Protecting Intellectual Property of Generative Adversarial Networks from Ambiguity Attacks

Ding Sheng Ong[1] Chee Seng Chan[1*] Kam Woh Ng[2] Lixin Fan[2] Qiang Yang[3]

[1] University of Malaya [2] WeBank AI Lab [3] Hong Kong University of Science and Technology

Abstract

Ever since Machine Learning as a Service emerges as a viable business that utilizes deep learning models to generate lucrative revenue, Intellectual Property Right (IPR) has become a major concern because these deep learning models can easily be replicated, shared, and re-distributed by any unauthorized third parties. To the best of our knowledge, one of the prominent deep learning models - Generative Adversarial Networks (GANs) which has been widely used to create photorealistic image are totally unprotected despite the existence of pioneering IPR protection methodology for Convolutional Neural Networks (CNNs). This paper therefore presents a complete protection framework in both black-box and white-box settings to enforce IPR protection on GANs. Empirically, we show that the proposed method does not compromise the original GANs performance (i.e. *image generation, image super-resolution, style transfer), and at the same time, it is able to withstand both removal and ambiguity attacks against embedded watermarks. Codes are available at* https://github.com/dingsheng-ong/ipr-gan.

1. Introduction

Intellectual Property (IP) refers to the protection of creations of the mind, which have both a moral and commercial value. IP is protected under the law framework in the form of, *e.g.* patents, copyright, and trademarks, which enable inventors to earn recognition or financial benefit from their inventions. Ever since Machine Learning as a Service emerges as a viable business which utilizes deep learning (DL) models to generate revenue, different effective methods to prove the ownership of DL models have been studied and demonstrated [1, 16, 25, 29, 30]. The application domains demonstrated with these pioneering works, however, are invariably limited to Convolutional Neural Networks (CNNs) for classification tasks. Based on our knowledge, the protection for another prominent DL models, *i.e.* Generative Adversarial Networks (GANs) [5] that create plausible realistic photographs is missing all together and therefore urgently needed.

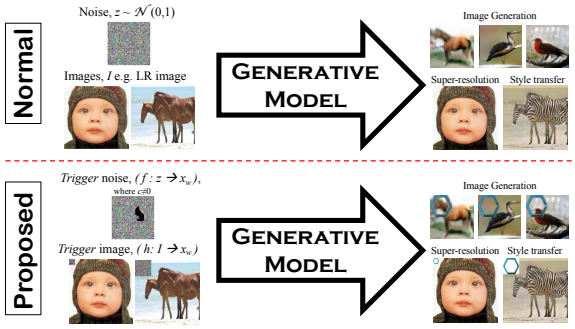


Figure 1: Overview of our proposed GANs protection framework in black-box setting. The idea is when a *trigger*, x_w is acted as an input, a watermarked image (*e.g.* with a hexagon as the watermark) will be synthesized to claim the ownership. Black area in the *trigger* noise ($f : z \rightarrow x_w$) indicates masked values (see Sec. 3.1.1, Eq. 1).

Generally, a common approach to deep neural network IP protection is based on digital watermarks embedding methods which can be categorized into two schools: i) the black-box trigger-set based solutions [1, 30]; and ii) the white-box feature-based methods [3, 6, 25]. The principle of digital watermarking is to embed an identification information (*i.e.* a digital watermark) into the network parameters without affecting the performances of original DL models. In the former, the watermark is embedded in the input-output behavior of the model. The set of input used to trigger that behavior is called *trigger set*. The non-triviality of ownership of a watermarked model is constructed on the extremely small probability for any other model to exhibit the same behavior. In the latter, the watermark is embedded in the static content of CNNs (*i.e.* weight matrices) with a transformation matrix. The ownership is verified by the detection of the embedded watermarks.

For the verification process, a suspicious online model will be first remotely queried through API calls using a specific input keys that were initially selected to *trigger* the

*Corresponding author, e-mail: cs.chan@um.edu.my

Trained Model	**BER**
DCGAN with $\boldsymbol{X}$ and $\boldsymbol{b}$	0.00
DCGAN with $\boldsymbol{X'}$ and $\boldsymbol{b'}$	0.00
SRGAN with $\boldsymbol{X}$ and $\boldsymbol{b}$	0.00
SRGAN with $\boldsymbol{X'}$ and $\boldsymbol{b'}$	0.00

Table 1: Top row - Bit-error rate (BER) of the trained model using Uchida *et al*. method [1]. Bottom row - BER of the model using counterfeit watermark, $\boldsymbol{b'}$ and optimized transformation matrix, $\boldsymbol{X'}$. DCGAN is trained on CIFAR10 dataset while SRGAN is trained on DIV2K dataset.

watermark information. As such, this is a *black-box verification* where a final model prediction (*e.g*. image classification results) is obtained. This initial step is usually performed to collect evidence from everywhere so that an owner can identifies a suspected party who used (*i.e*. infringed) his/her models illegally. Once the owner has sufficient evidence, the second verification process which is to extract watermark from the suspected model and compare if the watermark is originated from the owner. This process is a *white-box verification*, which means the owner needs to have to access the model physically, and usually this second step is gone through the law enforcement.

