Pluggable Watermarking of Deepfake Models for Deepfake Detection

Han Bao[1], **Xuhong Zhang**[1*], **Qinying Wang**[2], **Kangming Liang**[3], **Zonghui Wang**[2*], **Shouling Ji**[2] and **Wenzhi Chen**[2]

[1]School of Software Technology, Zhejiang University.
[2]College of Computer Science, Zhejiang University
[3]College of Engineering, Zhejiang University.
{baohan21, zhangxuhong, wangqinying, liangkm, zhwang, sji, chenwz}@zju.edu.cn

Abstract

Deepfake model misuse poses major security concerns. Existing passive and active Deepfake detection methods both suffer from a lack of generalizability and robustness. In this study, we propose a pluggable and efficient active model watermarking framework for Deepfake detection. This approach facilitates the embedding of identification watermarks across a variety of Deepfake generation models, enabling authorities to extract them easily for detection purposes. Specifically, our method leverages the universal convolutional structure in generative model decoders. It employs convolutional kernel sparsification for adaptive watermark embedding positioning and introduces convolutional kernel normalization to seamlessly integrate watermark parameters with those of the generative model. For watermark extraction, we jointly train a watermark extractor based on a Deepfake detection model and use BCH encoding to identify watermark images effectively. Finally, we apply our approach to eight major types of Deepfake generation models. Experiments show our method successfully detects Deepfakes with an average accuracy exceeding 94% even in heavy lossy channels. This approach operates independently of the generation model's training without affecting the original model's performance. Furthermore, our model requires training a very limited number of parameters and is resilient against three major adaptive attacks. The source code can be found at https://github.com/GuaiZao/Pluggable-Watermarking

1 Introduction

Deepfake technology has seen remarkable progress, the face of one individual can be flawlessly swapped with another from a distinct photograph. The potential misuse and rampant proliferation of such Deepfakes have raised significant societal concerns and have imperiled the credibility of online media[Li *et al.*, 2022]. Consequently, there's an urgency for face forgery detection techniques.

Primary Deepfake detection methods [Dong *et al.*, 2022; Schwarcz and Chellappa, 2021; Li *et al.*, 2023] typically identify Deepfake images by extracting biological features and high-frequency artifacts within the image in a passive way. However, when Deepfake images become more realistic or undergo transmission via lossy channels, passive detection may fail to detect them and thus have limited performance. Furthermore, such passive detection methods lack adaptability across various Deepfake models since they heavily rely on training data and only show good detection results for the Deepfake methods encountered during training [Shiohara and Yamasaki, 2022]. Active watermarking has recently emerged as a promising alternative. Such techniques introduce unique markers or signals into the synthesized images, which can then be extracted to identify images. However, existing active watermarking methods primarily focus on using watermark content to trace users linked to specific generative models and are unsuitable for direct application in Deepfake detection. Several active watermarking methods [Wu *et al.*, 2020; Zhang *et al.*, 2019] add watermarks either within the training dataset or in post-generation output images. Such methods introduce additional operational steps or independent networks, compromising the stealth of the watermark, which causes the watermark to be easily removed [Fernandez *et al.*, 2023]. On the other hand, some methods are designed for specific models, demanding a particular structure in the target generative model [Uchida *et al.*, 2017; Yang *et al.*, 2022; Zhang *et al.*, 2020; Yu *et al.*, 2021b]. Furthermore, existing watermarking methods often require retraining or fine-tuning of the original network when embedding watermarks. In some cases, they even need to perform inference or training on individual samples. Given the need to detect the Deepfake images generated by a vast array of models, the computational cost is unacceptable. In conclusion, utilizing a watermarking method of Deepfake models to detect Deepfake images remains a pressing and unresolved problem.

In this work, we propose a pluggable active watermarking framework that can simultaneously address the above challenges for Deepfake detection. To support different types of Deepfake models, we observe that mainstream generative models usually employ a form of convolution coupled with upsampling in decoders to produce the final image from the latent vector, despite the differences in latent vector structures. Based on this observation, we first introduce a wa-

[*]Co-corresponding authors

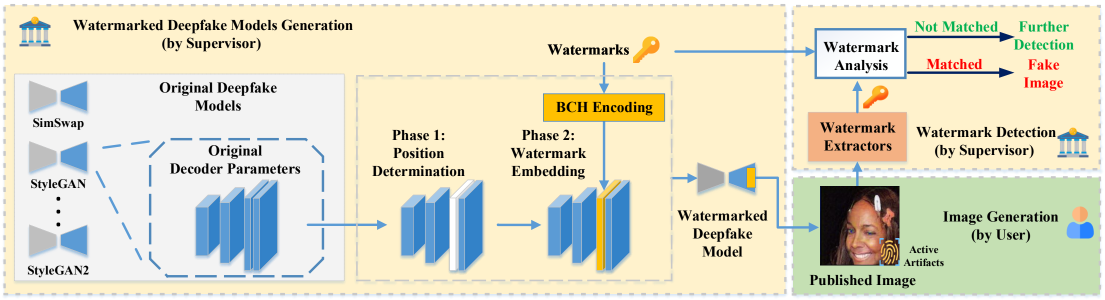

Watermarked Deepfake Models Generation (by Supervisor)
Original Deepfake Models
SimSwap
StyleGAN
StyleGAN2
Original Decoder Parameters
Watermarks
BCH Encoding
Phase 1: Position Determination
Phase 2: Watermark Embedding
Watermarked Deepfake Model
Watermark Analysis
Not Matched
Further Detection
Matched
Fake Image
Watermark Extractors
Watermark Detection (by Supervisor)
Image Generation (by User)
Active Artifacts
Published Image

Figure 1: The pipeline of our framework.

termarking technique that manipulates the parameters within the convolutional layer in the decoder. Our framework adaptively identifies redundant parameter positions via pruning the parameters of pre-trained models and sparsifying the convolution kernels. Watermark parameters are then embedded into these identified positions. This design ensures the quality of generation while remaining independent of the original model's training method. Given the diverse convolutional kernel parameters, we design a normalization method to embed parameter values, thereby guaranteeing stability across various models and preventing gradient explosions. To address efficiency, our framework solely focuses on training the substituted parameters. In addition, we incorporate Bose–Chaudhuri–Hocquenghem (BCH) codes to improve the accuracy and robustness of the watermark extraction.

