Generalized Adversarial Overwriting Attack on Deep Learning-Based Watermarking Under Restricted Black-Box Access

Sudev Kumar Padhi
Department of CSE
Indian Institute of Technology Bhilai, Durg, India
Email: sudevp@iitbhilai.ac.in

Sk. Subidh Ali
Department of CSE
Indian Institute of Technology Bhilai, Durg, India
Email: subidh@iitbhilai.ac.in

***Abstract*—The increasing sophistication of AI-driven image manipulation and generation poses significant challenges to copyright protection and ownership verification. Deep learning-based watermarking techniques have emerged as a robust solution to embed imperceptible yet resilient watermarks within images. However, these techniques remain vulnerable to adversarial attacks, threatening their reliability in digital forensics and intellectual property protection. This work introduces a generalizable adversarial overwriting attack under restricted black-box access that systematically replaces an existing watermark with an attacker-specified one, effectively compromising ownership claims. Our technique is the first to successfully attack the decoders without any access (not even Oracle access). We leverage a surrogate model attack in a restricted black-box setting under no access to the decoder to generate imperceptible adversarial perturbations, forcing watermarking techniques to extract an attacker-specified watermark. Evaluated across several watermarking methods, our technique effectively overwrites watermarks while preserving image quality, making detection difficult. Our findings expose critical vulnerabilities in deep learning-based watermarking, highlighting the need for more resilient frameworks to safeguard copyright protection and prevent unauthorized ownership claims.**

I. Introduction

The increasing use of AI-driven image manipulation and generation has blurred the boundary between authentic and altered content, raising concerns about misinformation, unauthorized image modifications, and data privacy. The ability of generative models to generate images from scratch and significantly manipulate images has introduced severe risks, particularly in copyright enforcement and content authentication. Image watermarking techniques have emerged as a fundamental technique for mitigating these risks. The most popular watermarking techniques [1], [2], [3] embed a robust and imperceptible watermark in a post-processing manner within existing images (cover images) without distorting the quality of the cover image. The embedded watermark can later be extracted and compared with the original reference watermark to authenticate ownership and verify copyright protection. Given the increasing sophistication of AI-driven techniques, watermarking has become a proactive measure for digital forensics and copyright protection, safeguarding intellectual property rights and preventing unauthorized alterations or fraudulent claims.

Recently, deep learning has emerged as the key enabler of AI applications. In Deep Neural Network (DNN) based watermarking techniques, watermark embedding and extraction are performed using deep generative models, such as autoencoders and Generative Adversarial Networks (GANs). A core security property of watermarking is robustness, which states that an attacker can corrupt the watermark by substantially degrading the image's quality. Despite notable advancements in enhancing the robustness of DNN-based watermarking by incorporating distortions into the training process [1], [4], [3], [5]. They are still susceptible to Deep Learning-based Removal (DLR) attacks [6], which pose significant risks to copyright protection by enabling unauthorized watermark erasure. DLR attacks mostly employ techniques such as denoising autoencoders to treat watermarks as noise [7], pixel distribution analysis to identify and remove distorted regions [6], and pixel inpainting to seamlessly erase embedded watermarks [8]. DLR attacks serve a limited purpose, primarily aiming to invalidate the rightful owner's ownership claim over a watermarked image. However, these attacks do not enable the attacker to claim ownership of the image. To achieve this, the attacker must overwrite the original watermark in the watermarked image with their own, ensuring that the watermark extraction process retrieves the attacker's watermark instead of the original one.

