GaussianMarker: Uncertainty-Aware Copyright Protection of 3D Gaussian Splatting

Xiufeng Huang[1,2], **Ruiqi Li**[1], **Yiu-ming Cheung**[1], **Ka Chun Cheung**[2], **Simon See**[2], **Renjie Wan**[1*]
[1] Department of Computer Science, Hong Kong Baptist University
[2] NVIDIA AI Technology Center
xiufenghuang@life.hkbu.edu.hk, {csrqli, ymc}@comp.hkbu.edu.hk
{chcheung, ssee}@nvidia.com, renjiewan@hkbu.edu.hk

Abstract

3D Gaussian Splatting (3DGS) has become a crucial method for acquiring 3D assets. To protect the copyright of these assets, digital watermarking techniques can be applied to embed ownership information discreetly within 3DGS models. However, existing watermarking methods for meshes, point clouds, and implicit radiance fields cannot be directly applied to 3DGS models, as 3DGS models use explicit 3D Gaussians with distinct structures and do not rely on neural networks. Naively embedding the watermark on a pre-trained 3DGS can cause obvious distortion in rendered images. In our work, we propose an uncertainty-based method that constrains the perturbation of model parameters to achieve invisible watermarking for 3DGS. At the message decoding stage, the copyright messages can be reliably extracted from both 3D Gaussians and 2D rendered images even under various forms of 3D and 2D distortions. We conduct extensive experiments on the Blender, LLFF, and MipNeRF-360 datasets to validate the effectiveness of our proposed method, demonstrating *state-of-the-art* performance on both message decoding accuracy and view synthesis quality. Project page: https://kevinhuangxf.github.io/GaussianMarker.

1 Introduction

3DGS [1] has introduced a new category of 3D assets that can be readily created and extensively distributed online [2]. However, the ownership of these created 3D assets can be vulnerable if malicious users distribute and manipulate the 3DGS without authorization. *How can we effectively protect the ownership of those created 3DGS models?*

3DGS represents the scene via 3D Gaussian parameters, which can be standardized into point cloud formats. Such formats can be easily shared and show strong compatibility with the mainstream 3D assets processing pipeline [3]. However, unauthorized users can exploit this convenience to distribute 3DGS models and maliciously alter the 3D Gaussian parameters. These unauthorized 3DGS models can then be easily used to produce 2D images. Since ownership of 3DGS models can be compromised through unauthorized manipulations of 3D Gaussian parameters and 2D images, an effective ownership solution should enable owners to assert their rights over both the 3D Gaussian parameters and the corresponding 2D images.

Similar to copyright protection for digital assets such as videos and images, protecting copyright for 3DGS models can be achieved via digital watermarking. Aligned with the established principles in digital watermarking [4, 5], effective copyright protection methods for 3DGS models should satisfy

*Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-1040

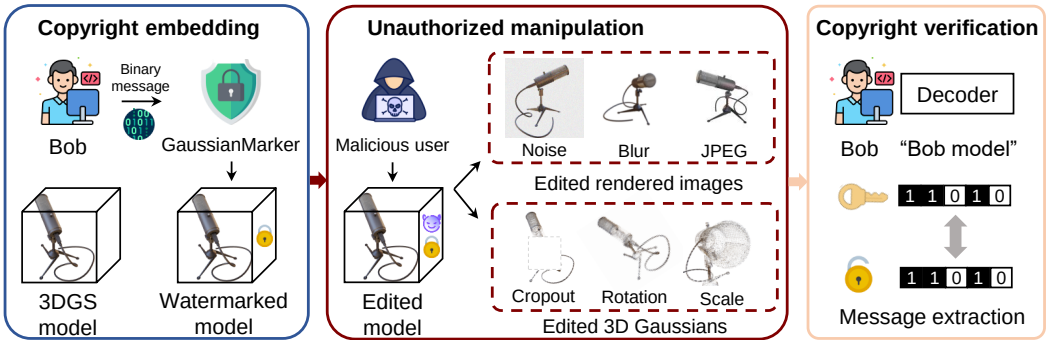


Figure 1: Our proposed scenario for copyright protection over the 3DGS assets. Once users have created 3DGS assets, they can apply our proposed 3DGS watermarking method to create watermarked 3DGS models. If unauthorized users maliciously apply 3D editing or different volume splatting settings on the watermarked 3DGS model, the 3DGS model owners can reliably retrieve the copyright message from the altered 3D Gaussian parameters or rendered 2D images to verify ownership.

two key standards. First, they should maintain **invisibility**, ensuring that the embedded copyright messages do not cause significant distortion in both 3D Gaussian parameters and the rendered 2D images. Second, they should exhibit **robustness**, enabling reliable extraction of the copyright messages even under various 2D or 3D distortions.

Although several methods [4, 6] have been investigated to protect the copyright of radiance fields, these methods are specifically designed for Neural Radiance Field (NeRF) [7], a framework known for its implicit property. For example, CopyRNeRF [4] embeds copyright messages via multilayer perceptrons (MLPs) into the implicit neural parameters in NeRF [7] and extracts the copyright messages from rendered 2D images. However, embedding messages into 3D Gaussian parameters via MLPs can easily undermine the 3D Gaussian positions and lead to noticeable geometry distortions in the rendered images, thereby degrading the **invisibility**. Additionally, since current copyright solutions for NeRF can only extract messages from rendered images, 3DGS model owners lack approaches to directly extract ownership messages from the 3D Gaussian parameters. This hinders the direct assertion of ownership over 3DGS model, thereby undermining the **robustness**.

Rather than directly embedding the copyright messages into the 3D Gaussian parameters, we propose an uncertainty-aware watermarking method to optimize the embedded copyright messages. We apply Laplace approximation to estimate the uncertainty [8] in the radiance fields for determining how large we can add perturbations to different 3D Gaussian parameters. From a Bayesian inference perspective [9], 3D Gaussian parameters with high uncertainty can tolerate larger perturbations. Thus, we keep the original 3D Gaussian parameters unchanged and densify 3D Gaussian parameters with high uncertainty. These newly densified 3D Gaussians are regarded as the perturbations for embedding copyright messages. Such perturbations can be transmitted into rendered 2D images with unperceivable distortion, which ensures **invisibility**.

To ensure robust copyright message extraction on both 3D Gaussian parameters and rendered 2D images, we utilize both 3D and 2D message decoders. The 3D message decoder extracts the copyright messages on the 3D Gaussian parameters based on a PointNet [10] architecture. The 2D message decoder based on the image watermarking method HiDDeN [11] extracts the copyright messages on rendered 2D images. We incorporate 3D and 2D distortion layers into our training process to ensure **robustness** of copyright message extraction against various malicious manipulations. The 3D distortion layer is designed to defend against malicious 3D editing, such as noise, translation, rotation, and cropping. Meanwhile, the 2D distortion layer is designed to withstand significant degradation in the rendered images, such as noise, JPEG compression, scaling, and blurring.

Our whole framework is shown in Figure 2, we estimate the uncertainty for the 3DGS model to add perturbations to different 3D Gaussian parameters. Then, we keep the original 3D Gaussian parameters unchanged and densify 3D Gaussian parameters with high uncertainty. These newly densified 3D Gaussians are regarded as the perturbations for embedding copyright messages and can be verified via 3D Gaussian parameters and 2D images. Our contribution can be summarized as follows:

https://doi.org/10.52202/079017-1040

- A novel method to help claim the ownership of 3D Gaussian Splatting models.
- A uncertainty-aware message embedding strategy to incorporate 3D perturbations into selected 3D Gaussian parameters to achieve invisibility.
- The copyright messages can be extracted from both 3D Gaussian parameters and rendered 2D images, showing robustness to different 3D and 2D distortions.