In summary, 1) we propose a novel active model watermarking framework for Deepfake detection. Our framework is pluggable, operating independently of the original training of the generative model; 2) we design convolutional kernel sparsification and kernel normalization to embed watermarks covertly without altering the original network performance; 3) our proposed framework is scalable, e.g., it can be effectively deployed across eight major types of Deepfake generative models; 4) experimental results show our method's ability to robustly embed and extract watermarks for Deepfake detection while keeping the image quality.

2 Related Work

2.1 Deepfake Methods

Common Deepfake generation techniques include image synthesis, face identity swap, facial reenactment, 3D modeling, etc [Zhang, 2022; Dagar and Vishwakarma, 2022]. For instance, StyleGAN [Karras *et al.*, 2019], StyleGAN2 [Karras *et al.*, 2020], are comprehensive generative models. These models manipulate the generation of facial images with varying styles, guided by the provided latent vectors. Stable Diffusion Model (SDM) [Rombach *et al.*, 2022] modifies input image content by prompt information. For face identity swap, SimSwap [Chen *et al.*, 2020] and FaceShifter [Li *et al.*, 2019] separate attributes and arbitrary features and swap them from source to target face. HifiFace [Wang *et al.*, 2021] can generate a 3D shape-aware identity to control the face shape with the geometric supervision from the 3D face reconstruction method. For facial reenactment, FirstOrder [Siarohin *et al.*, 2019] uses a representation consisting of a set of learned key points along with their local affine transformations to change the facial attitude. LATS [Or-El *et al.*, 2020] is a method of modification of facial attributes that is based on StyleGAN2. Despite the varied architectures and loss functions of these models, our watermarking technique can effectively embed and extract watermarks from each model.

2.2 Watermarks for Deepfake

Watermarks have multiple applications in defending against Deepfake. For source samples of Deepfake, some models [Wang *et al.*, 2022; Ruiz *et al.*, 2020; Huang *et al.*, 2022] utilize adversarial watermarking to disrupt the generative ability, with the aim of protecting the input samples. [Yu *et al.*, 2021a] preserves the watermark in the training data into the generated results through joint training. For output samples of Deepfake, carrier watermarking RivaGAN [Zhang *et al.*, 2019] has already been applied to SDM. For model watermarking, [Fernandez *et al.*, 2023] uses a fine-tuned decoder to set watermarking in SDM. [Nie *et al.*, 2023; Kim *et al.*, 2020] incorporate the watermark into the latent vectors, endowing them with semantic features to maintain the quality of the model's generative output. However, these watermarked models are effective only for specific networks or require specific structures, rendering them unsuitable for the aforementioned Deepfake methods. In contrast, our approach exhibits strong generalizability.

3 Method

3.1 Pipeline Overview

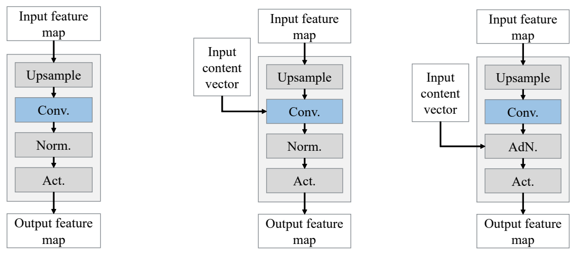
The applicable scenario of our method lies in embedding a watermark into the well-trained Deepfake model after its training has been completed. The overall process of embedding and detection is illustrated in Figure 1. The unique watermark content is encoded using BCH and then embedded into different Deepfake models that require regulation. Embedding a watermark entails two primary phases. The first one is to determine the position for embedding the watermark. We propose a pruning strategy to locate the positions of redundant parameters in the well-trained model, which are then used as the insertion points for the watermark. Second, the watermark parameters are trained to ensure the model's output aligns with that of the original network, all while bearing the embedded watermark. The output from the watermarked Deepfake model has active artifacts. Consequently, a result of the watermarked Deepfake model disseminating misleading information is easily detectable. When evaluating images, we cycle through all the extractors linked to the published Deepfake models. If any extractor reveals the associated BCH-coded watermark content, it is inferred that the image originated from that specific model. Otherwise, the images are not generated by any watermarked model and are subjected to further analysis. Subsequent sections provide a detailed explanation of our method, covering watermark positioning, parameter training, and detection.

3.2 Obtaining the Watermark Position

Generative models usually encode semantic content into latent space vectors, which are then converted into images. Different generative models obtain latent space vectors in different ways. For example, in autoencoders (AEs), the latent space is derived from the encoder. In generative adversarial networks (GANs) [Goodfellow *et al.*, 2014] and Diffusion models [Cao *et al.*, 2022], it comes from a Gaussian distribution, as well as the prompt embedding in some language models. Regardless of the source, the fundamental method by which these models transform latent space vectors into images is consistent, mainly utilizing a decoder. This process entails the conversion of a latent content vector into an image via upsampling and convolutional layers. Considering our goal to introduce a scalable watermark independent of image content, the optimal stage for embedding this watermark is the decoder. In the decoder, the core of the image generation comes from the upsampling layer and convolutional layer (or transpose convolutional layer). We design a convolution kernel sparsification method to nullify the minimal contribution convolutional kernels in the decoder. These nullified locations are then leveraged for watermark embedding. We outline three major upsample blocks in the decoder. Based on these structures, we offer varied sparsification strategies and embedding layer options as shown in Figure 2.