We introduce a generalizable adversarial overwriting attack under restricted black-box access, where the attacker has access only to the encoder but not the decoder or extraction process. Unlike conventional DLR attacks that erase watermarks, our technique forcibly replaces the original watermark with an attacker-specified one, compromising the integrity of deep learning-based watermarking techniques. Our technique is inspired by adversarial machine learning (AML) [9], where well-crafted perturbations manipulate a model's decision boundary to induce misclassification. We apply this principle to watermarking techniques by generating imperceptible adversarial perturbations that subtly modify the watermarked image such that, when passed through the

watermarking pipeline, the extracted watermark aligns with the attacker's target watermark instead of the original one. Since the attacker has no access to the decoder or watermark extraction process, our technique leverages a surrogate model attack that approximates only the encoder to generate adversarial perturbations. This enables a black-box attack that remains effective across multiple watermarking methods, even when the underlying watermarking model is unknown. This paper makes the following key contributions:

1) We propose an adversarial overwriting attack that replaces the original watermark with an attacker-specified one.
2) We develop a generalizable attack under restricted black-box access, where the attacker has access only to the encoder and not the decoder.
3) We evaluate our technique across different watermarking methods to assess its adaptability and effectiveness.

II. Related Works

Recently, many DNN-based watermarking techniques have been proposed, which surpass the performance of traditional watermarking by utilizing the efficient feature extraction ability of the neural networks. The main architecture used in DNN-based watermarking involves using an encoder network that embeds the watermark into the cover image and a decoder network that extracts the watermark from the watermarked image. DNN-based watermarking can embed an image or bit string as a watermark. However, it is mostly used to embed bit strings. Bit strings work as metadata and provide more robustness than embedding images as a watermark. This is due to the fact that embedding an image requires the decoder to learn the spatial information of the watermark, which can hamper robustness. Generally, the encoder and decoder's training is done end-to-end as a pipeline [1], [4]. To enhance the quality and robustness of the watermarked image, a discriminator is added in the pipeline while training and noise layers are added in the model architecture of the DNN-based watermarking [3], [5], [2]. The discriminator predicts whether a watermark is embedded, while residual connections and randomized distortion layers improve robustness and data-hiding capacity [10], [11]. Though these methods achieve high image quality and withstand distortions like $JPEG$ compression, blurring, and cropping, they remain vulnerable to adversarial attacks.

III. Proposed Approach

DNN-based watermarking techniques consist of an encoder for embedding the watermark into the cover image and a decoder to decode or extract the watermark from the cover image. The encoder E takes the cover image (I) and the watermark (α) as inputs and produces a watermarked image W by embedding the watermark α into the cover image I as shown in Eq. (1). In contrast, the decoder D takes W as input and extracts the embedded watermark α as the output, as shown in Eq. (2). The attacker's aim is to launch the attack to fool D by inducing adversarial perturbation δ in W such that D decodes the target watermark β instead of the original embedded watermark α as shown in Eq. (3).

$$E(I + \alpha) \rightarrow W \tag{1}$$
$$D(W) \rightarrow \alpha \tag{2}$$
$$D(W + \delta) \rightarrow \beta \tag{3}$$

1) White-Box Access: Having white-box access to the decoder gives the attacker enough information to simulate the network by devising a targeted adversarial attack and using the gradients of the decoder to create the desired perturbation δ, where α is the original watermark, β is the target watermark and ϵ is the perturbation limit. We minimize the loss (ℓ) of β, which is the target watermark while maximizing the loss of α, which is the original watermark, $i.e.$, we solve the optimization problem as shown in Eq. (4). This is the easiest approach but does not align with the use cases of watermarking, where access to the decoder is not allowed.

$$\min_{\delta} \ \ell(D(W+\delta), \beta) - \ell(D(W+\delta), \alpha), \quad \delta \in [-\epsilon, \epsilon] \tag{4}$$

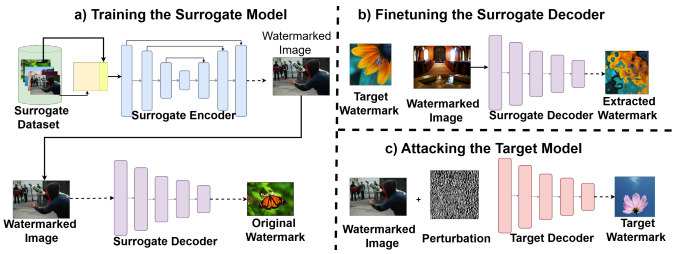

Fig. 1. Overview of surrogate model attack **a)** Training the surrogate model using surrogate dataset **b)** Fine-tuning the surrogate decoder with the watermarked image of the target DNN-based watermarking technique **c)** Attacking the decoder of the target DNN-based watermarking technique after generating the well-crafted perturbation from the surrogate decoder.

