previous results.

1) *Impact of the loss terms:* Table VI reports the Acc and FID obtained by training the model with and without the L_z and L_{const} terms and considering two generator instances corresponding to two random watermarks WM_1 and WM_2 . We see that removing L_z or L_{const} does not affect the watermark extraction accuracy. However, removing L_z results in a very high FID, as without this loss term the content of the generated images tends to be controlled solely by the watermark, thereby reducing the diversity. Fig. 3 shows some examples of images generated by the models marked with WM_1 and WM_2 , when they are trained with and without L_z . Although the visual quality of the generated images is good in both cases, without L_z , the generated images lack diversity.



Fig. 3. Images generated by the generators marked with WM_1 and WM_2 when fed with different input noises.

Regarding L_{const} , Table VI shows that when training is carried out without this loss term, both FID and Acc remain good. However, the purpose of this loss is to ensure that model instances with different watermarks have the same functionality, that is, that the watermarked

TABLE VI
PERFORMANCE OF G_{w1} AND G_{w2} WHEN TRAINING WITH/WITHOUT L_z AND L_{const} .

Approach	Acc $_{WM_1}$ / Acc $_{WM_2}$	FID $_{WM_1}$ / FID $_{WM_2}$
With all losses	100.00 / 100.00	20.72 / 20.68
Without L_z	100.00 / 100.00	233.38 / 205.86
Without L_{const}	100.00 / 100.00	21.18 / 20.94

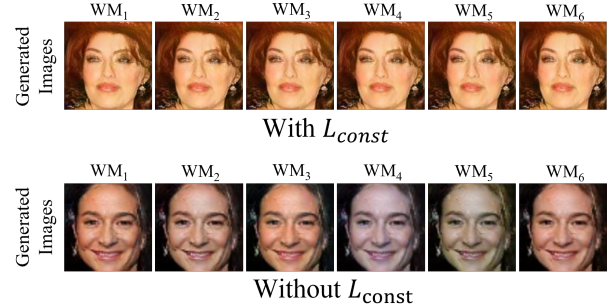


Fig. 4. Images generated by generator instances with different watermarks.

models generate images that are visually the same when fed with the same noise. Fig. 4 shows some images produced by 6 generator instances with different watermarks when they are fed with the same input noise z , in the case where training is performed with and without L_{const} . We can observe that without L_{const} , the images obtained for the same z from the 6 generators are visually different. This does not happen when the training includes the L_{const} loss.

TABLE VII
IMPACT OF THE POSITION OF THE PN LAYER (FT = FINETUNING).

Model	Metric	Input	Mid	Output (default)	All
BEGAN	Acc	100.00	100.00	100.00	100.00
	FID	21.24	20.89	20.72	20.24
	Acc after FT	68.24	72.68	75.92	64.46
SNGAN	Acc	99.99	99.90	99.99	99.98
	FID	24.15	24.76	24.70	24.01
	Acc after FT	72.54	84.32	88.21	64.24

2) *Impact of the position of the PN layer:* The previous experiments are carried out with the PN layer included as the penultimate layer (default position). In this section, we report the results of the experiments we run considering different positions for this layer. In particular, we consider the following cases: i) the PN layer is added as the second layer after the input layer (*Input*); ii) the PN layer is inserted as the middle layer (*Mid*); iii) multiple PN layers are included, after every convolutional block of the generator (*All*). In the *All* case, all the ParamGen Nets have the same internal architecture and the watermark message is fed as input to all of them. For these experiments, we consider only BEGAN and SNGAN, since in the PGGAN case the dynamic growth of the generator during training complicates the implementation of the approach. In particular, we find that including the PN layer as an intermediate layer at a late stage during training (when the final output resolution is achieved) makes training unstable.

The results are reported in Table VII. We observe that the position of the PN layer has a few impacts on Acc and FID. However, an impact is observed on watermark robustness. In particular, the *All* case

is the worst from the point of view of robustness, with a reduction of 11.46% for BEGAN and 23.97% for SNGAN, compared to the default setting. The *Mid* and penultimate layer position (*Default*) are those maximizing the robustness against finetuning attacks, with the default achieving the best results.

E. Discussion on collusion attacks

In this section, we pause to discuss a specific watermark removal attack, namely the collusion attack, that is particularly relevant in the GAN fingerprinting scenario. In this attack, different generator instances are combined together in order to produce a new generator that can achieve the same functionality and does not contain the watermark information of the source instances. Let G_{w_A} and G_{w_B} be two generator instances distributed to users A and B containing the user-specific fingerprints w_A and w_B . Then, an attacker who has access to the model parameters of G_{w_A} and G_{w_B} can generate a new model $G_{attk} = \alpha_1 G_{w_A} + \alpha_2 G_{w_B}$, with $\alpha_1 + \alpha_2 = 1$. Since the parameters of G_{w_A} and G_{w_B} are the same except for the PN layers, the interpolation only affects the parameters of the PN layer. Arguably, the watermark extracted by D_w from G_{attk} would contain only part of the watermark information of w_A and w_B . Fig. 5 shows the bit matching accuracy between the watermark extracted from G_{attk} and, respectively, w_A and w_B , for different values of α_1 . The FID, also reported in the figure, is approximately constant over α_1 . We observe that, in the case where $\alpha_1 = 0.5$, the correlation of the extracted watermark with the two watermarks remains high, and Acc is around 75% and 83% for watermark w_A and w_B respectively. Please note that watermark A and B are randomly selected once. With the increase in the number of experiments, when $\alpha_1 = 0.5$, the Acc for both watermarks will approach 50%. In