Case 1. Regular Upsampling Blocks

Each regular upsampling block includes the upsampling layer, convolutional layer, activation layer, and normalization layer. We aim to introduce watermarks as minimal perturbations without altering the original image content. Consequently, the mean and standard deviation parameters of the Norm layer must remain nearly consistent before and after embedding. Thus, our operations are confined to the convolutional layer. We design a trainable convolution kernel mask as a pruning operation that removes redundant parameters from the convolutional kernel. If a bias exists in the convolution, a corresponding mask is also trained to identify redundant positions. We take the original output I of the network as a reference, minimize the mask, and make the output I^M through the masked convolution layers almost the same as the original output. Although different generative networks are trained on distinct datasets, we can select a specific dataset independently to generate the output. Though the quality of I might degrade, our focus lies in the difference between I^M and I. We use mean squared error (MSE) and perceptual loss [Johnson *et al.*, 2016] to measure the difference in output. we define $K \in \mathbb{R}^{q\times p\times m\times n}$ as the original convolution kernel, $M_K \in \mathbb{R}^{q\times p}$ is the mask of convolution kernel and $M_B \in \mathbb{R}^q$ is the mask of bias. Therefore, the loss is defined as follows.



(a) Regular upsampling block (b) Modulated convolution (c) Adaptive normalization

Figure 2: Three major upsample blocks in the decoder.

$$L_M = \alpha L_{MSE}(I, I^M) + \beta L_{Per}(I, I^M) + \gamma(M_K + M_B) \tag{1}$$

Where α, β and γ are hyperparameters. Notice that with certain network structures that do not operate sequentially, such as ResBlock [He *et al.*, 2016] or UNet [Ronneberger *et al.*, 2015], our approach remains effective. Since each branch still uses the regular upsampling blocks and the convolution operation satisfies the distributive law, adding watermark parameters in each branch is equivalent to adding a watermark in a single structure.

Case 2. Modulated Convolution

The method represented by StyleGAN2, which embeds semantic vectors into the convolution kernels of each layer using modulated convolution [Karras *et al.*, 2020], aligns with Case 1. Yet, a trainable convolution kernel mask needs to be added to the convolution kernel after semantic vector embedding. This strategy ensures that the sparsified convolution kernel preserves the expression of the semantic vector.

Case 3. Adaptive Normalization

Certain methods [Huang and Belongie, 2017; Park *et al.*, 2019] embed semantic vectors into the normalization layer. Generally, Adaptive Norm will perform the following transformation on the intermediate results:

$$AdN(x, y) = \sigma(y) \cdot \frac{x - \mu(x)}{\sigma(x)} + \mu(y) \tag{2}$$

Where $\mu(\cdot)$ and $\sigma(\cdot)$ are the mean and standard deviation, x is the input of the parameters, and the results are related to the semantic vector y. Given the diverse adaptations of the normalization layer, we choose not to prune it to ensure the generation quality. Then, the convolution kernel operations remain consistent with Case 1.

Choices of Embedding Layers

Previous work [Karras *et al.*, 2019] has established that the larger the size generated by the upsampling layer, the finer the content generated. Given our objective to minimize the damage to the image content, our watermark, similar to the disturbance, should be fine-grained. Given our objective to min-

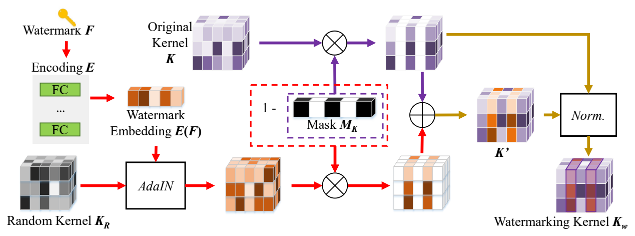


Figure 3: The process of embeddeding the watermark in convlution kernel.

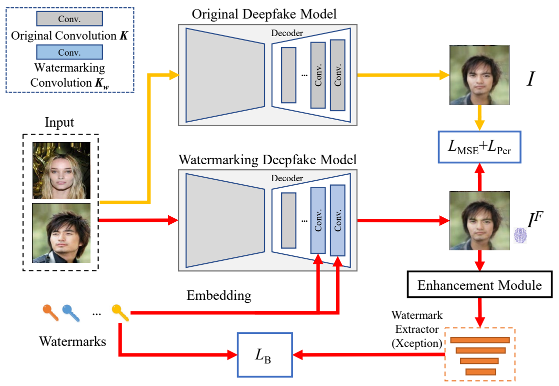


Figure 4: The process of training the watermark parameters.

imize the damage to the image content, our watermark, similar to the disturbance, should only change the fine-grained features of target images. Therefore, we traverse to choose the convolution layers from the last output layer forwards. Experiments have shown that embedding at max as 64*64 is suitable with a relatively small impact on the quality of generation. There is no limit to the number of embedding layers, but experiments have shown that 3-8 layers can meet the requirements in a normal decoder. Too many embedding layers may also compromise both the network's output quality and the convergence speed of the watermark.