2 Related works

3D Gaussian Splatting. 3DGS has been rapidly adopted across multiple domains and has demonstrated remarkable results. Unlike NeRF [7] and its variants [12–14] reply on the implicit neural representation (INR) to reconstruction the 3D scene, 3DGS [1] has an explicit point cloud structure and has been expanded to various developments and applications. Mip-Splatting [15] utilizes a 3D smoothing filter and a 2D Mip filter to address frequency constraints for effective anti-aliasing. Dynamic 3DGS [16] represents the dynamic motion of the scene by processing the center location and rotation of each Gaussian over time, which enables dense non-rigid 6-DOF tracking of the entire scene. SuGar [17] reconstructs the mesh surface with a regularization term for the Gaussian Splatting optimization to promote alignment of the Gaussians with the scene's surface. 3DGS avatar [18] is a brand-new way to create digital humans compared with traditional methods based on 3D human meshes such as SMPL [19]. With the rapid development of point-based 3D Gaussian rendering, it is necessary to develop an efficient copyright protection method for 3DGS [1] models.

2D digital watermarking. Traditional 2D watermarking methods typically embed information in the least significant bits (LSB) of image pixels [20]. Other advanced methods encode information into the frequency domains based on the Discrete Wavelet Transform (DWT) and Singular Value Decomposition (SVD) [21, 22]. Deep-learning [23–26] has made significant progress in image watermarking [27–33]. HiDDeN [11] is one of the first deep image watermarking methods that outperformed traditional methods. RedMark [5] introduces scalable residual connections for embedding binary images in any transform domain. Robustness is a critical requirement for watermarking, ensuring resilience against various distortions and even adversarial attacks [34, 35]. Deep-learning-based watermarking methods have emerged as a crucial component in video copyright protection [5, 28, 36], such as RivaGAN [36], which utilizes an attention-based mechanism for embedding hidden messages in videos. However, the 2D digital watermarking methods for images or videos can differ significantly from 3D digital watermarking methods for explicit 3D models.

3D digital watermarking. Most 3D digital watermarking approaches are designed for explicit 3D models [37–41]. For example, Deep 3D-to-2D [41] can embed messages in 3D meshes [38, 42] and retrieve them from 2D rendered views [43]. Recently, several 3D digital watermarking approaches [4, 6, 44, 45] have emerged for NeRF [7] to watermark the implicit neural representation (INR) and extract the hidden information from the rendered images. CopyRNeRF [4] generates watermarked color representations to ensure the invisibility of hidden copyright messages. StegaNeRF [6] designs an optimization framework for steganographic information embedding in NeRF renderings. However, both explicit 3D watermarking [37, 38, 41] and NeRF watermarking approaches [4, 6] are not applicable for 3DGS to simultaneously protect the explicit 3D Gaussians and the 2D rendered images. This motivates us to develop digital watermarking for 3DGS models.

3 Preliminary of 3D Gaussian Splatting

Starting from a sparse set of Structure-from-Motion (SfM) [46] points, the goal of 3DGS [1] is to optimize a scene representation that enables high-quality novel view synthesis. The scene is modeled as a collection of 3D Gaussians:

$$G(x) = e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}, \tag{1}$$

where x is any positions in the 3D scene, μ is the 3D Gaussian center position, and Σ is the 3D Gaussian covariance matrix. By utilizing a scaling matrix S and rotation matrix R, we can determine the corresponding $\Sigma = RSS^TR^T$ and ensure Σ is positive semi-definite. The 3D Gaussians need to be further projected to 2D Gaussians for rendering by volume splatting [47] method. During rendering, 3DGS follows a typical neural point-based approach [48] to compute the color C of a

 https://doi.org/10.52202/079017-1040

pixel by blending $\mathcal{N}$ depth ordered points:

$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{2}$$

where c_i is the color estimated by the spherical harmonics (SH) coefficients of each Gaussian, and α_i is given by evaluating a 2D Gaussian with covariance $\Sigma^{'}$ [49] multiplied with a per-point opacity. Consequently, the 3D Gaussians $\mathcal{G}$ contain parameters $\boldsymbol{\theta}$ including five different properties $\{\mu, R, S, c, \alpha\}$ to represent a 3D scene.

4 Proposed method

Our **scenario** is shown in Figure 1. We propose embedding copyright messages into 3D Gaussian parameters to protect the copyright of 3DGS models. These messages can be extracted from both the 3D Gaussian parameters and the rendered 2D images. The proposed uncertainty-aware watermarking method can claim ownership over both 3D and 2D assets derived from 3DGS models. As mentioned in Section 1, an effective watermarking algorithm for 3DGS models should achieve both invisibility in rendered novel views, and robustness in decoded messages, via the optimization goal of:

$$\mathcal{L} = \underbrace{d_1\{\mathbf{I},\ \hat{\mathbf{I}}\}}_{\text{rendered view}} + \underbrace{d_2\{D[\mathbf{I}],\ \mathbf{M}\}}_{\text{message decoding}}, \tag{3}$$

where $\mathbf{I}$ is the rendered novel view, $\hat{\mathbf{I}}$ is the ground truth image, D is the message decoder and $\mathbf{M}$ is the copyright message. We can use appropriate distance metrics d_1 and d_2 to estimate and minimize the error in rendered views and decoded messages. A straightforward method could be embedding the copyright messages as perturbations into the 3D Gaussian parameters. However, directly embedding perturbations into 3D Gaussian parameters without constraints can easily undermine the position and geometry of 3D Gaussians and cause obvious distortion in the rendered images. To solve this issue, we propose an uncertainty-aware perturbation strategy to embed copyright messages, as illustrated below.

4.1 Uncertainty-aware 3DGS watermarking

Estimating the uncertainty of Gaussian parameters. To ensure the invisibility of embedded messages in both the 3D and 2D domains, we allow only a subset of the 3D Gaussian parameters where ownership messages can be embedded. Specifically, as previous works [9] have already shown that the Gaussian parameters with high uncertainty are more tolerant to external perturbations, we select parameters with high uncertainty to incorporate ownership messages. If we estimate the model parameter posterior $p(\boldsymbol{\theta}|\mathcal{D})$, where $\boldsymbol{\theta}$ is the model parameters of the 3D Gaussians model $\mathcal{G}$ and $\mathcal{D}$ is the training dataset, then the predictive distribution $p(\mathbf{I}|\mathbf{V}, \mathcal{D})$ can be computed by marginalize over the model posterior:

$$p(\mathbf{I}|\mathbf{V}, \mathcal{D}) = \int_{\boldsymbol{\theta}} p(\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} = \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathcal{D})}[p(\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta})], \tag{4}$$

where $\mathbf{I}$ is rendered image at a test view $\mathbf{V}$. In this inference integration, for a converged model, parameters $\boldsymbol{\theta}^*$ with larger uncertainty quantified by posterior variance can tolerate greater perturbation. Therefore, we densify only the parameters above an uncertainty threshold τ_{unc} to add perturbations for less impacting the rendered images. Laplace approximation provides an analytical expression for a posterior distribution in the form of a Gaussian distribution with the mean equal to the maximum a posterior (MAP) estimation $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ [50], and the covariance equal to the reciprocal of observed Fisher information: $p(\boldsymbol{\theta}|\mathcal{D}) \sim \mathcal{N}(\boldsymbol{\theta}^*, \boldsymbol{\Gamma})$. Thus, the uncertainty of the 3DGS parameters can be estimated by the Hessian matrix $\mathbf{H}[\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}^*]$ as the approximated Fisher information [51]:

$$\mathbf{H}[\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}^*] = \nabla_{\boldsymbol{\theta}} f(\mathbf{V}; \boldsymbol{\theta}^*)^T \nabla^2_{f(\mathbf{V}; \boldsymbol{\theta}^*)} H[\mathbf{I} \mid f(\mathbf{V}; \boldsymbol{\theta}^*)] \nabla_{\boldsymbol{\theta}} f(\mathbf{V}; \boldsymbol{\theta}^*), \tag{5}$$

where $f(\mathbf{V}; \boldsymbol{\theta}^*)$ is the rendered image with the converged 3D Gaussians parameters $\boldsymbol{\theta}^*$ at view $\mathbf{V}$ and $\nabla^2_{f(\mathbf{V}; \boldsymbol{\theta}^*)} H[\mathbf{I} \mid f(\mathbf{V}; \boldsymbol{\theta}^*)] = 1$ as we assume the covariance of RGB in images is equal to one [8]. Hence, the Hessian matrix can be simplified as: $\mathbf{H}[\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}^*] = \nabla_{\boldsymbol{\theta}} f(\mathbf{V}; \boldsymbol{\theta}^*)^T \nabla_{\boldsymbol{\theta}} f(\mathbf{V}; \boldsymbol{\theta}^*)$.