1.1. Problem Statement

Literally, both black-box and white-box schemes have been successfully demonstrated for CNNs [1, 16, 25, 29, 30], however it remains an open question to apply these protection mechanisms to important GANs variants (see [5] for a survey). We believe, intuitively, the lack of protection might be i) partially ascribed to the large variety of GANs application domains, for which how to embed watermarks through appropriate regularization terms is challenging, and ii) directly applying the popular CNN-based watermarking approach (*i.e*. Uchida *et al*. [25]) on GANs has limitation in ambiguity attack as shown in Table 1. It is shown that the ownership is in doubt as indicated by the BER results[1] (*i.e*. both the original $\boldsymbol{b}$ and forged $\boldsymbol{b'}$ watermarks are detected).

1.2. Contributions

Thus, we are motivated to present a complete IP protection framework for GANs as illustrated in Fig. 1. The contributions are twofold: i) we put forth a general IPR protection formulation with a novel regularization term $\mathcal{L}_w$ (Eq. 3) that can be generalized to all GANs variants; and ii) we propose a novel and complete ownership verification method for different GANs variants (*i.e*. DCGAN, SRGAN and CycleGAN). Extensive experiments show that ownership verification in both white and black box settings are effective without compromising performances of the original tasks (see Tables 3, 4, 5 and Fig. 6). At the same time, we tested the proposed method in both *removal* and *ambiguity* attacks scenario (see Tables 7-8 and Fig. 7-8).

[1]In general, bit-error rate (BER) measures how much the watermark is deviated. BER=0 implies that the watermark is exactly the same as to original, so ownership is claimed.

2. Related Work

Conventionally, digital watermarks were extensively used in protecting the ownership of multimedia contents, including images [10, 23], videos [2, 19], audio [8, 13, 22], or functional designs [18]. The first effort that propose to use digital watermarking technology in CNNs was a white-box protection by Uchida *et al*. [25], who had successfully embedded watermarks into CNNs without impairing the performance of the host network. It was shown that the ownership of network models were robustly verified against a variety of *removal attacks* including model fine-tuning and pruning. However, Uchida *et al*. [25] method was limited in the sense that one has to access all the network weights in question to extract the embedded watermarks. Therefore, [16] proposed to embed watermarks in the classification labels of adversarial examples, so that the watermarks can be extracted remotely through a service API without the need to access the network internal weights parameters. Later, [1] proved that embedding watermarks in the networks' (classification) outputs is actually a designed *backdooring* and provided theoretical analysis of performances under various conditions.

Also in both black box and white box settings, [3, 6, 14] demonstrated how to embed watermarks (or fingerprints) that are robust to watermark overwriting, model fine-tuning and pruning. Noticeably, a wide variety of deep architectures such as Wide Residual Networks (WRNs) and CNNs were investigated. [30] proposed to use three types of watermarks (*i.e*. *content, unrelated* and *noise*) and demonstrated their performances with MNIST and CIFAR10. Recently, [9, 29] proposed passport-based verification schemes to improve robustness against ambiguity attacks.

However, one must note that all aforementioned existing work are invariably demonstrated to protect CNN only. Although *adversarial examples* have been used as watermarks *e.g*. in [16], based on our knowledge, it is not found any previous work that aim to provide IP protection for GANs. The lack of protection might be partially ascribed to the large variety of GANs application domains, for which how to embed watermarks through appropriate regularization terms is challenging and remains an open question. For instance, the generic watermarked framework proposed by Uchida et al. [25] for CNNs could not be applied to GANs due to a major different in the input and output of GANs against the CNNs. Specifically, the input source for GANs can be either a latent vector z or image(s), I rather than just image(s) in CNN; while the output of GANs is a synthesis image(s) instead of a classification label. Nonetheless, our preliminary results (Table 1) and Fan *et al*. [9] disclosed that [25]

is vulnerable against *ambiguity* attacks.

Last but not least, one must differentiate a plethora of neural network based watermarking methods, which aim to embed watermarks or hide information into digital media (e.g. images) instead of networks parameters. For instance, [21] employed two CNN networks to embed a one-bit watermark in a single image block; [26] investigated a new family of transformation based on deep learning networks for blind image watermarking; and [31] proposed an end-to-end trainable framework, HiDDeN for data hiding in color images based on CNNs and GANs. Nevertheless, these methods are meant to protect the IP of processed digital media, rather than that of the employed neural networks.

3. Watermarking in GANs

GANs consists of two networks, a generative network, G that learns the training data distribution and a discriminative network D that distinguishes between synthesize and real samples [11]. This paper proposes a simple yet complete protection framework (black-box and white-box) by embedding the ownership information into the generator, G with a novel regularization term. Briefly, in black-box scenario, we propose the reconstructive regularization to allow the generator to embed a unique watermark, at an assigned location of the synthesize image when given a trigger input (see Fig. 1). While, in white-box scenario, we adopt and modify the sign loss in [9] that enforces the scaling factor, γ in the normalization layer to take either positive or negative values. With this, the sign of γ can be transformed into binary sequences that carry meaningful information.