3.3 Training the Watermark Parameters

Embedding Watermark into Convolution

The pruning operation mentioned previously outputs the parameters of the original network and a mask, which are set as constants during this phase. We embed the watermark information into the positions of the mask. The process is shown in Figure 3. Here we define a trainable watermark convolution kernel K_R of the same size as the original kernel. We use FC (Fully Connected) layers E to transfer the watermark F to the embedding vector. AdaIN [Huang and Belongie, 2017] is used to embed the watermark into K_R, where the embedding vector is the style vector for AdaIN. If the original convolution has a bias, we also add the watermark embedding into bias redundant positions. As a result, the watermarking convolution kernel is:

$$K' = K \cdot M_K + AdaIN(K_R, E(F)) \cdot (1 - M_K) \tag{3}$$

Given that the sizes of the convolution kernel parameters in the original network vary, the range of variations in the watermark convolution kernel may not match the original, which could potentially lead to gradient explosion during training. We utilize normalization to unify the new watermark convolution kernel and the original convolution kernel. Now we have $\mu(K)$, $\sigma(K)$, $\mu(K')$ and $\sigma(K')$, the normalized kernel K'' is :

$$K'' = \sigma(K) \cdot \frac{K' - \mu(K')}{\sigma(K')} + \mu(K) \tag{4}$$

Note that in Case 2., the value of mean and standard deviation is affected by semantic vectors. $\mu(K)$, $\sigma(K)$ in Case 2 are calculated from the original kernel. It ensures the stability of the watermark embedding and prevents gradient explosion. The normalization operation not only standardizes the watermark parameters but also alters the original parameters. To ensure that the original parameters are not modified, the final watermark convolution kernel is shown as:

$$K_w = K \cdot M_K + K'' \cdot (1 - M_K) \tag{5}$$

For the Adaptive Norm mentioned in Case 3, however, the parameters of the Norm layer will disrupt the transmission of the watermark in the convolution kernel through the network. Therefore, we add a trainable parameter to the result of the adaptive norm layer. Since this disturbance is additive and independent of the semantic vector, it keeps the additive result of the watermark in the Norm layer. Similar to the pruning stage, in order to ensure output consistency, we use perceptual loss, MSE, and total variation loss compared with the original output of the network to control image quality.

Training Watermarking Network

While training the watermark parameters, we also jointly train a watermark extractor. Here we use Xception [Chollet, 2017] and the pre-trained Deepfake detection model [Rössler *et al.*, 2018] as the watermark extractor. The advantage of the Deepfake detection model lies in its ability to identify subtle nuances and artificial inconsistencies within an image. This trait aligns perfectly with watermark detection, as both involve identifying deliberate distortions or details within the image that may not be easily discernible to the human eye. We replace the final output of the Xception classification model to make the output dimension consistent with the watermark F. We use binary cross-entropy loss L_B to measure the difference between the embedded watermark and the output of the watermark extractor. For the input to the watermark extractor, we adopt the image enhancement module in Hidden [Zhu *et al.*, 2018] to enhance the robustness of watermark extraction. We directly use compression methods to replace the random bit selection method for JPEG in Hidden. We define I^F as the output of the watermark embedded network, and $\hat{F}$ is the watermark prediction from Xception. The training process is shown in Figure 4 and the final training loss is:

$$\begin{aligned} L =& \alpha L_{MSE}(I, I^F) + \theta L_{TV}(I^M) + \beta L_{Per}(I, I^F) \\ &+ \gamma(-\frac{1}{N}\sum_{i=1}^{N}[F_i \log \hat{F}_i + (1 - F_i)\log(1 - \hat{F}_i)]) \end{aligned} \tag{6}$$

However, not all the extractors are suitable for the Xception pre-trained model since not all artifacts generated by

Deepfake models can be detected by Xception. Its performance in detecting artifacts depends heavily on its training regimen. If the model hasn't been exposed to certain kinds of artifacts during training or if it's trained on a limited dataset, its efficacy in detecting those specific artifacts in the wild can be compromised. To handle it, we utilize a pre-trained watermark extractor to help the training model converge faster. While each Deepfake model traditionally aligns with a specific Xception watermark extractor, our methodology allows for shared watermark extractors in certain situations, thereby reducing computational overhead. For instance, the watermark extractor can be shared across various models or different versions of a model if they have the same decoder structure. Even for different models with similar decoder structures (For example, the decoder is all composed of simple upsampling blocks), they can still share the same extractor. To enable the extractor sharing, once training for a single watermarked model is completed, we maintain the watermark extractor's parameters while only training the embedding parameters in other watermarked models. Such a training strategy enhances the scalability of our method.

3.4 Watermark Detection

The well-trained watermark extractor can extract watermark information from test images. We match the extracted information with the watermark embedded into the model. If they are matched, it's confirmed as a generated image. Otherwise, it's considered a real image or an image generated by a non-watermarked model, ready for further detection methods. A pivotal observation is that watermark content extracted from natural images without significant watermark features adheres to a Bernoulli distribution with $p = 0.5$. The likelihood of extracting a sequence that conforms to some specific coding from natural images is extremely low. Therefore, during the process of embedding the watermark, we utilize BCH encoding to encode the watermark content, facilitating image detection based on watermark content distribution. Moreover, given the potential for lossy transformations of generated images during public transmission, errors in watermark extraction can arise. The BCH encoding also offers the added advantage of correcting certain errors.