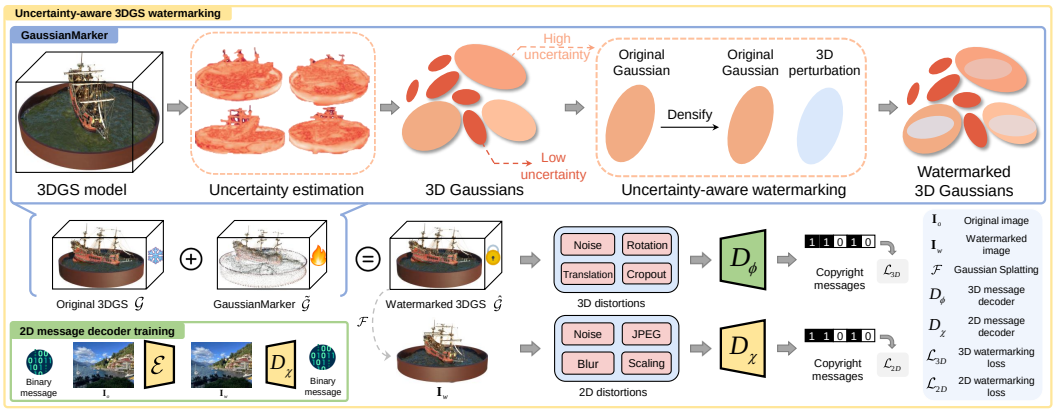


Figure 2: The overview of our proposed uncertainty-aware 3DGS watermarking. We apply uncertainty estimation to the created 3DGS model. The 3D Gaussians with high uncertainty will be densified. These new densified Gaussians will be regarded as the 3D perturbations and embedded into the original Gaussians to create watermarked 3D Gaussians. The copyright messages can be retrieved from the watermarked 3D Gaussians via the 3d message decoder under various 3D editing. The copyright messages can also be retrieved from the watermarked images via the 2D message decoder against various 2D distortions.

As Fisher Information is additive, we compute the model uncertainty $\mathbf{U}$ by summing the Hessians of model parameters across all different views in the training dataset $\mathcal{D}$:

$$\mathbf{U} = \sum_{i=1}^{N} \mathbf{H}\left[\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}^*\right], \tag{6}$$

where i is the index and N is the total samples in $\mathcal{D}$, and all Gaussian parameters are used to calculate the uncertainty of the 3DGS model: $\mathbf{H}\left[\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}^*\right] = \mathbf{H}\left[\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}^*_{\mu}\right] + \mathbf{H}\left[\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}^*_{\mathbf{R}}\right] + \mathbf{H}\left[\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}^*_{\mathbf{S}}\right] + \mathbf{H}\left[\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}^*_{\mathbf{c}}\right] + \mathbf{H}\left[\mathbf{I} \mid \mathbf{V}, \boldsymbol{\theta}^*_{\alpha}\right]$.

GaussianMarker. As shown in Figure 2, we demonstrate the overall framework of our proposed uncertainty-aware 3DGS watermarking, aka GaussianMarker. By leveraging the quantified uncertainty $\mathbf{U}$ of the created 3DGS model, we can effectively distinguish between parameters that are resilient to perturbations and those that are vulnerable. Parameters exhibiting low uncertainty are identified as highly sensitive to perturbations. Conversely, parameters characterized by high uncertainty are more tolerant to perturbations, implying that perturbations can be embedded in these areas with negligible impact on the quality of the final rendered images. By targeting 3D Gaussians with high uncertainty, we can incorporate effective 3D perturbations that remain detectable by our designated message decoders, and maintain invisibility on the 3D Gaussians and rendered images. To achieve this, we retain the integrity of the original 3D Gaussians, denoted as $\mathcal{G}$. We then densify those 3D Gaussians with high uncertainty. The new densified Gaussians are regarded as the perturbations $\tilde{\mathcal{G}}$ for copyright message embedding:

$$\tilde{\mathcal{G}} = \{g(G_i) \mid G_i \in \mathcal{G}, \mathbf{U}_i > \tau_{unc}\}, \tag{7}$$

where $g(G_i)$ is the densified Gaussian with the densify function $g(\cdot)$ on the i^{th} Gaussian G_i by random sampling new position $\tilde{\mu}_i$ under the distribution $\tilde{\mu}_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ and other Gaussian parameters are cloned, $\mathbf{U}_i$ is the uncertainty of G_i, and τ_{unc} is the threshold for uncertainty. We compute the average uncertainty value of all original 3D Gaussians $\mathcal{G}$ as the default uncertainty threshold: $\tau_{unc} = \mathbf{U}/L$, where L is the total number of the 3D Gaussians in $\mathcal{G}$ [2]. We dub $\tilde{\mathcal{G}}$ as our proposed GaussianMarker for embedding the copyright messages. Similar to image watermarking methods apply 2D perturbation on the cover images, we directly embed GaussianMarker $\tilde{\mathcal{G}}$ into the original Gaussians $\mathcal{G}$ to compose the watermarked Gaussians $\hat{\mathcal{G}} = \mathcal{G} \cup \tilde{\mathcal{G}}$. Under the position sampling distribution of $\tilde{\mu}_i \sim \mathcal{N}(\mu_i, \Sigma_i)$, GaussianMarker $\tilde{\mathcal{G}}$ have a subtle geometry difference with the

[2]The influence of different uncertainty thresholds is further discussed in the supplementary materials.

https://doi.org/10.52202/079017-1040

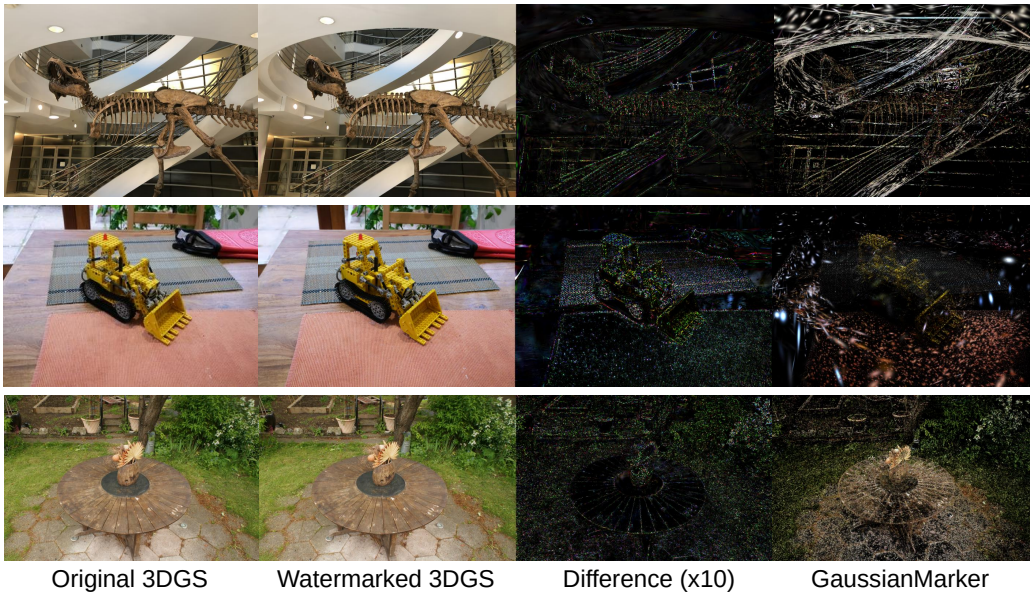


Figure 3: Visualization of the results obtained by our proposed approach. For each row, we display the original rendered image, the watermarked rendered image, the difference ($\times 10$) between the watermarked and original rendered images, and our proposed GaussianMarker as 3D perturbations for copyright message embedding. The scalings of GaussianMarker are adjusted for better visualization. We provide more visualization examples in the supplementary material.

original Gaussians $\mathcal{G}$. Moreover, GaussianMarker $\hat{\mathcal{G}}$ can be effectively transmitted into the rendered images based on the point-based rendering in Equation (2). We optimize GaussianMarker $\hat{\mathcal{G}}$ in the Section 4.3 so that such 3D perturbations can be effectively detected by both 2D and 3D message decoders. In Figure 3, we present visualization results, which demonstrate its capability to embed copyright messages as imperceptible 3D perturbations.