2) Black-Box Access: The Watermarking technique is available as a service [12], [13], [14]. The attacker can subscribe to the service and get Oracle access to both the encoder and decoder through its API and it can obtain a set of watermarked images and their watermarks by querying the decoder with watermarked images. Once this data set is available, the attacker can train a surrogate decoder. Afterwards, a white-box attack is performed on the surrogate decoder to craft the desired perturbation δ, which is used to launch the final attack on D. However, in stringent security applications, even the decoder is not available and it is only present with the verifier. Therefore, we consider only having limited instances of watermarked images whose watermarks are known. One of the easiest ways for the attacker to gain access to such data is to request the subscribed copyright protection service provider to copyright on, say, n pairs of cover images and their watermarks.

Under this scenario, the attacker will train its own DNN-based surrogate watermarking encoder (E') and decoder (D')

models with its own dataset (also known as the surrogate dataset) $i.e.$, a set of cover images and their watermarks. Once the surrogate model is trained, D' is fine-tuned with the limited instances of watermarked images available from the target decoder D to be attacked. While fine-tuning, loss between the extracted watermark and the original watermark is used for the training of the surrogate decoder, as shown in Eq (5). We emphasize that functional equivalence between the surrogate encoder E' and the target encoder E is *not required* for the success of the proposed attack. The surrogate encoder is used solely to generate valid watermarked samples during surrogate model training. Transferability is enforced entirely at the decoder level by fine-tuning the surrogate decoder D' on watermarked images produced by the target encoder. The surrogate decoder is trained and fine-tuned to act as the target decoder D, making the restricted black-box attack transferable. Therefore, the attacker can launch a white-box attack on D' using the gradient information to craft the desired perturbation δ as shown in Eq. (6). The same δ can be used to fail D when added with W (Eq. (7)). The value ϵ is chosen judiciously such that the induced perturbation (δ) to the watermarked image is imperceptible. Figure 1 refers to the training procedure of the surrogate model and fine-tuning of the surrogate decoder to perform an attack on the target decoder.

$$minimize \ \ell(D'(W), \alpha) \tag{5}$$

$$\underset{\delta}{minimize} \ \ell(D'(W+\delta), \beta), \quad \delta \in [-\epsilon, \epsilon] \tag{6}$$

$$D(W+\delta) \rightarrow \beta \tag{7}$$

A. Crafting algorithm

The adversarial perturbation crafting algorithm is shown in Algo 1. The algorithm shows how to craft a perturbation using our technique in a white box attack scenario. Inputs to the algorithm are, a watermarked image W, the target decoder D, the target watermark β, a perturbation δ (initialized as zero) with the same size ($k \times k$) as W, a limiting range ϵ of δ (-ϵ $\leq \delta \leq \epsilon$). δ is added with W and passed into the decoder D, which decodes the secret as γ. The loss between γ and β is computed using the chosen loss function ℓ. In each iteration of the loop, the optimizer tries to minimize the loss between γ and β and maximize the loss between β and α. Accordingly, Δ is updated. This process is repeated until the model converges and the desired δ is obtained, which is the realization of the attack on W to overwrite α with β.

The hyperparameters (parameters that we explicitly define) for this attack include:

1) ϵ: The maximum amount of allowable perturbation.
2) Optimizer: Used to find the well-crafted perturbation δ.
3) ℓ: The loss function for minimizing the loss between γ and β and maximizing the loss between β and α.

This algorithm works similarly in the restricted black-box scenario where the decoder D is replaced by a surrogate decoder D' in line 3 and 5, and the corresponding loss will be $\ell(\beta, \gamma)$ + $\ell(D'(W), \alpha)$ in line 6 of the algorithm.