Specifically, we group the watermark content into sets of 7 bits each, where each group is BCH(7,3,1). Let encoded watermark $F^l = \{B_1, B_2, ..., B_n\}$, B is a group of BCH code. The advantage of group BCH is: 1) Compared to performing BCH encoding on the entire content, it offers higher encoding efficiency. Each group of content can correct 1-bit errors, which enhances the extraction robustness. 2) In the entire coding space, assuming the watermark bits extracted from natural images are independent and follow a Bernoulli distribution, the probability of constructing a valid coding group is $(2^4/2^7)^n$. Given the extremely low probability, it can serve as the foundation for classification. When detecting watermark images generated by different types of models, due to the unique watermark features, they will also not conform to BCH encoding. We determined that if more than $\{B_1, B_2, ..., B_k\}, (n \geq k)$ groups cannot be correctly decoded, it is considered a natural image. Watermarked images will also experience BCH decoding failures when transmitted through lossy channels. Therefore, we define a tolerance range based on the watermark content. There are l bits to compare with the input watermark content. If more than τ bits are mismatched, we conclude that the image was not generated by any of the models. Images that satisfy the above two criteria are considered to be real. In our experiment, the watermark sequence l is 64, hence $n = 9$, $k = 5$,$\tau = 16$.

Detection Methods	Real (TPR)	Fake with Watermark (TNR)	Lossy Processing (TNR)							
			Blur		Compress		Random Crop		Resize	
			4	12	60	80	0.6	0.8	0.6	0.8
Arrtibuting (StyleGAN2)	1	1	0.940	0.960	0.980	0.960	0.960	0.960	0.980	0.980
ArtificalGAN (StyleGAN2)	1	1	1	1	1	1	0.000	0.000	1	1
StableSignature (SDM-img2img)	1	1	0.999	0.999	1	1	1	1	1	1
Invisible (All)	1	0.692	0.074	0.058	0	0.028	0.002	0.002	0.002	0.002
Ours (SimSwap)	1	1	1	1	0.998	0.999	0.998	1	0.892	1
Ours (FirstOrder)	1	1	1	1	0.994	0.999	0.998	0.998	0.557	0.999
Ours (HifiFace)	1	0.984	0.990	0.991	0.954	0.976	0.957	0.976	0.870	0.981
Ours (SDM-img2img)	1	0.998	0.998	0.998	0.921	0.997	0.974	0.995	0.877	0.985
Ours (StyleGAN)	1	1	0.950	0.984	0.727	0.918	1	1	0.902	1
Ours (FaceShifter)	1	0.990	0.928	0.908	0.688	0.878	0.683	0.848	0.697	0.947
Ours (LATS)	1	0.999	0.993	0.993	0.849	0.897	0.995	0.999	0.938	0.998
Ours (StyleGAN2)	1	0.977	0.965	0.966	0.945	0.964	0.904	0.970	0.939	0.971

Table 1: Detection accuracy comparison with other watermarking methods.

Detection Methods	Real (TPR)	Fake without Watermark (TNR)	Fake with Watermark (TNR)	Lossy Processing (TNR)							
				Blur		Compress		Random Crop		Resize	
				4	12	60	80	0.6	0.8	0.6	0.8
Seferbekov	0.011	0.137	0.149	0.066	0.083	0.162	0.144	0.091	0.148	0.105	0.118
WS-DAN	0.689	0.473	0.486	0.709	0.724	0.584	0.568	0.366	0.476	0.519	0.493
LR	0.738	0.822	0.828	0.839	0.849	0.807	0.814	0.486	0.860	0.831	0.809
Xception	0.261	0.386	0.391	0.523	0.524	0.116	0.262	0.287	0.436	0.241	0.240
F3net	0.179	0.355	0.488	0.530	0.515	0.204	0.346	0.353	0.578	0.356	0.380
Ict-no-ref	0.483	0.525	0.533	0.507	0.509	0.531	0.529	0.421	0.526	0.530	0.534
Ours(Average)	1	/	0.994	0.978	0.980	0.885	0.954	0.938	0.973	0.834	0.985

Table 2: Detection accuracy comparison with other Deepfake detection methods.

4 Experiment

We first present our datasets and experiment settings. Following, we compare our active Deepfake detection method to other detection methods on detection accuracy and image quality for different Deepfake generation models. Moreover, we systemically conduct an ablation study. Finally, we examine the watermark's resistance to intentional tampering.

4.1 Datasets and Experiment Settings

In our experiments, we select 8 popular Deepfake generation models as the original model awaiting watermark embedding. Among them, SimSwap, FisrtOrder, HifiFace, and SDM-img2img use regular upsamples, FaceShifter and StyleGAN use adaptive norm, and LATS and StyleGAN2 use modulated convolution in the decoder. In the training process, since Simswap, FaceShifter, FirstOrder, HifiFace, SDM-img2img, and LATS need face images as inputs, we use the CelebA [Liu *et al.*, 2015] dataset to train the watermark of these Deepfake models. For StyleGAN and StyleGAN2, we train these models by random style vectors, and the number of vectors is the same as in CelebA images. In the evaluation process, we randomly select 3000 images in FFHQ [Karras *et al.*, 2019] for positive samples,. For negative samples, each Deepfake model generates 3000 images. Specifically, the inputs of the above six models, whose inputs are images, are the positive samples. For the other two models, we randomly generate negative samples by 3000 style vectors which are not contained in the training process. Due to the similarity of upsample blocks in the SimSwap, FirstOrder, and HifiFace's decoders, we found that they can share the same watermark extractor. Hence, the evaluation results for the three methods come from one shared extractor. For the remaining five Deepfake models, due to their more complex structures, we train separate watermark extractors for evaluation. We train the mask for about 1e5 iterations with batchsize 32 and learning rate 1e-2. We train the watermarked models about 5e7 iterations with batch size 32, learning rate 1e-7 for watermarked parameters, and learning rate 1e-5 for the extractor.