For this work, we decided to demonstrate on three GANs variants, namely, DCGAN [24], SRGAN [17] and CycleGAN [32] to present the flexibility of our proposed framework. With trivial modifications, our method can easily extend to other deep generative models, *i.e.* VAE, as long as X outputs an image given a vector or image as the input.

3.1. Black-box

In general, we propose a reconstructive regularization that instructs the generator, G to map a *trigger* input to a specific output. Herein, the challenge is defining an appropriate transformation function to ensure that the distribution of *trigger set* is distinct from the actual data. In GANs, since the generator, G always output (synthesize) an image, the specific output will be a watermark-based image since the watermark (*e.g.* company's logo) holds unambiguous visual information which is straightforward to verify the ownership. The detailed implementations are described below:

3.1.1 DCGAN

Technically, the input to DCGAN is a latent vector randomly sampled from a standard normal distribution, $\boldsymbol{z} \sim \mathcal{N}(0, 1)$. Hence, we define a new input transformation function, f, that maps the latent vector to a *trigger* latent vector ($f : \boldsymbol{z} \mapsto \boldsymbol{x}_w$) as follow:

Figure 2: Image pair of the generated images using latent inputs, $G(\boldsymbol{z})$ (left) and *trigger* inputs, $G(\boldsymbol{x}_w)$ (right), respectively. Each pair is a DCGAN model trained on different watermarks.

$$f(\boldsymbol{z}) = \boldsymbol{z} \odot \boldsymbol{b} + c(1 - \boldsymbol{b}) \quad \text{and} \quad \boldsymbol{b} \in \{0, 1\}^{\mathbb{D}(\boldsymbol{z})} \tag{1}$$

Intuitively, Eq. 1 masks the $n \in W$ value of the latent vector, $\boldsymbol{z}$ to a constant value, $c \in R$ where the position of the n values are determined by a predefined bitmask, $\boldsymbol{b}$ and $\mathbb{D}$ is the dimension.

Then, in order to transform the generator output to a specific target, we define the new output transformation function as $g : G(\boldsymbol{z}) \mapsto \boldsymbol{y}_w$ where it will apply an unique watermark on the generator output. The equation can be pictorially represented as:

(2)

After defining both the input/output transformation functions, we define the reconstructive regularization derived from the structural similarity (SSIM) [27] which measures the perceived quality between two images. Since the range of SSIM is in $[0, 1]$, we define the regularization to optimize as:

$$\mathcal{L}_w\left(\boldsymbol{x}_w, \boldsymbol{y}_w\right) = 1 - \text{SSIM}\left(G(\boldsymbol{x}_w), \boldsymbol{y}_w\right) \tag{3}$$

For the experiment purpose, we have chosen Spectral Normalization GAN (SN-GAN) [20] which is a variant of DCGAN. Taking the generator's objective function (Eq. 4), we optimize the regularization term defined in Eq. 3 and the generator's objective function simultaneously:

$$\mathcal{L}_{\text{DC}} = -\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[\hat{D}\left(G\left(\boldsymbol{z}\right)\right)\right] \tag{4}$$

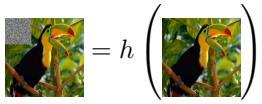
$$\mathcal{L}_{\text{DC}_w} = \mathcal{L}_{\text{DC}} + \lambda \mathcal{L}_w \tag{5}$$

with the reconstructive regularization scaled by associated hyper-parameter, λ to balance between the quality of generated image and the perceived similarity of the generated watermark when the *trigger* input is provided.

Figure 3: First image is a sample of *trigger* input $\boldsymbol{x}_w$ to SRGAN. Next three images are the special targets $G(\boldsymbol{x}_w)$ from SRGAN models trained on different watermarks.

3.1.2 SRGAN

SRGAN has been the foundation for most, if not all, the recent super-resolution tasks, in which the generator of SRGAN takes a low resolution image, I and generate a respective high resolution image. As such, the input transformation function, f defined in Section 3.1.1 will not be applicable. For that reason, we define an alternative input transformation function, h that maps an image input to a *trigger set* ($h : \boldsymbol{I} \mapsto \boldsymbol{x}_w$). This function is almost identical to Eq. 2 with an exception that a random noise is embedded on the input image, rather than a watermark. The function, h can be visually represented as:



$$= h \left(\right) \tag{6}$$

For the output transformation function, since the output from all variant of GANs is the same (*i.e*. an image), we can re-use g and reconstructive regularization (Eq. 3) to transform the output of SRGAN to embed a unique watermark on the generated high-resolution image. The generator loss function composed of a content loss and an adversarial loss and we use the VGG loss defined on feature maps of higher level features as described in [17]:

$$\mathcal{L}_{\text{SR}} = l^{SR}_{VGG/5,4} + 10^{-3} l^{SR}_{Gen} \tag{7}$$

To this end, the new objective function of our protected SRGAN is denoted as

$$\mathcal{L}_{\text{SR}_w} = \mathcal{L}_{\text{SR}} + \lambda \mathcal{L}_w \tag{8}$$