Algorithm 1 Adversarial Perturbation Crafting

Require: W, D, β, δ, ϵ

1: $\delta \leftarrow [0]_{k \times k}$ ▷ Initial perturbation
2: max_iter=$k \times k$
3: γ = $D(W+\delta)$ ▷ Embedded Watermark
4: **while** $i \leq max\_iter$ & $\gamma \neq \beta$ **do**
5: $\gamma \leftarrow D(W+\delta)$ ▷ Intermediate decoder's output
6: $\Delta \leftarrow \ell(\beta, \gamma) - \ell(\beta, \alpha)$
7: Update $\delta : \delta_i \leftarrow \delta_{i-1} - \eta \nabla \Delta(\delta)$ ▷ Update delta with respect to the loss
8: $\delta \leftarrow Clip(\delta, [-\epsilon, \epsilon])$ ▷ δ is clipped
9: **end while**
10: return δ

B. Reason For Successful Attack

As we know in the classification task, whatever the input may be, the classifier will always classify the input into one of the classes. These classes are also known to the attacker while performing a targeted adversarial attack. Thus, the attacker adds perturbation such that the boundary of the current class is crossed to the target class and the confidence level of the classifier corresponding to the target class is the highest. Suppose the watermark is of N-bit, $i.e.$, the output of the decoder is a N-bit watermark. Now, we can consider the decoder as a classifier that classifies the watermarked image into one of 2^N possible watermark classes. Therefore, our technique can be considered a targeted adversarial attack where the target class is the target watermark among one of the 2^N possible cases. DNN-based watermarking techniques are trained such that the perceptual similarity of the cover image remains unaltered after embedding the watermark, which makes the embedding process region-specific and susceptible to attack. Even if the models are trained for robustness against prominent image manipulation attacks, the same factor is responsible for attack.

IV. Experimental Results

We validate the effectiveness of the attack on nine well-known DNN-based watermarking techniques. We have evaluated the accuracy of these models by watermarking 1000 images using each of the techniques and extracting the watermark from the corresponding watermarked image.

A. Training Surrogate Model

Each technique we have taken for evaluation uses different cover image resolutions and watermark sizes. The first eight techniques use bit-strings, while the last one uses a $200 \times 200 \times 3$ image as a watermark. We target both techniques to demonstrate our technique's generalizability. To show the transferability (architecture level and data level) of our technique, we have considered multiple target watermarking techniques where bit-string is used as the watermark and training one surrogate model can attack all the target watermarking techniques. We have used $UNet$ architecture for the surrogate encoder throughout our evaluation. Whereas

for the surrogate decoder, we have used two different architectures: one is spatial transformer [15] with seven convolutional layers followed by two fully connected layers and the other is self-supervised vision transformer (SiT) [16] based autoencoder. The first one is used to build the surrogate decoder for $HiDDeN$ [3], $ReDMark$ [10], $PIMoG$ [5], $Stegastamp$ [2], $Aparecium$ [17], Tree-Ring [18], Stable Signature [19], and $DADW$ [11], where a bit-string is used as the watermark. The second architecture is used to build the surrogate decoder for *Hiding Images in an Image* [1], where an image is used as the watermark. We have used the $Mirflickr$, $COCO$, and $Open\ Images$ as our surrogate datasets for training our model. Once the surrogate encoder's and decoder's architecture is fixed, we use two different approaches to train the surrogate model. In the first approach, we train surrogate models according to the target watermarking techniques, whereas in the second approach, we train one surrogate model and test it on all target watermarking techniques.