4.2 Message decoders

2D message decoder for rendered images. We select the classical HiDDeN [11] as the 2D message decoder D_χ to retrieve copyright messages from the rendered 2D images. The HiDDeN [11] encoder takes a cover image x_o and a binary message $\mathbf{M}$ with length N_b as the inputs. The HiDDeN [11] decoder outputs a residual image of the same size as 2D perturbation applied on the original image to produce a watermarked image $x_w = x_o + \delta$. Specifically, we first pre-train HiDDeN [11] encoder and decoder to obtain a comprehensive understanding of image watermark embedding and extraction processes. During training, similar to the settings in HiDDeN [11] for the robustness at the image level, we apply several types of 2D distortions including Gaussian noise, random rotation, random cropping, and JPEG compression. After training, the HiDDeN [11] encoder and the adversarial network are discarded. We then use this pre-trained HiDDeN decoder to optimize our GaussianMarker $\tilde{\mathcal{G}}$ mentioned in Section 4.3.

3D message decoder for 3D Gaussians. The inherent complexity of the implicit neural representation in NeRF [7] presents significant challenges for copyright message extraction directly from the neural network parameters. On the contrary, 3DGS [1] represents the 3D scene by 3D Gaussian parameters with an explicit geometry. Traditional 3D watermarking embeds and retrieves messages from 3D meshes. The deep learning-based 3D watermarking methods utilize networks such as PointNet [10] to enhance the message embedding process in 3D meshes[41]. Although 3D Gaussians have different geometrical representations from the 3D meshes, the PointNet [10] architectures can be easily adapted to 3D Gaussians by regarding the 3D Gaussian mean μ as the point position and other parameters as the associated point features. Thus, we adopt the PointNet [10] as our 3D message decoder D_ψ to retrieve copyright messages from the watermarked 3D Gaussians $\hat{\mathcal{G}}$. In specific, we randomly sample a subset Gaussians (from $10k$ to $50k$) from the watermarked Gaissians $\hat{\mathcal{G}}$ and treat these

selected Gaussians as watermarked points $\mathbf{P}_w$. A PointNet-like 3D message decoder D_ψ is used to decode the copyright message $\hat{\mathbf{M}}$ from these watermarked points as $\hat{\mathbf{M}} = D_\psi(\mathbf{P}_w)$. Similarly, we also randomly sample a same number subset Gaussians on the original Gaussians $\mathcal{G}$ and treat these selected Gaussians as original points $\mathbf{P}_o$. A PointNet-like discriminator D_ξ is used to distinguish the original points $\mathbf{P}_o$ and the watermarked points $\mathbf{P}_w$. The 3D message decoder D_ψ uses a PointNet [10] architecture, with a modified fully connected layer to predict the copyright messages. The 3D discriminator D_ξ also uses a PointNet [10] architecture with a fully connected output layer for binary classification.

4.3 Optimization

Our optimization contains two phases. In the first phase, we distill the watermarking knowledge from the 2D message decoder to the embedded GaussianMarker $\tilde{\mathcal{G}}$ for predicting copyright messages on rendered 2D images. In the second phase, we optimize the 3D message decoder with the watermarked Gaussians $\hat{\mathcal{G}}$ for predicting copyright messages on 3D Gaussian parameters.

Distilling watermarking knowledge. We keep the original 3D Gaussians $\mathcal{G}$ unchanged, and optimize the embedded GaussianMarker $\tilde{\mathcal{G}}$ via teacher-student knowledge distillation. As discussed in [52], the pre-trained feature from 2D space can be distilled to the 3D space. Thus, we use the pre-trained 2D message decoder D_χ as the teacher network to distill watermarking knowledge from the 2D perturbation on images to the 3D perturbation in GaussianMarker $\tilde{\mathcal{G}}$. During the optimization for the embedded GaussianMarker $\tilde{\mathcal{G}}$, the copyright messages can be decoded from the watermarked image $\mathbf{I}_w$ via the 2D message decoder D_χ as $\hat{\mathbf{M}} = D_\chi(\mathbf{I}_w)$, where $\hat{\mathbf{M}}$ is the copyright messages decoded by D_χ, and $\mathbf{I}_w$ is rendered by the watermarked Gaussians $\hat{\mathcal{G}}$. We compute binary cross entropy (BCE) between the original message $\mathbf{M}$ and the decoded message $\hat{\mathbf{M}}$ as the message loss $\mathcal{L}_{msg}$ to ensure watermarking capability:

$$\mathcal{L}_{msg} = -(\mathbf{M}\log(\hat{\mathbf{M}}) + (1-\mathbf{M})\log(1-\hat{\mathbf{M}})). \tag{8}$$

We also use the photometric loss $\mathcal{L}_{rec} = \|\mathbf{I}_w - \mathbf{I}_o\|_2^2$ between the watermarked image $\mathbf{I}_w$ and the original images $\mathbf{I}_o$ for multi-view consistency. We combine the $\mathcal{L}_{msg}$ and $\mathcal{L}_{rec}$ into the final 2D watermarking loss $\mathcal{L}_{2D} = \lambda_1\mathcal{L}_{msg} + \lambda_2\mathcal{L}_{rec}$, where λ_1, λ_2 are the weights for adapting the losses.

Optimizing 3D message decoder. Once the embedded GaussianMarker $\tilde{\mathcal{G}}$ is optimized, we proceed to train the 3D message decoder D_ψ with the watermarked Gaussians $\hat{\mathcal{G}}$. To make our 3D message decoding robust to different 3D distortions, we add a 3D distortion layer T during the 3D message decoder optimization. Several commonly used 3D distortions are used: 1) additive random Gaussian noise with parameter σ; 2) random axis-angle rotation with parameter r; 3) random translation with parameter t; and 4) random cropout with parameter cr. The copyright messages $\hat{\mathbf{M}}^{'}$ can be decoded from the randomly selected watermarked points $\mathbf{P}_w$ from the watermarked Gaussians $\hat{\mathcal{G}}$ via the 3D message decoder D_ψ as $\hat{\mathbf{M}}^{'} = D_\phi(T(\mathbf{P}_w))$. We compute the BCE between the original message $\mathbf{M}$ and extracted message $\hat{\mathbf{M}}^{'}$ as the 3D message loss $\mathcal{L}_{msg'} = -(\mathbf{M}\log(\hat{\mathbf{M}}^{'}) + (1-\mathbf{M})\log(1-\hat{\mathbf{M}}^{'}))$. We also apply the adversarial loss $\mathcal{L}_{adv}$ to optimize the 3D discriminator D_ξ for classifying the original points $\mathbf{P}_o$ and watermarked points $\mathbf{P}_w$:

$$\mathcal{L}_{adv} = \log(1 - D_\xi(T(\mathbf{P}_o))) + \log(D_\xi(T(\mathbf{P}_w))). \tag{9}$$

We combine $\mathcal{L}_{msg'}$ and $\mathcal{L}_{adv}$ into the final 3D watermarking loss $\mathcal{L}_{3D} = \lambda_1^{'}\mathcal{L}_{msg'} + \lambda_2^{'}\mathcal{L}_{adv}$, where $\lambda_1^{'}$, $\lambda_2^{'}$ are the weights for adapting the losses.