4.2 Comparison Results

Comparison Methods

We selected six popular passive detection methods and four active watermark defense methods for comparison. Passive detection methods include two leading detection methods in the frequency domain, including Local Relation (LR) [Chen *et al.*, 2021] and F3net [Qian *et al.*, 2020], top two detection methods in DFDC [Dolhansky *et al.*, 2020], including Seferbekov [Seferbekov, 2021] and WS-DAN [Jing *et al.*, 2021], basic model Xception and anomaly detection ICT [Dong *et al.*, 2022]. Four active methods include three model watermarking methods Attributing [Nie *et al.*, 2023], ArtificalGAN [Yu *et al.*, 2021a] and Stable Signature [Fernandez *et al.*, 2023], and carrier watermarking method Invisible [Zhang *et al.*, 2019]. To align with our criteria, we also consider it as a watermarked image if the watermark has a mismatch rate of less than 1/4 length (16 bits).

Detection Accuracy

To validate the accuracy of Deepfake detection and show the robustness and generalizability of our framework, we analyze detection results across various image conditions. These include original images, JPEG compressed images (compression ratios of 0.6 and 0.8), randomly cropped images (retaining 0.6 and 0.8 of the original size), scaled images (scaling ratios of 0.6 and 0.8), and Gaussian blurred images (kernel size=5, $\sigma \in [4, 12]$). We then compare the True Positive Rate (TPR) and True Negative Rate (TNR) of our method with those of active watermarking and passive Deepfake detection methods. Table 1 demonstrates that our method parallels the performance of specific watermarking methods while exhibiting greater generalizability. Additionally, we outperform the generalized watermarking method named Invisible in terms of accuracy rate. Table 2 reveals that our method significantly enhances detection accuracy compared to passive detection methods. Moreover, the active artifacts introduced by our method improve the detection performance of passive detectors, contributing to a more comprehensive evaluation by supervisors.

Watermarking Methods	PSNR ↑	SSIM ↑
Arrtibuting	27.243	0.936
ArtificalGAN	33.591	0.932
Stable Signature	29.250	0.917
Ours	31.284	0.919

Table 3: Image quality comparisons.

Image Quality

We present our watermarking results demo in Figure 5. The results show that our method does not modify the content and structure of the original image. In addition, we compared our method with three state-of-the-art model watermarking methods, and the results are shown in Table 3. We chose SSIM [Wang *et al.*, 2004] and PSNR as image quality metrics. The higher the two metrics are, the better the image quality is. The results show that our method is comparable to their results. Our method has lower SSIM because the disturbances in AdN lead to a decrease in image quality. However, the SSIM is within acceptable range.

4.3 Ablation Study

The Influence of Different Watermark Lengths

Figure 6 shows the influence of watermark length on the bit extraction accuracy. There is a decline in accuracy for watermarks of length 96 and beyond. In addition, longer watermarks require more training time and usually will not fully converge. Therefore, we opt to use a length of 64 bits.

The Influence of Different Convolution Layers Choices

Figure 7 presents the influence of different upsample numbers on the output image quality and the extraction accuracy for embedded watermarks. We first explain how we calculate the upsample numbers. We consider the theoretical scenario, assuming the output size of the image is a^2, where the final layer comprises two convolutional kernels $[3, x, k, k]$, $[x, 2x, k, k]$. x represents the base channels of the feature map $\mathbf{F} \in \mathbb{R}^{x \times a \times a}$, and k is the size of a single convolutional kernel. Then, $\mathbf{F} \in \mathbb{R}^{2^i x \times a/2^i \times a/2^i}$ in upsample i needs a convolution size of $[2^i x, 2^{i+1}x, k, k]$. As a result, the accumulated parameter size is $p = k^2 \sum_{i=0}^{n} 2^{2i+1}x^2$, and i corresponds to the x-axis in Figure 7. The blue line in the figure shows that embedding fewer parameters results in a decrease in bit extraction accuracy. Thus, if more parameters are chosen, the image quality deteriorates, and thus we cannot extract the watermark. The red line shows that excessive parameters cause the output image to collapse. Therefore, controlling the parameters within a certain range ensures both model performance and the embedding of an effective watermark. Table 4 displays the instantiation of the upsamples numbers, layers numbers, and size of watermarking parameters we choose in each model. Generally, we select $i = 1$. However, there are some exceptions. For instance, since HifiFace has few parameters in each layer, we choose more layers and upsamples to embed the watermark. SDM has many layers and parameters in each upsample. For StyleGan, the more AdN layers we select, the worse image quality we get. Therefore, we set $i = 0$ for them. Overall, we only need to train a small number of parameters to embed the watermark. Compared to training or finetuning the whole network

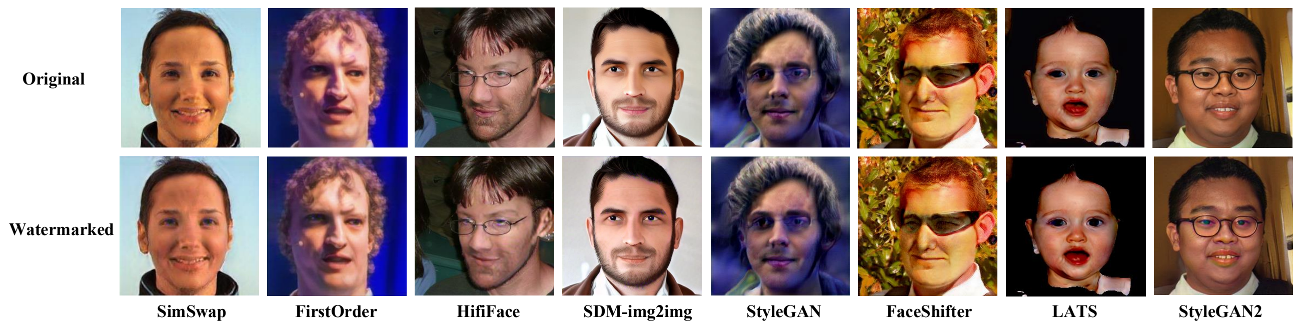


Figure 5: Our watermarking demos.