Approach 1: Training different instances of surrogate model for each technique: We have trained nine surrogate models, $i.e.$, one model for each target watermarking technique. The size of the cover image, watermarked image, and watermark for each surrogate model is set according to the target watermarking technique to which it corresponds. We are using three different datasets $i.e.$, $COCO$, $MIRFLICKR$, and $Open\ Image$ and each of the nine surrogate models is individually trained on each of the $three$ datasets. This makes a total of 27 instance of surrogate model $i.e.$, three instances for each target watermarking technique corresponding to three datasets as shown in Table I. We used 50k images in our training set and 10k in our test set to train every instance of the surrogate models. All the 27 surrogate model instances are trained in an end-to-end manner for *200* epochs, which is the general approach followed in DNN-based watermarking techniques. We used MSE, $LPIPS$, and L_2 residual regularization loss functions between the cover image and the watermarked image for training the surrogate encoders. While training surrogate decoders for $HiDDeN$, $ReDMark$, $PIMoG$, $Stegastamp$, $Aparecium$, Tree-Ring, Stable Signature, and $DADW$, where a bit-string is used as the watermark, we used binary cross entropy (BCE) to calculate the loss between the extracted and the original watermarks. In the same line, MSE and $LPIPS$ loss are used to train the surrogate decoder for [1].

Approach 2: Training a single Surrogate Model: In this setup, the objective is to figure out a common surrogate model (we only consider cases where a bit string is embedded as a watermark) that can be used to attack all the eight watermarking technique. We have considered 224×224 as the size of the cover images, and the length of the bit string used are 50-bits, 100-bits, and 200-bits. We have trained three instances of the surrogate model where the watermark of size 50-bits is trained on $COCO$ dataset, the watermark of size 100-bits is trained on $MIRFLICKR$ dataset, and the watermark of size 200-bits is trained on $Open\ Image$ dataset. We used 50k images in our training set and 10k in our test set. We trained the surrogate model in an end-to-end manner for *200* epochs. The same MSE, $LPIPS$, and L_2 residual regularization loss functions are used between the cover and watermarked images for training the surrogate encoders. While training the surrogate decoders, we used BCE to calculate the loss between the extracted and the original watermarks.

B. Fine-tuning Surrogate Decoder

We will next show that our technique (Algo 1) can successfully adapt to different watermarking techniques through experiments. Before going into the details of our results, we discuss our fine-tuning setup. For fine-tuning, we collected a maximum of 2000 watermarked images and their corresponding watermarks from each of the nine watermarking techniques and fine-tuned for a maximum of 100 epochs. The optimal number of watermarked images and epochs required per technique is shown in Table I and Table II. In the case of $HiDDeN$, $ReDMark$, $PIMoG$, $Stegastamp$, $Aparecium$, Tree-Ring, Stable Signature, and $DADW$, each watermarked image is embedded with a unique watermark generated from randomly sampled bits, whereas for *Hiding Images in an Image*, we use randomly sampled images from the $ImageNet$ dataset as the watermark.

Fine-tuning in Approach 1: The 27 instances of the trained surrogate decoder are fine-tuned on 2000 watermarked images of the respective target technique. Fine-tuning the surrogate decoder for 100 epochs is sufficient to attack the decoder of the target watermarking techniques successfully.

Fine-tuning in Approach 2: Before fine-tuning, watermarked images from target techniques are resized to 224×224 to match the surrogate decoder's input. Subsequently, the number of nodes in the last fully connected layer is replaced as per the length of the bit-string embedded in the target watermarking technique. For example, the first instance of the three surrogate model instances uses 224×224 size cover image and 50-bit watermark. In order to make it transferable to $ReDMark$, we need to fine-tune this surrogate model instance using 2000 watermarked images from $ReDMark$. Please note that $RedMark$ uses $32 \times 32 \times 3$ size cover image and 16-bit watermark. Therefore, the 2000 watermarked images from $ReDMark$ have to be upsampled to 224×224, and the last fully connected layer of the surrogate decoder is modified to 16 nodes to extract the 16-bit watermark embedded in the watermarked image of $ReDMark$. Similarly, each instance of the three surrogate models is fine-tuned with all the eight target watermarking techniques for 100 epoch.