5 Experiments

5.1 Experimental settings

Dataset. We use three benchmark datasets for evaluation: **Blender** [7] (8 detailed synthetic objects), **LLFF** [53] (9 real-world scenes), and **Mip-NeRF360** [54] (9 real-world scenes). For Blender [7], we directly follow the dataset splitting to use 100 viewpoints for training and 200 views for testing. For LLFF [53], we follow the dataset splitting in NeRF [7]. In general, $1/8$ images in each scene are

https://doi.org/10.52202/079017-1040

used for testing and others for training. For Mip-NeRF360 [54], we use a train/test split suggested by Mip-NeRF360, taking every 8^{th} photo for testing and others for training. All testing viewpoints are used to compute the average values during the evaluation session.

Implementation details. As our motivation is to protect the copyright of 3DGS that has already been created, we define our training into two stages. In the first stage, we create 3DGS models by training them on Blender [7], LLFF [53], and Mip-NeRF360 [54] datasets following standard settings [1]. We also train HiDDeN [11] to obtain the pre-trained 2D message decoder. In the second stage, we apply our proposed uncertainty-aware watermarking method to generate GaussianMarker $\tilde{\mathcal{G}}$ via Equation (7). We then freeze the original Gaussians $\mathcal{G}$ and use the pre-trained 2D message decoder to supervise the training of our GaussianMarker $\tilde{\mathcal{G}}$. We then train our 3D message decoder to retrieve the copyright message from the watermarked 3D Gaussians $\hat{\mathcal{G}}$. We apply several types of 2D and 3D distortion layers on the watermarked 2D images and 3D Gaussians to achieve robustness. We use the default optimization setting in 3DGS [1] to optimize our GaussianMarker $\tilde{\mathcal{G}}$. We use the Adam optimizer [24] to optimize the 3D message decoder D_ψ and classifier D_ξ with default values $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$, and a learning rate 1×10^{-4} that decays following the exponential scheduler during optimization. We set $\lambda_1 = 10.0, \lambda_2 = 1.0$ for 2D watermarking loss $\mathcal{L}_{2D}$ and $\lambda_1^{'} = 2.0, \lambda_2^{'} = 1.0$ for 3D watermarking loss $\mathcal{L}_{3D}$ to adapt the training losses. The training takes 1000 (Blender, LLFF) or 2000 steps (MipNeRF360) and can finish within 20 minutes using a single NVIDIA V100 GPU.

Baselines. We design experiments to validate the message extraction on both rendered 2D images and 3D Gaussian parameters, demonstrating the effectiveness of our proposed method. For 2D message extraction, we compare our proposed method with four baselines for a fair comparison: 1) **CopyRNeRF**[4]: A state-of-the-art method for protecting the copyright of NeRF [7] by using watermarked color representation; 2) **HiDDeN** [11] + **3DGS** [1]: Preprocessing images with the classical image watermarking method HiDDeN [11] before the training of 3DGS [1]; 3) **3DGS with message:** Creating message embedding by MLPs and concatenating the message embedding with 3D Gaussian parameters; 4) **3DGS with fine-tuning:** Fine-tuning all of the 3D Gaussian parameters for embedding copyright messages. For 3D message extraction, since NeRF watermarking methods do not have explicit 3D parameters, we compare our methods with the 3DGS baselines, including HiDDeN + 3DGS, 3DGS with message, and 3DGS with fine-tuning.

Evaluation methodology. We evaluate the performance of our proposed method by comparing it with other digital watermarking baselines using the standard of capacity, invisibility, and robustness for both 2D images and 3D Gaussians. For *capacity*, we set the bit length of copyright messages to 48 bits, aligning with the maximum length previously employed in 3D model watermarking methods [41, 4]. For *invisibility*, we evaluate the reconstruction quality with PSNR, SSIM, and LPIPS [55] for 2D images, and we evaluate geometry difference with the $\mathcal{L}_1$ norm of position difference ($\mathcal{L}_1 Diff$), and signal-to-noise ratio (SNR) for 3D Gaussian positions. For *robustness*, we evaluate whether the copyright messages in 2D images can remain consistent against various distortions, including 2D Gaussian noise, JPEG compression, scaling, and Gaussian blur. We also evaluate whether the copyright messages in 3D Gaussians can remain consistent against various 3D attacks, including 3D Gaussian noise, translation, rotation, and crop-out.

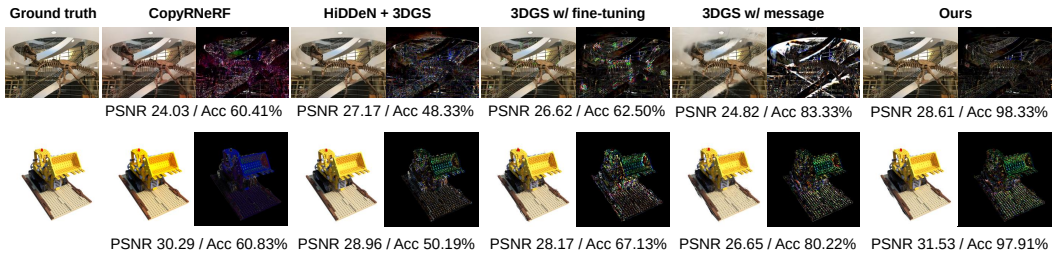


Figure 4: Comparisons between each baseline and our proposed method. We display the differences ($\times 10$) between the synthesized results and the ground truth for each method. Our proposed GaussianMarker demonstrates superior reconstruction quality and bit accuracy.

https://doi.org/10.52202/079017-1040

Dataset	Method	PSNR/SSIM↑	LPIPS↓	Bit accuracy ↑ (%) None	Noise ($\nu = 0.1$)	JPEG ($Q = 50$)	Scaling ($s \leq 25\%$)	Blur ($\xi = 0.1$)
Blender	CopyRNeRF [4]	30.29/0.8878	0.0813	60.83	59.92	58.52	57.44	60.22
	HiDDeN [11] + 3DGS [1]	28.96/0.8812	0.0829	50.19	49.84	50.12	50.09	50.16
	3DGS [1] w/ messages	22.65/0.8066	0.1584	80.22	78.66	75.80	78.08	79.64
	3DGS [1] w/ fine-tuning	28.17/0.9047	0.0878	67.13	67.06	63.43	64.04	66.38
	Ours	**31.53//0.9082**	**0.0759**	**97.91**	**96.93**	**91.66**	**96.17**	**97.43**
LLFF	CopyRNeRF [4]	24.03/0.7747	0.2575	60.77	60.23	58.06	58.89	60.35
	HiDDeN [11] + 3DGS [1]	27.17/0.8543	0.1210	48.26	48.14	46.26	46.89	48.12
	3DGS [1] w/ messages	24.82/0.8452	0.1310	83.33	82.39	79.17	81.04	83.18
	3DGS [1] w/ fine-tuning	26.62/0.8566	0.1117	60.61	59.99	55.49	57.52	60.40
	Ours	**28.61/0.8930**	**0.0999**	**98.33**	**97.83**	**91.45**	**95.89**	**98.23**
MipNeRF360	CopyRNeRF [4]	22.47/0.8053	0.4825	58.55	57.22	55.26	55.80	57.59
	HiDDeN [11] + 3DGS [1]	27.20/0.8151	0.2143	48.75	48.03	45.93	47.75	48.56
	3DGS [1] w/ messages	24.84/0.7992	0.1705	77.08	76.75	74.26	75.54	77.00
	3DGS [1] w/ fine-tuning	27.04/0.8452	0.1357	61.67	61.45	59.94	60.56	61.51
	Ours	**29.16/0.8808**	**0.1197**	**97.32**	**97.01**	**90.77**	**95.32**	**97.18**

Table 1: Reconstruction qualities and bit accuracy compared with different baselines. PSNR/SSIM and LPIPS are computed between the original and watermarked rendered images. The results are computed on the average of all examples.