3.1.3 CycleGAN

The generators in CycleGAN [32] take an image, I from a domain as input and translate the image into a same size image of another domain. Provided with this fact, we can use the function h defined in Eq. 6 to map a training input to a *trigger set* and consistently employ function g defined in Eq. 2 to embed the watermark on the output image. Yet, we use the same reconstructive regularization defined in Eq. 3 and add to the generator loss of CycleGAN. Even though there are two generators in CycleGAN, we only need to select one of them as our target for protection. The objective function of the selected generator is given as:

Figure 4: Image pairs, $\boldsymbol{x}_w / G(\boldsymbol{x}_w)$ from different CycleGAN models trained on horse2zebra (first row) and Cityscapes (second row) datasets, respectively

$$\begin{aligned}\mathcal{L}_{\text{GAN}} =& \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}(\boldsymbol{y})} \left[log D_Y(y) \right] + \\ & \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[log(1 - D_Y(x)) \right]\end{aligned}$$

$$\mathcal{L}_{\text{Cyc}} = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\| F(G(x)) - x \|_1 \right]$$

$$\mathcal{L}_{\text{C}} = \mathcal{L}_{\text{GAN}} + \mathcal{L}_{\text{Cyc}} \tag{9}$$

Thus, the new objective for our CycleGAN is:

$$\mathcal{L}_{\text{C}_w} = \mathcal{L}_{\text{C}} + \lambda \mathcal{L}_w \tag{10}$$

Verification. For the verification in black-box setting, initially, any suspected online GAN models will be queried remotely by owner (company) via API calls to gather evidence. That is to say, owner (company) submits a list of *trigger set* data as query to the GANs online service that is in question. Evidence will be collected as a proof of ownership if the response output is embedded with the designated watermark logo (see Fig. 2, 3, 4 for examples). Moreover, the verification of the embedded watermark can be measured by calculating the SSIM between the expected output $\boldsymbol{y}_w$ and the output generated by the model $G(\boldsymbol{x}_w)$, with *trigger* input is provided, and the sample results are shown in Fig. 7a. SSIM score reflects the perceived similarity between the generated watermark and the ground truth watermark and a score of above 0.75 should give an unambiguous, distinctive watermark that can be used in ownership verification (see Fig. 5).

3.2. White-box

In order to provide a complete protection for GANs, we adopt the sign loss introduced in [9] as a designated key (*i.e*. signature) which have been proven to be robust to both removal and ambiguity attacks. Specifically, such signatures are embedded into the *scaling factors*, $\boldsymbol{\gamma}$ of normalization layers with C channels in the generators, which can be

0.00 0.25 0.50 0.75 1.00

Figure 5: Different perceived quality of watermark and the SSIM score respectively.

GAN	Channels	Capacity
DCGAN	448	56 bytes
SRGAN	2112	264 bytes
CycleGAN	5248	656 bytes

Table 2: The amount of information that can be embedded into GAN generators.

then retrieved and decoded for ownership verification purpose. Eq. 11 serves as a guidance for the sign of a weight in the normalization layers.

$$\mathcal{L}_s(\boldsymbol{\gamma}, \boldsymbol{B}) = \sum_{i=1}^{C} \max\left(\gamma_0 - \gamma_i b_i, 0\right) \tag{11}$$

where $\boldsymbol{B} = \{b_1, \cdots, b_C \mid b \in \{-1, 1\}\}$ is the defined binary bit signature that, when optimize this objective, will enforce the i-th channel's scaling factor, γ_i to take either positive or negative polarity (+/-) as designated by b_i. γ_0 is a constant to control the minimum value of $\boldsymbol{\gamma}$ (to avoid all 0s $\boldsymbol{\gamma}$).

Then, this regularization term is added to the objective functions of DCGAN (Eq. 5), SRGAN (Eq. 8) and CycleGAN (Eq. 10). To this end, the overall objective for the generators are respectively denoted as:

$$\begin{aligned}
\mathcal{L}_{\text{DC}_{ws}} &= \mathcal{L}_{\text{DC}} + \lambda\mathcal{L}_w + \mathcal{L}_s \\
\mathcal{L}_{\text{SR}_{ws}} &= \mathcal{L}_{\text{SR}} + \lambda\mathcal{L}_w + \mathcal{L}_s \\
\mathcal{L}_{\text{C}_{ws}} &= \mathcal{L}_{\text{C}} + \lambda\mathcal{L}_w + \mathcal{L}_s
\end{aligned}$$

With the sign loss incorporated into the training objective, the scaling factor of normalization layers in generator are now in either positive or negative value where the unique binary sequence can be used to resemble the ownership information of a particular network. The capacity of embedded information (see Table 2) is constrained by the total number of channels in normalization layers. For example in our DCGAN model, the total number of channels for each layer are 256, 128 and 64 respectively. Thus, we can embed at most 448 bits, equivalent to 56 bytes into the model. As for SRGAN, intuitively, more information can be embedded as it has more layers than DCGAN model and so does CycleGAN. The detail information is represented in supp. material. We refer readers to Section 4.6 for superior performances of the sign-loss based method, demonstrated by extensive experiment results.