C. Attack Validation

In order to validate our technique, we generate 10000 watermarked images using each of the nine target watermarking techniques. Then, these images are given to their corresponding fine-tuned surrogate decoders. Algo 1 is used to generate corresponding well-crafted perturbations by forcing the surrogate decoder to extract the target watermark as an output. These perturbations are added to the watermarked images and are used to attack the target decoder to evaluate the success rate of our technique. We tried our technique

TABLE I
PERFORMANCE OF OUR TECHNIQUE USING DIFFERENT SURROGATE MODELS FOR EACH WATERMARKING TECHNIQUE.

Technique	Surrogate Model		Fine-Tuning		Pert. Limit (ϵ)	Evaluation Metrics				
	Dataset	Acc. (%)	Epoch	Images		PSNR	SSIM	LPIPS	MSE	ASR (%)
ReDMark	COCO	96	40	600	0.009	41	0.98	0.08	0.05	97
	Mirflickr	98	40	700	0.008	40	0.97	0.07	0.06	96
	Open Images	97	40	600	0.008	40	0.98	0.07	0.05	98
HiDDeN	COCO	98	60	900	0.010	38	0.99	0.08	0.15	97
	Mirflickr	96	60	900	0.010	37	0.99	0.09	0.17	95
	Open Images	96	60	1000	0.009	37	0.99	0.09	0.15	95
PIMoG	COCO	96	80	1400	0.040	37	0.99	0.10	0.27	96
	Mirflickr	96	80	1600	0.030	37	0.99	0.10	0.31	93
	Open Images	96	80	1600	0.040	37	0.99	0.10	0.30	93
Stegastamp	COCO	98	70	1300	0.010	34	0.99	0.10	0.25	94
	Mirflickr	98	70	1200	0.030	35	0.99	0.09	0.22	96
	Open Images	98	70	1300	0.010	34	0.99	0.10	0.24	94
Aparecium	COCO	98	70	1500	0.030	40	0.99	0.08	0.19	96
	Mirflickr	97	70	1700	0.040	39	0.99	0.10	0.22	94
	Open Images	97	70	1700	0.010	39	0.99	0.10	0.21	94
DADW	COCO	98	60	1200	0.009	38	0.99	0.10	0.31	93
	Mirflickr	98	60	1400	0.009	38	0.99	0.10	0.29	92
	Open Images	98	60	1500	0.008	38	0.99	0.10	0.29	92
Tree Ring	COCO	96	100	2000	0.010	34	0.99	0.10	0.28	94
	Mirflickr	95	100	2000	0.020	34	0.99	0.10	0.23	94
	Open Images	96	100	2000	0.010	33	0.99	0.10	0.26	96
Stable Signature	COCO	95	100	2000	0.020	35	0.99	0.09	0.27	95
	Mirflickr	95	100	2000	0.030	34	0.99	0.10	0.29	96
	Open Images	96	100	2000	0.010	34	0.99	0.09	0.25	96
Hiding Img.	COCO	93	100	2000	0.080	32	0.95	0.19	0.41	91
	Mirflickr	92	100	2000	0.090	33	0.94	0.22	0.36	91
	Open Images	93	100	2000	0.090	32	0.95	0.21	0.37	91

on the surrogate decoder with taking loss l as MSE loss (Line 06, Algo 1 and Equation 4). The initial perturbation δ is initialized as a zero-filled vector, whereas ϵ is chosen as $-0.1 \leq \epsilon \leq 0.1$. Table I and Table II shows the performance of our technique with *Approach 1 and 2*. We employ the $Adam$ optimizer with an initial learning rate of 0.001. For all the target watermarking techniques, the attack converges within 5000 iterations (Algo 1).