Method	Geometry difference $\mathcal{L}_1 Diff \downarrow$	SNR↑	Bit accuracy ↑ (%) None	Noise ($\sigma = 0.1$)	Translation ($t = [0, 1000]^3$)	Rotation ($r = \pm\pi/6$)	Cropout ($cr = 0.1$)
HiDDeN [11] + 3DGS [1]	0.00912	40.90	68.20	67.65	67.32	66.67	64.24
3DGS [1] w/ messages	0.10513	32.93	85.41	84.91	85.35	81.52	79.57
3DGS [1] w/ fine-tuning	0.01829	37.24	69.79	69.70	68.78	65.88	64.84
Ours w/ 2D decoder	**0.00003**	**43.23**	97.85	57.05	59.07	53.88	48.23
Ours w/ 3D decoder	**0.00003**	**43.23**	**100**	**99.91**	**98.95**	**95.83**	**92.70**

Table 2: Geometry difference and bit accuracy compared with different baselines. $\mathcal{L}_1$ distance and SNR are computed between the original and watermarked 3D Gaussians. The results are computed on the average of all examples from Blender, LLFF, and MipNeRF360.

5.2 Experimental results

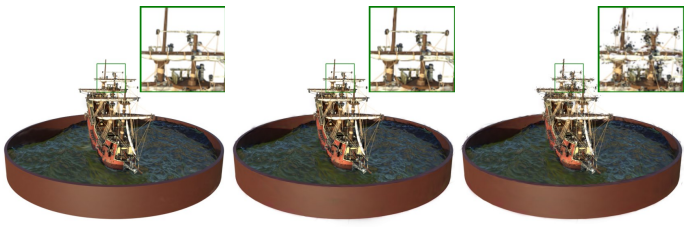
Messages extraction with 2D images. We compare the reconstruction qualities and bit accuracies with all baselines, and the qualitative and quantitative results are shown in Figure 4 and Table 1. CopyRNeRF [4] can limitedly extract hidden messages from the renderings and show undermined robustness to different image distortions. Although HiDDeN [11] + 3DGS [1] can achieve high reconstruction quality, it fails to extract the copyright messages from the rendered 2D images. This result is aligned with the previous method [4] and proves the message can not be transmitted from the 2D images into the 3D Gaussians. 3DGS [1] with messages directly embed copyright messages into 3D Gaussian parameters. It can retrieve the copyright message with relatively high accuracy, but the reconstruction quality is poor and shows obvious distortions in the rendered images. 3DGS [1] with fine-tuning shows better reconstruction quality, but the message extraction accuracy is limited. This is because 3D Gaussians usually contain millions of parameters, and directly fine-tuning all of the parameters without proper regularization can be less effective for message extraction. Our method can achieve both high reconstruction quality and high decoding accuracy. Even with different distortions to the rendered images, our method can still achieve high decoding accuracy to reliably safeguard the 3DGS models.

Messages extraction with 3D Gaussians. We evaluate the geometry differences and the bit accuracies with all 3DGS baselines, and the results are shown in Table 2. HiDDeN [11] + 3DGS [1] has small geometry difference, but it has limited message decoding accuracy. 3DGS with messages shows reasonable message decoding accuracy, but it displays high geometry differences. 3DGS with fine-tuning shows a relatively small geometry difference, but it struggles to decode the message. Our method has the smallest geometry difference, obtains accurate decoding accuracy, and shows robustness to different 3D attacks. Furthermore, 3DGS can be easily edited in the 3D space to influence the rendered 2D images. We conduct experiments to apply 3D attacks and then render the 2D images. As shown in Table 2, the 3D attacks can easily fool the 2D message decoder, while our 3D message decoder is robust to such 3D attacks and can reliably extract copyright messages.

https://doi.org/10.52202/079017-1040

5.3 Ablation study

Perturb low-uncertainty Gaussians. We design ablation experiments to add perturbations into 3D Gaussian parameters with low uncertainty. We show qualitative results in Figure 5, the low-uncertainty Gaussians corresponding to the fine details in the 3D scene, such as the halyard on the ship. Adding perturbations to these areas can easily make the perturbation visible, thus undermining the image quality. We evaluate the quantitative results in Table 3. Adding perturbations to 3D Gaussian parameters with low uncertainty can easily undermine the reconstruction quality and degrade the decoding accuracy. Our method preserves 3D Gaussians with low uncertainty, maintaining the geometric structure to ensure imperceptible perturbations and high message extraction accuracy.



(a) Origin (b) High-uncertainty (c) Low-uncertainty

Figure 5: Qualitative results of applying perturbation into (b) high uncertainty Gaussians and (c) low uncertainty Gaussians.

Metric	Low	High
PSNR	26.61	28.71
SSIM	0.8782	0.9016
LPIPS	0.0948	0.0763
Acc	76.56	97.07
Noise	75.51	95.83
Blur	76.27	96.30
Resize	73.92	93.22

Table 3: Quantitative results of adding perturbation into low-uncertainty Gaussians and high-uncertainty Gaussians.

6 Conclusion

In conclusion, protecting the copyright of 3D Gaussian Splatting (3DGS) assets is crucial due to their vulnerability to unauthorized distribution and manipulation. Existing methods for copyright protection in the radiance field are not directly applicable to 3DGS. Our proposed method involves using uncertainty estimation to add invisible 3D perturbations to the 3D Gaussian parameters, ensuring both invisibility and robustness. Overall, our proposed approach introduces an effective solution with a positive societal impact on the copyright protection of the 3DGS models.

Limitations. Our method is an effective technical solution for the copyright protection of 3DGS models. However, as we discussed before, our mechanism may still face threats from some malicious operations. More measures should be implemented for such malicious attacks beyond the technology. Furthermore, we will explore enhancing the robustness of GaussianMarker in dynamic 3DGS scenarios, via the motion transfer-based data augmentation approach, to maintain high bit accuracies while improving robustness [56] in future work.

7 Acknowledgement

This work was done at Renjie's Research Group at the Department of Computer Science of Hong Kong Baptist University. Renjie's Research Group is supported by the National Natural Science Foundation of China under Grant No. 62302415, Guangdong Basic and Applied Basic Research Foundation under Grant No. 2022A1515110692, 2024A1515012822, and the Blue Sky Research Fund of HKBU under Grant No. BSRF/21-22/16. This work was supported in part by the RGC Senior Research Fellow Scheme under the grant: SRFS2324-2S02.

References

[1] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics (TOG)*, 2023.

[2] Jonathon Luiten Georgios Kopanas, Bernhard Kerbl and Antoine Guedon. 3D Gaussian Splatting 3DV Tutorial. `https://3dgstutorial.github.io/3dv_part1.pdf`, 2024.

[3] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. GaussianEditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[4] Ziyuan Luo, Qing Guo, Ka Chun Cheung, Simon See, and Renjie Wan. CopyRNeRF: Protecting the CopyRight of Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[5] Mahdi Ahmadi, Alireza Norouzi, Nader Karimi, Shadrokh Samavi, and Ali Emami. ReDMark: Framework for Residual Diffusion Watermarking based on Deep Networks. *Expert Systems with Applications*, 2020.

[6] Chenxin Li, Brandon Y Feng, Zhiwen Fan, Panwang Pan, and Zhangyang Wang. StegaNeRF: Embedding Invisible Information within Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[7] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Communications of the ACM*, 2021.

[8] Wen Jiang, Boshu Lei, and Kostas Daniilidis. FisherRF: Active View Selection and Uncertainty Quantification for Radiance Fields using Fisher Information. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2024.

[9] Christopher M Bishop. Pattern Recognition and Machine Learning. *Springer google scholar*, 2:1122–1128, 2006.

[10] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet:Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] 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)*, 2018.

[12] Chengxuan Zhu, Renjie Wan, and Boxin Shi. Neural Transmitted Radiance Fields. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[13] Chengxuan Zhu, Renjie Wan, Yunkai Tang, and Boxin Shi. Occlusion-Free Scene Recovery via Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[14] Ziyuan Luo, Boxin Shi, Haoliang Li, and Renjie Wan. Imaging Interiors: An Implicit Solution to Electromagnetic Inverse Scattering Problems. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2024.