Verification. Given the evidence from black-box verification step in Section 3.1, the owner can subsequently go through law enforcement and perform white-box verification which to access the model physically to extract the signature. As an example shows in our supp. material, we embed an unique key "EXAMPLE" into our DCGAN's batch normalization weight. It shows how to decode the trained scale, $\boldsymbol{\gamma}$ to retrieve the signature embedded. Also, please note that even that there are two or more similar alphabets, their $\boldsymbol{\gamma}$ are different from each other, respectively.

4. Experimental Results

This section illustrates the empirical study of our protection framework on the GAN models. To make a distinction between the baseline models and the protected models, we denote our proposed GAN models with subscript w and ws where GAN_w models (*i.e.* DCGAN_w, SRGAN_w, CycleGAN_w) are the protected GANs in black-box setting using only the reconstructive regularization, $\mathcal{L}_w$ whereas GAN_{ws} models (*i.e.* DCGAN_{ws}, SRGAN_{ws}, CycleGAN_{ws}) represent the protected GAN generators in both black-box and white-box settings using both of the regularization terms, $\mathcal{L}_w$ and sign loss, $\mathcal{L}_s$.

4.1. Hyperparameters

We strictly followed all the hyperparameters and the architecture defined in the original works for each GAN model. The only modification that we had made is adding regularization terms to the generator loss. As discussed in Section 3.1, we trained the DCGAN models using CIFAR10 dataset [15] aligned using the architecture and the loss function proposed in [20]. We used the logos shown in the top left corner of Fig. 2 as our watermark that revealed when the *trigger* input is presented as illustrated in Fig. 1. The coefficient, λ is set to 1.0 for all experiments unless stated otherwise. Unlike SRGAN and CycleGAN, the transformation function, f (Eq. 1) used in DCGAN has extra parameters n and c to consider, in which we decided to employ $n = 5$ and $c = -10$ after a simple ablation study as reported in Section 4.7. The size of the watermark is 16×16 compared to the generated image with resolution 32×32 so that the watermark is not too small to be visible. Besides, CIFAR10, the exactly same setting were used to train on the CUB200 dataset [28] which has a higher resolution (64×64).

Likewise, we trained SRGAN on randomly sampled 350k images from ImageNet [7] and adopted the architecture and hyper-parameters presented in [17]. In the super resolution task, the training images are up-sized 4 times from 24×24 to 96×96. As discussed in Section 3.1, we used the transform function, h (Eq. 6) to paste a random noise of size 12×12 onto the input image, at the same time, we employed function, g (Eq. 2) to attach a watermark of size 48×48 onto the output image.

As for CycleGAN, we trained the model on Cityscapes dataset [4] but only protect one of the generator (label $\rightarrow$

	CIFAR-10	CUB-200
DCGAN	26.54 ± 1.04	58.34 ± 1.50
DCGAN$_w$	24.83 ± 0.37	53.07 ± 4.07
DCGAN$_{ws}$	26.27 ± 0.54	56.64 ± 2.74

Table 3: Fidelity in DCGAN: Scores are in FID ($\downarrow$ is better).

photo) as to prevent redundancy. Except the regularization terms ($\mathcal{L}_w$, $\mathcal{L}_s$), we keep to the parameters defined in [33]. The setting is very similar to SRGAN's with the resolution of the random noise and watermark in 32×32 compared to the size of the training images in 128×128.

4.2. Evaluation Metrics

To evaluate the generative models quantitatively, we use a set of metrics to measure the performance of each model. For image generation task with DCGAN, we calculate the Frechet Inception Distance (FID) [12] between the generated and real images tested on CIFAR10 and CUB-200 as the benchmark datasets. For image super-resolution with SRGAN, we use PSNR and SSIM as our metrics and employ Set5, Set14, BSD100 (testing set of BSD300) as the benchmark datasets. According to the original paper [17], all measures were calculated on the y-channel. We performed the same in order to have a fair comparison with [17]. As for CycleGAN, we measure the FCN-scores as presented in [33] on the Cityscapes label $\rightarrow$ photo which consists of per-pixel acc., per-class acc. and class IoU.

The watermark quality is measured in SSIM between the ground truth watermark image and the generated watermark image when a *trigger*, x_w is presented. To avoid a confusion with SSIM used in SRGAN, we denote this metrics as Q_{wm} which implies the quality of watermark. The signature embedded into the normalization weights are measured in bit-error rate (BER) when compared to the defined signature, $\boldsymbol{B}$ (see Eq. 11).

4.3. Fidelity

In this section, we compare the performance of each GAN models against the GAN models protected using the proposed framework. According to Table 3, it is observed that the performances of the protected DCGAN (*i.e.* DCGAN$_w$ and DCGAN$_{ws}$) are comparable or slightly better in terms of FID score on CIFAR-10 datasets. However, there is a slightly drop in performance when trained using CUB-200 datasets.