We evaluate the quality and similarity of attacked watermarked images using MSE, $PSNR$, $SSIM$, and $LPIPS$ (from $AlexNet$), along with visual analysis. A high-quality attack disrupts the target decoder while introducing no visible artifacts in the image. MSE and $PSNR$ measure pixel-level differences between the attacked and original watermarked images, while $SSIM$ and $LPIPS$ assess perceptual similarity and ensure minimal visual degradation. Higher $PSNR$ and $SSIM$, along with lower MSE and $LPIPS$, indicate better image quality. Additionally, we use Attack Success Rate (ASR) as the final evaluation metric, which represents the percentage of adversarial examples that successfully fool the target decoder into extracting the target watermark. In our experiments, ASR is computed over 10000 watermarked images for each watermarking technique.

D. Evaluation

Evaluation for Approach 1: Table I lists the optimal epoch, number of watermarked images (4^{th}–5^{th} columns), and optimal ϵ (6^{th} column). The last five columns report performance across watermarking techniques. High $PSNR$ and $SSIM$, along with low $LPIPS$ and MSE, indicate minimal perturbation. The high ASR further demonstrates the generalizability of our method across different watermarking techniques, image sizes, and watermark types.

Evaluation for Approach 2: Table II presents the performance of our technique across three instances of each surrogate model using $Approach$ 2. The optimal epoch and number of watermarked images are listed in the 2^{nd} and 3^{rd} columns, while the optimal ϵ appears in the 4^{th} column. The last fifteen columns report attack results on different watermarking techniques. High $PSNR$ and $SSIM$, along with low $LPIPS$

TABLE II
PERFORMANCE OF OUR TECHNIQUE WHERE A SINGLE SURROGATE MODEL IS USED TO ATTACK DIFFERENT WATERMARKING TECHNIQUES.

Technique	Finetuning		Pert Limit (ϵ)	50-bit Watermark					100-bit Watermark					200-bit Watermark				
	Epoch	Images		PSNR	SSIM	LPIPS	MSE	ASR	PSNR	SSIM	LPIPS	MSE	ASR	PSNR	SSIM	LPIPS	MSE	ASR
ReDMark	50	1200	0.009	40	0.97	0.08	0.05	89	41	0.97	0.09	0.07	88	41	0.97	0.10	0.09	89
HiDDeN	70	1600	0.020	37	0.99	0.10	0.13	86	37	0.99	0.10	0.11	86	38	0.99	0.10	0.14	88
PIMoG	70	1800	0.050	37	0.99	0.10	0.29	85	38	0.99	0.15	0.34	83	38	0.99	0.20	0.32	85
Stegastamp	80	1800	0.040	34	0.99	0.09	0.23	85	35	0.99	0.10	0.20	86	34	0.99	0.15	0.22	86
Aparecium	100	2000	0.040	38	0.99	0.10	0.20	86	38	0.99	0.10	0.23	86	37	0.99	0.15	0.21	87
DADW	70	1400	0.030	38	0.99	0.08	0.11	86	38	0.99	0.09	0.11	89	38	0.99	0.09	0.08	88
Tree Ring	100	2000	0.040	34	0.99	0.09	0.23	84	34	0.99	0.09	0.24	86	33	0.99	0.12	0.26	87
Stable Signature	100	2000	0.040	33	0.99	0.10	0.23	85	33	0.99	0.11	0.23	86	33	0.99	0.11	0.24	88

and MSE, indicate imperceptible perturbations. The high ASR further demonstrates transferability, as a single surrogate model can attack eight watermarking techniques with minimal fine-tuning of the final layer.

Low ϵ in both approaches ensures imperceptible noise, reflected by high $PSNR$ and $SSIM$ and low $LPIPS$ and MSE. Although $Approach$ 1 achieves slightly higher ASR, $Approach$ 2 is more practical as it avoids training multiple models. Figure 2 compares image quality before and after the attack, while Figure 3 shows the extracted watermark quality. The attack fails when cosine similarity is below 0.2; achieving 100% success in such cases requires increasing ϵ to 0.5, which degrades visual quality.

Fig. 2. Image quality after addition of well-crafted imperceptible perturbation.

Fig. 3. Attacking the watermarked image created by *Hiding Images in Image*.