[15] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian Splatting. *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[16] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. In *Proceedings of International Conference on 3D Vision (3DV)*, 2024.

[17] Antoine Guédon and Vincent Lepetit. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[18] Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3DGS-Avatar: Animatable Avatars via Deformable 3D Gaussian Splatting. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[19] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 2015.

 https://doi.org/10.52202/079017-1040

[20] Ron G Van Schyndel, Andrew Z Tirkel, and Charles F Osborne. A digital watermark. In *Proceedings of International Conference on Image Processing (ICIP)*, 1994.

[21] Chih-Chin Lai and Cheng-Chih Tsai. Digital Image Watermarking Using Discrete Wavelet Transform and Singular Value Decomposition. *IEEE Transactions on Instrumentation and Measurement*, 2010.

[22] Thottempudi Pardhu and Bhaskara Rao Perli. Digital image watermarking in frequency domain. In *Proceedings of International Conference on Communication and Signal Processing (ICCSP)*, 2016.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Zhikai Hu, Yiu-ming Cheung, Mengke Li, and Weichao Lan. Cross-modal hashing method with properties of hamming space: A new perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2024.

[26] Mengke Li, Yiu-Ming Cheung, and Zhikai Hu. Key point sensitive loss for long-tailed visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022.

[27] Matthew Tancik, Ben Mildenhall, and Ren Ng. StegaStamp: Invisible Hyperlinks in Physical Photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[28] Xinyu Weng, Yongzhi Li, Lu Chi, and Yadong Mu. High-Capacity Convolutional Video Steganography with Temporal Residual Modeling. In *Proceedings of the International Conference on Multimedia Retrieval (ICMR)*, 2019.

[29] Eric Wengrowski and Kristin Dana. Light Field Messaging with Deep Photographic Steganography. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] Peng Yang, Yingjie Lao, and Ping Li. Robust watermarking for deep neural networks via bi-level optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[31] Chaoning Zhang, Philipp Benz, Adil Karjauv, Geng Sun, and In So Kweon. UDH: Universal deep hiding for steganography, watermarking, and light field messaging. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

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

[33] Ruofei Wang, Renjie Wan, Zongyu Guo, Qing Guo, and Rui Huang. Spy-Watermark: Robust Invisible Watermarking for Backdoor Attack. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[34] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[35] Ruofei Wang, Qing Cuo, Haoliang Li, and Renjie Wan. Event Trojan: Asynchronous Event-based Backdoor Attacks. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2024.

[36] KevinAlex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv: Multimedia,arXiv: Multimedia*, 2019.

[37] Emil Praun, Hugues Hoppe, and Adam Finkelstein. Robust mesh watermarking. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (PACMCGIT)*, 1999.

[38] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[39] Xingyu Chen, Yu Deng, and Baoyuan Wang. Mimic3D: Thriving 3D-Aware GANs via 3D-to-2D Imitation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[40] Jeongho Son, Dongkyu Kim, Hak-Yeol Choi, Han-Ul Jang, and Sunghee Choi. Perceptual 3D Watermarking Using Mesh Saliency. In *Proceedings of International Conference on Information Science and Applications (ICISA)*, 2017.

[41] Innfarn Yoo, Huiwen Chang, Xiyang Luo, Ondrej Stava, Ce Liu, Peyman Milanfar, and Feng Yang. Deep 3D-to-2D Watermarking: Embedding Messages in 3D Meshes and Extracting Them from 2D Renderings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[42] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[43] Yinghao Xu, Sida Peng, Ceyuan Yang, Yujun Shen, and Bolei Zhou. 3D-aware Image Synthesis via Learning Structural and Textural Representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[44] Qi Song, Ziyuan Luo, Ka Chun Cheung, Simon See, and Renjie Wan. Protecting NeRFs' Copyright via Plug-And-Play Watermarking Base Model. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2024.

[45] Xiufeng Huang, Ka Chun Cheung, Simon See, and Renjie Wan. GeometrySticker: Enabling ownership claim of recolorized neural radiance fields. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2024.

[46] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Transactions on Graphics (TOG)*, 2006.

[47] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization (VIS)*, 2001.

[48] Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. Neural point catacaustics for novel-view synthesis of reflections. *ACM Transactions on Graphics (TOG)*, 2022.

[49] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 2019.

[50] M.P. Deisenroth, A.A. Faisal, and C.S. Ong. *Mathematics for Machine Learning*. 2020.

[51] Andreas Kirsch and Yarin Gal. Unifying approaches in active learning and active sampling via fisher information and information-theoretic quantities. *Transactions on Machine Learning Research (TMLR)*, 2022.

[52] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing NeRF for Editing via Feature Field Distillation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[53] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[54] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

 https://doi.org/10.52202/079017-1040

[55] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proceeding of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[56] Xiu-Feng Huang, Lai-Man Po, and Wei-Feng Ou. Motion transfer-driven intra-class data augmentation for finger vein recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[57] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. *arXiv preprint arXiv:2311.17245*, 2023.