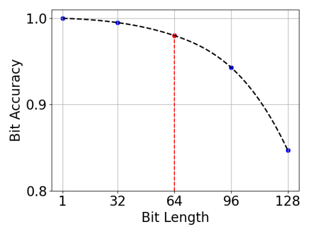


Figure 6: Trade-off between the embedded watermark length and extraction accuracy.

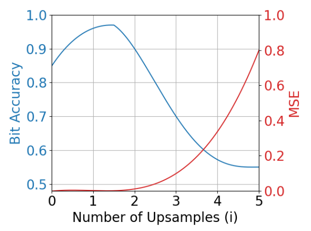


Figure 7: The influence of the embedding layer numbers on bit accuracy and image quality.

Methods	Upsample Numbers (i)	Layer Numbers	Parameter Size	Pruning Rate
Ours_Simswap	1	3	378,048	0.560
Ours_FirstOrder	1	5	2,368,704	0.496
Ours_HifiFace	2	12	356,688	0.431
Ours_SDM-Img2Img	0	8	1,330,560	0.747
Ours_StyleGAN	0	5	5,248	0.386
Ours_FaceShifter	1	8	154,368	0.636
Ours_LATS	1	5	61,632	0.378
Ours_StyleGAN2	1	3	442,572	0.363

Table 4: Watermarking choice of each model.

like most existing watermarking methods, our training efficiency is significantly improved. In addition, we show the average pruning rate of parameters. These values can ensure $L_{Per}(I, I^M) \leq 0.8$ after pruning.

4.4 Adaptive Attack

To examine the watermark's resistance to intentional tampering, we consider three representative adaptive attacks.

Replay Injection Attack

Attackers use the output of the watermarking generation model as the input to another watermarked model in order to confuse the watermark. Our watermarking method is naturally resistant to this attack since we embed the watermark at the final output layer, and the replayed watermark injected by the attacker will be lost at the encoding layer that only encodes the image content.

Pruning Attack

Attackers can remove the watermark parameters using the pruning method described in Section 3.2. To defend the attack, we introduce parameter confusion. This operation introduces perturbation $\epsilon \in \mathbb{R}^{q\times p\times m\times n}$ into the kernel parameters to hide the precise location of the watermark parameters. To prevent perturbations from altering the outputs of convolutions, we can split a single convolution operation into two parallel convolutions, utilizing the commutative and associative properties of convolution. For input x, the original watermarking convolution output is $y = x * K_w$. After parameter confusion, it becomes $y = x * (K_w + \epsilon) + x * (-\epsilon)$. We performed a proof of concept of the pruning attack on SimSwap. Even when 50% of the parameters are pruned, our watermarking method with parameter confusion achieves the same detection accuracy as the original model.

Image Reconstruction Attack

Attackers might use image reconstruction to remove watermark noise. However, our minimal and robust perturbations complicate their ability to find a precise filter that removes the watermark without changing the image content. We evaluate our method against a representative image reconstruction attack [Nie *et al.*, 2022]. While the attack removes the watermark, it significantly distorts the reconstructed image compared to the original. For example, the SSIM between the purified image and the watermarked image is 0.28 on average, while it is 0.49 with the original image.

5 Conclusion

To address generalizability and robustness in Deepfake detection, we propose an efficient, pluggable watermarking framework. Specifically, by deploying convolution kernel sparsification and normalization, we seamlessly embed watermarks, capitalizing on the ubiquitous nature of convolutions present in the Deepfake model decoders. Unique in its design, our approach is independent of the original model training, generating active artifacts that bolster detection efficacy while keeping image quality. Extensive experiments demonstrate our method's adaptability across various Deepfake models while preserving high accuracy even in lossy environments. Furthermore, ablation studies and adaptive attacks verify that our method is efficient with minimal training parameters and is resistant to adaptive attacks. In summary, our watermarking framework enhances Deepfake detection robustness and charts a promising path for future advancements.

Acknowledgments

This work was partly supported by NSFC under Grant No. 92373205 and No. 62102360, National Key R&D Program of China under Grant No.2023YFB4404400, and the Key R&D Program of Ningbo under Grant No.2023Z235.

References

[Cao *et al.*, 2022] Hanqun Cao, Cheng Tan, Zhangyang Gao, Guangyong Chen, Pheng-Ann Heng, and Stan Z Li. A survey on generative diffusion model. *arXiv preprint arXiv:2209.02646*, 2022.

[Chen *et al.*, 2020] Renwang Chen, Xuanhong Chen, Bingbing Ni, and Yanhao Ge. Simswap: An efficient framework for high fidelity face swapping. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2003–2011, 2020.

[Chen *et al.*, 2021] Shen Chen, Taiping Yao, Yang Chen, Shouhong Ding, Jilin Li, and Rongrong Ji. Local relation learning for face forgery detection. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 1081–1088, 2021.

[Chollet, 2017] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[Dagar and Vishwakarma, 2022] Deepak Dagar and Dinesh Kumar Vishwakarma. A literature review and perspectives in deepfakes: generation, detection, and applications. *International journal of multimedia information retrieval*, 11(3):219–289, 2022.

[Dolhansky *et al.*, 2020] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The deepfake detection challenge (dfdc) dataset. *arXiv preprint arXiv:2006.07397*, 2020.

[Dong *et al.*, 2022] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Ting Zhang, Weiming Zhang, Nenghai Yu, Dong Chen, Fang Wen, and Baining Guo. Protecting celebrities from deepfake with identity consistency transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9468–9478, 2022.