V. CONCLUSION

This work proposes a novel adversarial overwriting attack under restricted black-box access to DNN-based watermarking techniques, leveraging adversarial machine learning. The proposed attack shows that existing DNN-based watermarking techniques, which are done in a post-processing manner, are critically vulnerable to this attack. The DNN-based watermarking techniques are trained using different datasets, architecture, and types of watermarks to create watermarked images. Still, the process of extracting the watermark follows a similar approach that captures the common features. The surrogate decoders learn these common features to replicate the behaviour of the decoder of the target watermarking technique and make the attack possible. Extensive experimentation using different approaches is performed across diverse DNN-based watermarking techniques to validate the attack.

REFERENCES

[1] S. Baluja, "Hiding images within images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 7, pp. 1685–1697, 2019.
[2] M. Tancik, B. Mildenhall, and R. Ng, "Stegastamp: Invisible hyperlinks in physical photographs," in *IEEE/CVF CVPR*, pp. 2117–2126, 2020.
[3] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, "Hidden: Hiding data with deep networks," in *ECCV*, pp. 657–672, 2018.
[4] H. Kandi, D. Mishra, and S. R. S. Gorthi, "Exploring the learning capabilities of convolutional neural networks for robust image watermarking," *Computers & Security*, vol. 65, pp. 247–268, 2017.
[5] H. Fang, Z. Jia, Z. Ma, E.-C. Chang, and W. Zhang, "Pimog: An effective screen-shooting noise-layer simulation for deep-learning-based watermarking network," in *ACM MM*, pp. 2267–2275, 2022.
[6] D. Jung, H. Bae, H.-S. Choi, and S. Yoon, "Pixelsteganalysis: Pixel-wise hidden information removal with low visual degradation," *IEEE Transactions on Dependable and Secure Computing*, 2021.
[7] I. Corley, J. Lwowski, and J. Hoffman, "Destruction of image steganography using generative adversarial networks," *arXiv:1912.10070*, 2019.
[8] H. Liu, T. Xiang, S. Guo, H. Li, T. Zhang, and X. Liao, "Erase and repair: An efficient box-free removal attack on high-capacity deep hiding," *IEEE Transactions on Information Forensics and Security*, 2023.
[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv:1312.6199*, 2013.
[10] M. Ahmadi, A. Norouzi, N. Karimi, S. Samavi, and A. Emami, "Redmark: Framework for residual diffusion watermarking based on deep networks," *Expert Systems with Applications*, vol. 146, p. 113157, 2020.
[11] X. Luo, R. Zhan, H. Chang, F. Yang, and P. Milanfar, "Distortion agnostic deep watermarking," in *IEEE/CVF CVPR*, p. 13548, 2020.
[12] "Meta on ai-generated photos." https://about.fb.com/news/2024/02/labeling-ai-generated-images-on-facebook-instagram-and-threads/. Accessed: 2022-20-02.
[13] "Google on watermarking ai-generated contents." https://deepmind.google/technologies/synthid/, 2022. Accessed: 2022-20-02.
[14] "Adobe on watermarking ai-generated photos." https://blog.adobe.com/en/publish/2023/10/10/new-content-credentials-icon-transparency, 2022. Accessed: 2022-20-02.
[15] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," *NIPS*, vol. 28, 2015.
[16] S. Atito, M. Awais, and J. Kittler, "Sit: Self-supervised vision transformer," *arXiv:2104.03602*, 2021.
[17] Z. Lei, J. Zhang, J. Li, W. Zhang, and N. Yu, "Aparecium: Revealing secrets from physical photographs," *arXiv:2308.12141*, 2023.
[18] Y. Wen, J. Kirchenbauer, J. Geiping, and T. Goldstein, "Tree-rings watermarks: Invisible fingerprints for diffusion images," *NIPS*, vol. 36, pp. 58047–58063, 2023.
[19] P. Fernandez, G. Couairon, H. Jégou, M. Douze, and T. Furon, "The stable signature: Rooting watermarks in latent diffusion models," in *IEEE/CVF ICCV*, pp. 22466–22477, 2023.