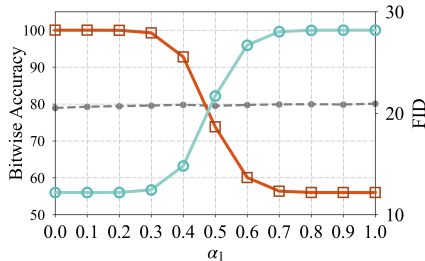


Fig. 5. Watermark accuracy and FID under collusion attack. The red curve with squares (green curve with circles) represents the bit matching accuracy between the watermark extracted from G_{attk} and w_A (w_B). The gray curve with dots reports the FID of the images generated by G_{attk} .

this condition, the collusion attack can be mitigated and the Acc recovered by incorporating error correction or traitor tracing codes. Since the number of distinct model instances that can be produced is extremely large (namely, 3.4×10^{38} with the payload of $d_w = 128$ bits considered in this paper), a fraction of bits could be utilized to perform error correction to prevent collusion by malicious users.

V. CONCLUSIONS

We have proposed a robust retraining-free GAN fingerprinting system that allows robust box-free watermarking of GAN generators, making it easy for the model owner to distribute copies of the generator with the same functionality but different watermarks (user-specific fingerprints). According to the experiments we carry out considering several architectures, the proposed method achieves very good results, always overcoming the state-of-the-art in terms of robustness against model-level and image-level attacks. Future work

will focus on the extension of the proposed method to different generative architectures, e.g., diffusion models, also considering images belonging to different domains. Investigating the use of channel coding and error correction codes to increase the robustness against watermark removal and collusion attacks is also a very interesting topic for future research.

ACKNOWLEDGMENT

This work is supported in part by the National Key R&D Program of China under Grant number 2022YFB3103100, in part by the National Natural Science Foundation of China under grant numbers 62122032, 62172233, 62102189. This work has been partially supported by the China Scholarship Council (CSC), file No.202109040029.

REFERENCES

- [1] M. Barni, F. Pérez-González, and B. Tondi, “Dnn watermarking: four challenges and a funeral,” in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 2021, pp. 189–196.
- [2] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3630–3639.
- [3] N. Yu, V. Skripniuk, D. Chen, L. S. Davis, and M. Fritz, “Responsible disclosure of generative models using scalable fingerprinting,” in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25–29, 2022*, 2022.
- [4] J. Zhang, D. Chen, J. Liao, W. Zhang, G. Hua, and N. Yu, “Passport-aware normalization for deep model protection,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 619–22 628, 2020.
- [5] Y. Li, H. Wang, and M. Barni, “A survey of deep neural network watermarking techniques,” *Neurocomputing*, vol. 461, pp. 171–193, 2021.
- [6] H. Wu, G. Liu, Y. Yao, and X. Zhang, “Watermarking neural networks with watermarked images,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 7, pp. 2591–2601, 2020.
- [7] N. Yu, V. Skripniuk, S. Abdelnabi, and M. Fritz, “Artificial fingerprinting for generative models: Rooting deepfake attribution in training data,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 448–14 457.
- [8] J. Fei, Z. Xia, B. Tondi, and M. Barni, “Supervised gan watermarking for intellectual property protection,” in *2022 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2022, pp. 1–6.
- [9] T. Qiao, Y. Ma, N. Zheng, H. Wu, Y. Chen, M. Xu, and X. Luo, “A novel model watermarking for protecting generative adversarial network,” *Computers & Security*, p. 103102, 2023.
- [10] Y. Quan, H. Teng, Y. Chen, and H. Ji, “Watermarking deep neural networks in image processing,” *IEEE transactions on neural networks and learning systems*, vol. 32, no. 5, pp. 1852–1865, 2020.
- [11] M. Tancik, B. Mildenhall, and R. Ng, “Stegastamp: Invisible hyperlinks in physical photographs,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2117–2126.
- [12] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, “Veegan: Reducing mode collapse in gans using implicit variational learning,” *Advances in neural information processing systems*, vol. 30, 2017.
- [13] D. Berthelot, T. Schumm, and L. Metz, “Began: Boundary equilibrium generative adversarial networks,” *arXiv preprint arXiv:1703.10717*, 2017.
- [14] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” in *International Conference on Learning Representations*.
- [15] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” in *International Conference on Learning Representations*.
- [16] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3730–3738.
- [17] 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,” *Advances in neural information processing systems*, vol. 30, 2017.