[Fernandez *et al.*, 2023] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. *arXiv preprint arXiv:2303.15435*, 2023.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Huang and Belongie, 2017] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[Huang *et al.*, 2022] Hao Huang, Yongtao Wang, Zhaoyu Chen, Yuze Zhang, Yuheng Li, Zhi Tang, Wei Chu, Jingdong Chen, Weisi Lin, and Kai-Kuang Ma. Cmua-watermark: A cross-model universal adversarial watermark for combating deepfakes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 989–997, 2022.

[Jing *et al.*, 2021] Shao Jing, Shi Huafeng, Yin Zhenfei, Fang Zheng, Yin Guojun, Chen Siyu, Ning Ning, and Liu Yu. Deepfake detection challenge solution: Robustforensics. https://github.com/Siyu-C/RobustForensics, 2021.

[Johnson *et al.*, 2016] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, pages 694–711. Springer, 2016.

[Karras *et al.*, 2019] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[Karras *et al.*, 2020] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[Kim *et al.*, 2020] Changhoon Kim, Yi Ren, and Yezhou Yang. Decentralized attribution of generative models. In *International Conference on Learning Representations*, 2020.

[Li *et al.*, 2019] Lingzhi Li, Jianmin Bao, Hao Yang, Dong Chen, and Fang Wen. Faceshifter: Towards high fidelity and occlusion aware face swapping. *arXiv preprint arXiv:1912.13457*, 2019.

[Li *et al.*, 2022] Changjiang Li, Li Wang, Shouling Ji, Xuhong Zhang, Zhaohan Xi, Shanqing Guo, and Ting Wang. Seeing is living? rethinking the security of facial liveness verification in the deepfake era. *USENIX Security 2022*, 2022.

[Li *et al.*, 2023] Gen Li, Xianfeng Zhao, and Yun Cao. Forensic symmetry for deepfakes. *IEEE Transactions on Information Forensics and Security*, 18:1095–1110, 2023.

[Liu *et al.*, 2015] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[Nie *et al.*, 2022] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. *arXiv preprint arXiv:2205.07460*, 2022.

[Nie *et al.*, 2023] Guangyu Nie, Changhoon Kim, Yezhou Yang, and Yi Ren. Attributing image generative models using latent fingerprints. *arXiv preprint arXiv:2304.09752*, 2023.

[Or-El *et al.*, 2020] Roy Or-El, Soumyadip Sengupta, Ohad Fried, Eli Shechtman, and Ira Kemelmacher-Shlizerman. Lifespan age transformation synthesis. In *Computer Vision–ECCV 2020*, pages 739–755. Springer, 2020.

[Park *et al.*, 2019] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019.

[Qian *et al.*, 2020] Yuyang Qian, Guojun Yin, Lu Sheng, Zixuan Chen, and Jing Shao. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In *European Conference on Computer Vision (ECCV)*, pages 86–103. Springer, 2020.

[Rombach *et al.*, 2022] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[Ronneberger *et al.*, 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[Rössler *et al.*, 2018] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*, 2018.

[Ruiz *et al.*, 2020] Nataniel Ruiz, Sarah Adel Bargal, and Stan Sclaroff. Disrupting deepfakes: Adversarial attacks against conditional image translation networks and facial manipulation systems. In *Computer Vision–ECCV 2020 Workshops*, pages 236–251. Springer, 2020.

[Schwarcz and Chellappa, 2021] Steven Schwarcz and Rama Chellappa. Finding facial forgery artifacts with parts-based detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 933–942, 2021.

[Seferbekov, 2021] Selim Seferbekov. Deepfake detection (dfdc) solution. https://github.com/selimsef/dfdc_deepfake_challenge, 2021.

[Shiohara and Yamasaki, 2022] Kaede Shiohara and Toshihiko Yamasaki. Detecting deepfakes with self-blended images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18720–18729, 2022.

[Siarohin *et al.*, 2019] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in neural information processing systems*, 32, 2019.

[Uchida *et al.*, 2017] 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.

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

[Wang *et al.*, 2021] Yuhan Wang, Xu Chen, Junwei Zhu, Wenqing Chu, Ying Tai, Chengjie Wang, Jilin Li, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Hififace: 3d shape and semantic prior guided high fidelity face swapping. *arXiv preprint arXiv:2106.09965*, 2021.

[Wang *et al.*, 2022] Xueyu Wang, Jiajun Huang, Siqi Ma, Surya Nepal, and Chang Xu. Deepfake disrupter: The detector of deepfake is my friend. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14920–14929, 2022.

[Wu *et al.*, 2020] Hanzhou Wu, Gen Liu, Yuwei Yao, and Xinpeng Zhang. Watermarking neural networks with watermarked images. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(7):2591–2601, 2020.

[Yang *et al.*, 2022] Tianyun Yang, Ziyao Huang, Juan Cao, Lei Li, and Xirong Li. Deepfake network architecture attribution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4662–4670, 2022.

[Yu *et al.*, 2021a] Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 14448–14457, 2021.

[Yu *et al.*, 2021b] Ning Yu, Vladislav Skripniuk, Dingfan Chen, Larry S Davis, and Mario Fritz. Responsible disclosure of generative models using scalable fingerprinting. In *International Conference on Learning Representations*, 2021.

[Zhang *et al.*, 2019] Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv preprint arXiv:1909.01285*, 2019.

[Zhang *et al.*, 2020] Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. Model watermarking for image processing networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12805–12812, 2020.

[Zhang, 2022] Tao Zhang. Deepfake generation and detection, a survey. *Multimedia Tools and Applications*, 81(5):6259–6276, 2022.

[Zhu *et al.*, 2018] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672, 2018.