The difference in performances of the protected SRGAN (*i.e.* SRGAN$_w$ and SRGAN$_{ws}$) and the baseline SRGAN is subtle, where the PSNR deviates for 0.87 and SSIM deviates for 0.02 at most across all the datasets. Moreover, qualitatively, we also illustrate in Fig. 6 that the performance of our proposal does not degrade much compared to the baseline after added regularization terms to the training objective. Meanwhile, CycleGAN$_w$ has an identical FCN-score with the baseline CycleGAN and CycleGAN$_{ws}$ has a noticeable improvement. In short, adding the regularization terms has minimal effect to the performance of the GANs in respective tasks while it may slightly improve the performance in some conditions.

	Set5	Set14	BSD
SRGAN	29.38/0.85	25.92/0.71	25.08/0.67
SRGAN$_w$	29.35/0.85	25.46/0.71	24.21/0.65
SRGAN$_{ws}$	29.14/0.85	26.00/0.72	25.35/0.67

Table 4: Fidelity in SRGAN: Scores are in PSNR/SSIM ($\uparrow$ is better).

	Per-pixel acc.	Per-class acc.	Class IoU
CycleGAN	0.55	0.18	0.13
CycleGAN$_w$	0.55	0.18	0.13
CycleGAN$_{ws}$	0.58	0.19	0.14

Table 5: Fidelity in CycleGAN: Scores are in per-pixel acc., per-class acc. and class IoU ($\uparrow$ is better).

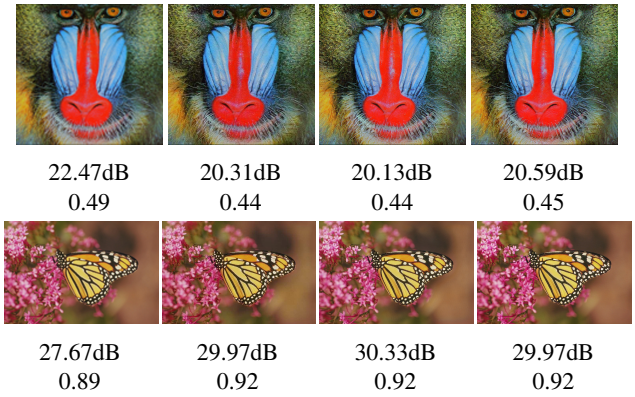
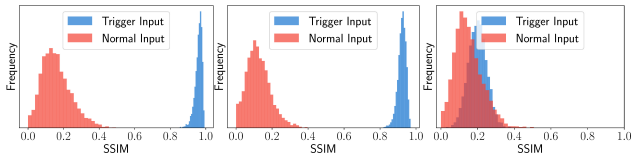


Figure 6: Fidelity (SRGAN): From left to right - bicubic upsample, output from SRGAN, SRGAN$_w$, SRGAN$_{ws}$, respectively. Scores are in (PSNR(db) / SSIM).

4.4. Verification

Black-box. In this section, we will discuss the verification process using the quality of the watermark, Q_{wm} which is the SSIM computed at the generated watermark with the ground truth watermark. Table 6 and Fig. 7a shows that the score is high (close to 1) when the *trigger* inputs are given in comparison to the normal inputs. This implies that the watermark generated is very similar to the ground truth watermark (see Fig. 2, 3, 4). As a result, this provides a strong evidence for the owner to claim the ownership to the specific GAN model as the model will output an unambiguous logo that represent the owner.

White-box. Subsequently, if the black-box verification does not provide convincing evidence, the next step is to further investigate the weights of suspicious model in used. That is, to extract the signature from the weights at the



(a) Verification (b) Fine-Tuning (c) Overwriting

Figure 7: Distribution of watermark quality, Q_{wm} measured in SSIM using 500 samples. (a) shows the distributions before the removal attacks (b) shows the distributions after fine-tuning, (c) shows the distributions after overwriting.

	Q_{wm}	BER
DCGAN$_{ws}$	0.97 ± 0.01	0
SRGAN$_{ws}$	0.93 ± 0.10	0
CycleGAN$_{ws}$	0.90 ± 0.02	0

Table 6: Quality of the watermark, Q_{wm} and BER in DCGAN, SRGAN and CycleGAN.

normalization layers and convert the signatures into ASCII characters as shown in supp. material. In this experiment, we embed the word "EXAMPLE" into the normalization layers, however, in real use case, the owner can embed any words such as company name etc. as the ownership information. In this experiment, all of the protected GAN models has BER=0 which implies the signature embedded 100% matches with the defined binary signature, $\boldsymbol{B}$.

4.5. Robustness against removal attacks

Fine-tuning. Here, we simulate an attacker fine-tune the stolen model with a dataset to obtain a new model that inherits the performance of the stolen model while trying to remove the embedded watermark. That is, the host network is initialized using the trained weights embedded with watermark, then is fine-tuned without the presence of the regularization terms, *i.e.* $\mathcal{L}_w$ and $\mathcal{L}_s$.

In Table 7, we can observe a performance drop (26.54 $\rightarrow$ 30.50) when the attacker fine-tune $DCGAN_{ws}$ to remove the embedded watermark while the watermark quality, Q_{wm} is still relatively high (0.92) indicates that the watermark generated is still recognizable, further supported by Fig. 7c which shows the distribution of Q_{wm} after fine-tuning has no obvious changes. We also observe the same behaviour when fine-tuning SRGAN$_{ws}$ and CycleGAN$_{ws}$ in which the performance is slightly declined (see Tables 8 and 9). Qualitatively, we also can clearly visualize that the watermark before and after the fine-tuning is well preserved for all the GAN models. Empirically, this affirms that our method is robust against removal attempt by fine-tuning as the attempt is not beneficial and failed in removing the embedded watermark.