A Additional visualization

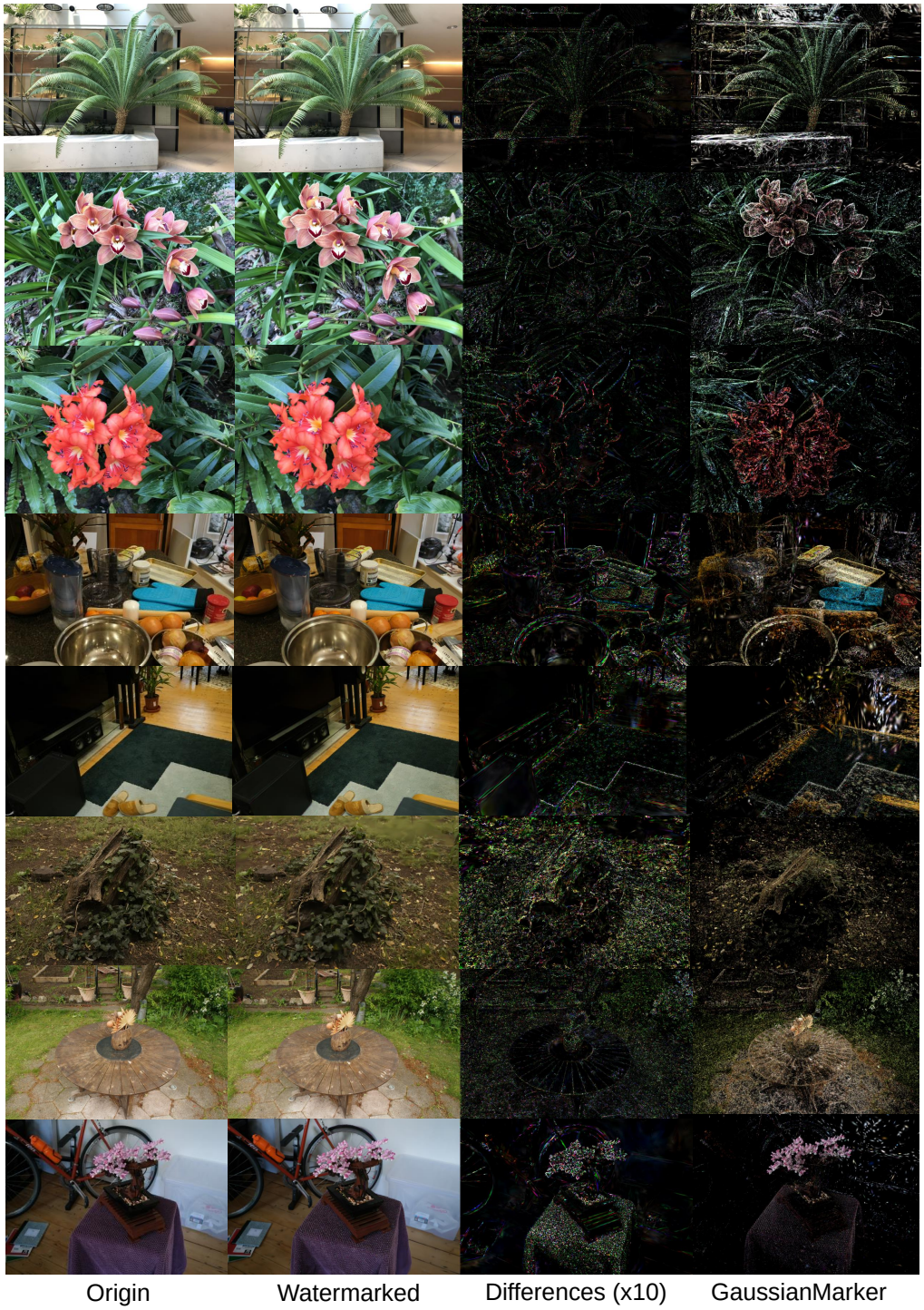


Figure 6: Visualization of our proposed method performance in LLFF and MipNeRF360 datasets. In each line, we display the original rendered image, the watermarked rendered image, the difference ($\times 10$) between the watermarked and original rendered images, and our proposed GaussianMarker as 3D perturbations for copyright message embedding. The scalings of GaussianMarker are adjusted for better visualization.

https://doi.org/10.52202/079017-1040

Figure 7: Visualization of our proposed method performance in Blender dataset.

B Additional quantitative results for real-world datasets

LLFF	PSNR	SSIM	LPIPS	Acc	Noise	JPEG	Scale	Blur
Fern	29.40	0.92	0.0867	1.0	1.0	0.8916	0.9791	1.0
Fortress	30.91	0.9084	0.1227	0.9375	0.9375	0.9166	0.9375	0.9375
Horn	27.46	0.8849	0.1094	0.9791	0.9791	0.9125	0.8958	0.9791
Orchids	25.177	0.8543	0.0853	1.0	1.0	0.9333	0.95833	1.0
Flower	30.09	0.8939	0.0952	1.0	1.0	0.9125	0.9791	1.0
Trex	28.165	0.9109	0.0734	1.0	1.0	0.8541	0.9791	1.0

Table 4: Quantative results on LLFF scenes.

MipNeRF360	PSNR	SSIM	LPIPS	Acc	Noise	JPEG	Scale	Blur
Stump	28.95	0.8521	0.1364	0.9791	0.9791	0.8291	0.9375	0.9791
Bicycle	25.68	0.8028	0.1774	0.9583	0.9583	0.8750	0.9125	0.9583
Kitchen	29.87	0.8937	0.0946	0.9791	0.9791	0.8333	0.9583	0.9791
Counter	28.68	0.8948	0.1300	0.9583	0.9583	0.8916	0.9166	0.9583
Bonsai	30.41	0.9249	0.1002	0.9583	0.9583	0.9125	0.8958	0.9583
Garden	28.58	0.8825	0.0760	1.0	1.0	0.9125	0.9375	1.0
Room	31.97	0.9149	0.1232	0.9791	0.9791	0.8750	0.9583	0.9791

Table 5: Quantative results on MipNeRF360 scenes.

C The correlation between uncertainty and image watermarking

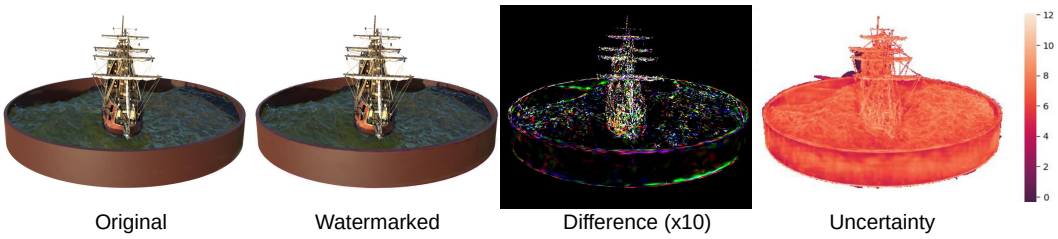


Figure 8: Uncertainty heatmap visualization.

The uncertainty estimation in the 3DGS model inherently identifies model parameters that are more robust to perturbations, making it highly suitable for the application of invisible watermarking in 3DGS. Additionally, the HiDDeN decoder [11], primarily focuses on decoding information along the boundary regions. As illustrated in Figure 8, these boundary regions exhibit high uncertainty values. This observation demonstrates the correlation between uncertainty and message embedding, highlighting how areas of high uncertainty can be leveraged for effective watermarking.

D The influence of uncertainty threshold

In our experiments, we set the average uncertainty value as the default threshold. We show more results to verify the influence of the uncertainty threshold. As shown Table 6, we select the ship scene in the Blender dataset to study the influence of the uncertainty threshold. A lower threshold can enhance the bit accuracy, though it slightly compromises image quality. Conversely, a higher threshold results in better image quality and a more lightweight model but also slightly compromising message decoding accuracy. However, in both situations, the compromises are moderate.

It is noteworthy that our method is also compatible with compressing the 3DGS model based on the uncertainty value, similar to how LightGaussian compresses 3DGS using importance values [57].

https://doi.org/10.52202/079017-1040

Threshold	Original points	Perturbation points	PSNR	SSIM	LPIPS	ACC
average $\times$ 3.7	340k	13k	26.44	0.9521	0.0396	92.08%
average $\times$ 1.0	340k	27k	26.37	0.9498	0.0397	95.21%
average $\times$ 0.24	340k	54k	26.31	0.9490	0.0399	96.88%
average $\times$ 0.13	340k	108k	26.17	0.9483	0.0399	96.31%

Table 6: The influence of uncertainty threshold.

This compatibility highlights that our approach can work seamlessly with existing 3DGS model compression techniques, effectively mitigating the impact of the increasing number of 3D Gaussians in our method.

E The geometry consistency

Figure 9: Visualization of our proposed GaussianMarker in MipNeRF360 **Room** scene. We select four camera angles which are never used in the training dataset as the font, left, back, right views of the scene to represent the multi-view consistency of incorporated perturbations.

Our method embeds watermarks into the 3DGS model by adding perturbations to 3D Gaussians with high uncertainty. As shown in Figure 8, these areas cover most object boundaries in the 3DGS scene. Figure 9 further illustrates geometry consistency by displaying the incorporated perturbations. These perturbations effectively cover the scene's general geometric structure. The geometry of these perturbations remains consistent across different camera angles and can be transmitted into the rendered images, which is essential for robust extraction from different viewing angles.

F Time analysis

Datasets	3DGS training	Our message embedding
Synthetic datasets (Blender)	30k steps / 30mins	1k steps / 3 mins
Real-world scenes (LLFF)	30k steps / 40mins	1k steps / 5 mins
Real-world scenes (MipNeRF360)	30k steps / 45mins	2k steps / 10 mins

Table 7: Time analysis on different datasets.

We present a time analysis of our method's training efficiency in Table 7. Compared to the original 3DGS model training, our method requires only 1,000 to 2,000 steps within a span of 10 minutes. This demonstrates that our approach is not only efficient but also practical for watermarking 3DGS models.

https://doi.org/10.52202/079017-1040

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction include the claims in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the main claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We create a "Limitation" section in the Conclusion to discuss the limitations of our work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

https://doi.org/10.52202/079017-1040

Justification: We provide complete proof for theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the paper's theorems, formulas, and proofs should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all information for reproduce the experimental results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide code implementation in our supplementary materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, our paper specifies all settings for training and testing.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We don't think error bars are necessary for our experimental results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

https://doi.org/10.52202/079017-1040

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We list our training resources and training time.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We strictly follow the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the societal impact in the conclusion and supplementary.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We have described all data and models we use.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all referenced code, data and models.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

https://doi.org/10.52202/079017-1040

Answer: [NA]

Justification: Our work does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.