Overwriting. We also simulate the overwriting scenario where the attacker is assumed to embed a new watermark into our trained model using the same method as proposed.

	FID	Q_{wm}	BER
DCGAN$_{ws}$	26.54 ± 1.04	0.97	0
Fine-tune	30.50 ± 1.10	0.96	0
Overwrite	35.68 ± 1.10	0.49	0

Table 7: First row is the FID scores, watermark quality, Q_{wm} and BER for DCGAN$_{ws}$. Second row shows the scores after fine-tuning and third row shows the scores after overwriting attack.

	Set5	Set14	BSD	Q_{wm}	BER
SRGAN$_{ws}$	29.14/0.85	26.00/0.72	25.35/0.67	0.93	0
Fine-tune	26.07/0.85	23.75/0.72	23.58/0.68	0.83	0
Overwrite	27.65/0.84	25.08/0.72	24.66/0.68	0.17	0

Table 8: First row is the PSNR/SSIM scores, watermark quality, Q_{wm} and BER for SRGAN$_{ws}$. Second row shows the scores after fine-tuning and third row shows the scores after overwriting attack.

Figure 8: From left to right shows the performance of DCGAN, SRGAN, CycleGAN when different percentage (%) of the sign(γ) is being modified (compromised).

	Per-pixel acc.	Per-class acc.	Class IoU	Q_{wm}	BER
CycleGAN$_{ws}$	0.58	0.19	0.13	0.90	0
Fine-tune	0.55	0.18	0.14	0.85	0
Overwrite	0.57	0.17	0.13	0.15	0

Table 9: First row is the FCN-scores, watermark quality, Q_{wm} and BER for CycleGAN$_{ws}$. Second row shows the scores after fine-tuning and third row shows the scores after overwriting attack.

Tables 7, 8, 9 show the results of the attempt. Although we can notice the proposed method is being compromised (*i.e.* Q_{wm} drops in all 3 GAN models), the performance has also worsened explicitly. However, if we ever met such condition, we can still claim the ownership by further investigate the normalization layers and retrieve the signature embedded into the weights since the signature remains intact in all sort of removal attacks (see next).

4.6. Resilience against ambiguity attacks

Through Tables 7, 8 and 9, we can observe that the embedded signature remains persistent even after removal attacks such as fine-tuning and overwriting as the BER remains 0 throughout the experiments. Thus, we can conclude that enforcing the sign in defined polarity using sign loss is

Figure 9: Image pairs from left to right is GAN_{ws} with 0% and 10% of the total signs were being randomly toggle.

rather robust against diverse adversarial attacks.

We also simulated a scenario of an insider threat where the watermark and scale signs were exposed completely. As shown in Fig. 8, it shows that the FID of DCGAN_{ws} increases drastically (from 26 $\rightarrow$ 91) and SSIM of SRGAN_{ws} drops, despite only 10% of the signs are modified. Qualitatively, Fig. 9 clearly shows the quality of the generated images is badly deteriorated when the signs are compromised. This is same for SRGAN_{ws} and CycleGAN_{ws} where the quality of the generated SR-images and (label $\rightarrow$ photo) images are very poor in quality where obvious artefact is observed even the signature signs are modified at only 10%.

In summary, we can deduce that the signs enforced in this way remain rather persistent against ambiguity attacks and attackers will not be able to employ new (modified) scale signs without compromising the GANs performance.

4.7. Ablation Study

4.7.1 Coefficient λ.

The coefficient, λ is multiplied to the reconstructive regularizing term, $\mathcal{L}_w$ to balance between the original objective and the quality of generated watermark. We perform an ablation study and from Table 10, we show that when λ is low (0.1), the FID score is at the lowest, meaning the GAN model has a very good performance in the original task. Oppositely, when λ is set to very high (10.0), the quality of the watermark, Q_{wm} is at the best, but the FID score is the lowest along the spectrum. However, qualitatively, it is hardly to visualize this. As a summary, there is a tradeoffs between GAN model performance and the watermarking quality. We find that $\lambda = 1.0$ is reasonable as the quality of watermark is relatively good without hurting the performance of the original task too much.

4.7.2 n vs. c.

This experiment investigates the effects of different n and c settings to the original DCGAN performance (measured in FID) and the quality of the generated watermark (measured in SSIM). We conclude that setting $n = 5$ and $c = -10$ perform the best (See Table 11) in terms of quality of both generated image and watermark, however, the choice can be vary depends on the situation. Notice that it performs

λ	0.1	0.5	1.0	5.0	10.0
FID	**25.88**	26.57	28.19	32.46	47.38
Q_{wm}	0.926	0.956	0.965	0.979	**0.982**

Table 10: λ vs. GAN performance measured in FID and quality of the generated watermark measured in SSIM. Image pairs from left to right is λ=0.1; λ=1.0 and λ=10.

n \ c	-10	-5	0	+5	+10
5	26.05	26.37	276.36	25.98	26.19
	0.961	0.960	0.367	0.953	0.958
10	28.35	27.49	332.88	26.18	27.11
	0.958	0.953	0.338	0.956	0.956
15	25.51	26.85	316.78	27.27	26.24
	0.954	0.945	0.343	0.951	0.953

Table 11: Effect of n and c to model's performance in terms of FID (above) and the quality of generated watermark measured in SSIM (below).

the worst when setting $c = 0$ and the performance increases when the magnitude of c increases, moving away from 0. This effect explains the reason why the *trigger* input set must have a very different distribution from the training data. For DCGAN, the training input has a normal distribution of $\mu = 0$, and setting c to 0 will not change the distribution, thus causing confusion between the normal input and *trigger* input.

5. Discussion and Conclusion

This paper illustrates a complete and robust ownership verification scheme for GANs in black-box and white-box settings. While extensive experiment results are conducted for three representative variants *i.e.* DCGAN, SRGAN and CycleGAN, the formulation lay down is generic and can be applied to any GAN variants with generator networks as the essential component. Empirical results showed that the proposed method is robust against removal and ambiguity attacks, which aim to either remove existing watermarks or embed counterfeit watermarks. It was also shown that the performance of the model's original tasks (*i.e.* image generation, super-resolution and style transfer) were not compromised. The importance of this work, in our view, can be highlighted by numerous disputes over IP infringements between giant and/or startup companies, which are now heavily investing substantial resources on developing new DNN models. We hope that the ownership verification for GANs will provide technical solutions in discouraging plagiarism and, hence, reducing wasteful lawsuit cases.

References

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1615–1631, 2018.

[2] Md Asikuzzaman and Mark R Pickering. An overview of digital video watermarking. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(9):2131–2153, 2018.

[3] Huili Chen, Bita Darvish Rohani, and Farinaz Koushanfar. DeepMarks: A Digital Fingerprinting Framework for Deep Neural Networks. *arXiv:1804.03648*, Apr. 2018.

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016.

[5] Antonia Creswell, Tom White, Vincent Dumoulin, Kailash Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35:53–65, 2018.

[6] Bita Darvish Rohani, Huili Chen, and Farinaz Koushanfar. DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models. *arXiv:1804.00750*, Apr. 2018.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[8] Yousof Erfani, Ramin Pichevar, and Jean Rouat. Audio watermarking using spikegram and a two-dictionary approach. *IEEE Transactions on Information Forensics and Security*, 12(4):840–852, 2017.

[9] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *NeurIPS*, pages 4714–4723, 2019.

[10] Han Fang, Weiming Zhang, Hang Zhou, Hao Cui, and Nenghai Yu. Screen-shooting resilient watermarking. *IEEE Transactions on Information Forensics and Security*, 14(6):1403–1418, 2019.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, pages 6626–6637, 2017.

[13] Min-Jae Hwang, JeeSok Lee, MiSuk Lee, and Hong-Goo Kang. Svd-based adaptive qim watermarking on stereo audio signals. *IEEE Transactions on Multimedia*, 20(1):45–54, 2018.

[14] Guo Jia and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2018.

[15] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[16] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020.

[17] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, pages 4681–4690, 2017.

[18] Zi-Xing Lin, Fei Peng, and Min Long. A low-distortion reversible watermarking for 2d engineering graphics based on region nesting. *IEEE Transactions on Information Forensics and Security*, 13(9):2372–2382, 2018.

[19] Hannes Mareen, Johan De Praeter, Glenn Van Wallendael, and Peter Lambert. A scalable architecture for uncompressed-domain watermarked videos. *IEEE Transactions on Information Forensics and Security*, 14(6):1432–1444, 2019.

[20] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[21] Seung-Min Mun, Seung-Hun Nam, Han-Ul Jang, Dongkyu Kim, and Heung-Kyu Lee. A robust blind watermarking using convolutional neural network. *arXiv:1704.03248*, 2017.

[22] Andrew Nadeau and Gaurav Sharma. An audio watermark designed for efficient and robust resynchronization after analog playback. *IEEE Transactions on Information Forensics and Security*, 12(6):1393–1405, 2017.

[23] Ehsan Nezhadarya, Z Jane Wang, and Rabab Kreidieh Ward. Robust image watermarking based on multiscale gradient direction quantization. *IEEE Transactions on Information Forensics and Security*, 6(4):1200–1213, 2011.

[24] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015.

[25] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 269–277, 2017.

[26] Vedran Vukoti, Vivien Chappelier, and Teddy Furon. Are deep neural networks good for blind image watermarking? In *International Workshop on Information Forensics and Security (WIFS)*, pages 1–7, 2018.

[27] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[28] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[29] Jie Zhang, Dongdong Chen, Jing Liao, Weiming Zhang, Gang Hua, and Nenghai Yu. Passport-aware normalization for deep model protection. In *NeurIPS*, 2020.

[30] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS)*, pages 159–172, 2018.

[31] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *ECCV*, pages 682–697, 2018.

[32] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[33] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017.