Protecting NeRFs' Copyright via Plug-And-Play Watermarking Base Model

Qi Song[1,2], Ziyuan Luo[1,2], Ka Chun Cheung[2],
Simon See[2] and Renjie Wan[1⋆]

[1] Department of Computer Science, Hong Kong Baptist University
[2] NVIDIA AI Technology Center, NVIDIA
{qisong,ziyuanluo}@life.hkbu.edu.hk {chcheung, ssee}@nvidia.com
renjiewan@hkbu.edu.hk

A Overview

This supplementary material provides additional discussions, implementation details, and further results complementing the paper:

- Sec. B presents an overview of our method's ownership verification process.
- Sec. C discusses the novelty of our proposed method, highlighting the key features and advantages that distinguish it from the existing approach [5].
- Sec. D provides a detailed description of the implementation of our method.
- Sec. E introduces the hyperparameter λ_3 for balancing representation and message distilling in Eq. 9.

B Overview of ownership verification.

In the application scenario, NeRF creators can freely choose one NeRF variant [1, 2, 7] and embed the watermark information during the creation process with our proposed method. Our method provides flexible copyright protection solutions for NeRF creators. Then, the watermarking base model can be directly utilized to extract the message from rendered views, even when faced with different image distortions. Creators can conveniently integrate our watermark embedding method to effectively prevent unauthorized rendered results from being misused. This flexibility helps our method gain wider application in the NeRF ecosystem, meeting the needs of different NeRF creators.

C Uniqueness of our method

Our method is dedicated to distilling copyright messages into NeRF without altering its core architecture, enabling seamless integration with various NeRF

⋆ Corresponding author.

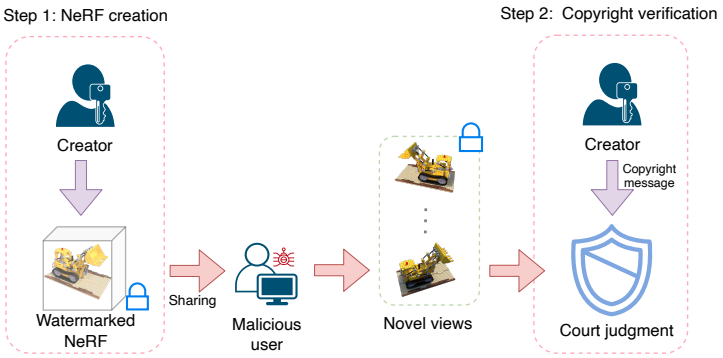

Fig. S1: The ownership verification process envisioned in our scenario involves two key steps: NeRF creation (Step 1) and copyright verification (Step 2). In Step 1, the owner employs a representation scheme (*e.g.*, Instant-NGP [7], TensorRF [1] and so on) to build the NeRF representation. Then, they can directly create a watermarked NeRF. In Step 2, if malicious users steal the NeRF representation, the NeRF creators can extract binary watermarks from rendered images for the ownership claim.

variants, including Instant-NGP [7], TensorRF [1], and Plenoxels [2]. This plug-and-play property showcases its high scalability (as illustrated in Fig. S2). As for CopyRNeRF [5], it modifies the NeRF architecture (Fig. S3) with dedicated message modules, resulting in incompatible with mainstream NeRF implementations. Consequently, NeRF creators lack the freedom to select their preferred NeRF architecture if they wish to claim ownership. In contrast, our approach empowers NeRF creators to choose their desired NeRF variants while still retaining the ability to claim ownership, as our method does not modify the core NeRF architecture. Moreover, unlike CopyRNeRF [5], our method embeds watermarks during the NeRF creation process, minimizing the window of opportunity for malicious actors to misuse the NeRF.

D Implementation details

D.1 Details on building watermarking base model

The message extractor of HiDDeN [10] is used as our watermarking base model. We keep the same architecture and training settings as in HiDDeN [10]. The code for training HiDDeN [10] can be found in the link below:
`https://github.com/ando-khachatryan/HiDDeN`.

Architecture of HiDDeN [10]. HiDDeN [10] contains an encoder for message embedding and a decoder for message extraction. The encoder comprises four Conv-BN-ReLU blocks, each equipped with 64 output filters, 3×3 kernels, a stride of 1, and padding of 1. The extractor consists of seven blocks, followed by a block with k output filters (where k denotes the number of bits

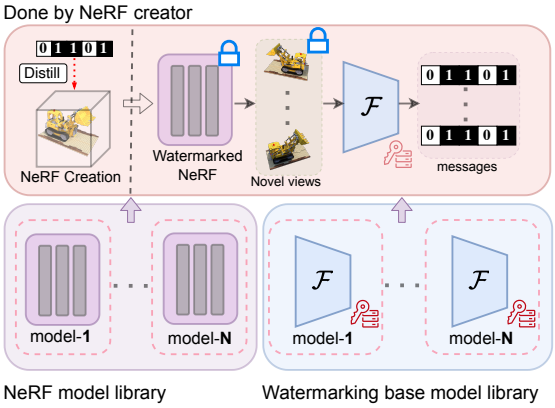

Fig. S2: The proposed creation process of our method. First, NeRF creators can choose a **NeRF variant** (*e.g.*, Instant-NGP [7], TensorRF [1], and Plenoxels [2] and so on). Second, NeRF creators can acquire a pre-trained watermarking base model $\mathcal{F}$.(*e.g.*, HiDDeN [10], MBRS [3] and so on), sourced from a third party (*e.g.*, open-source library) or train a message extractor separately via standard pipelines. This base model is considered to be "**plug-and-play**" in our scenario. During NeRFs' creation, creators can seamlessly integrate a base model to embed watermarks within their NeRFs. Upon completion of the NeRF optimization, they obtain a watermarked NeRF. Once this watermarked NeRF is distributed publicly, the creators can utilize the same base model to retrieve binary messages from newly rendered views, thereby asserting their ownership over the content.

to be concealed), an average pooling layer, and a $k \times k$ linear layer. For a more comprehensive understanding of the architecture, we recommend referring to the original paper [10].

Image distortion layer. The Image distortion layer is responsible for generating operated versions of the watermarked image in order to enhance its robustness against image processing and distortions. In our experiments, the distortion layer employs random operations, including cropping, resizing, and JPEG compression. The parameters for cropping or resizing are chosen randomly from 0.3 to 0.7. Additionally, there is a 50% probability of applying JPEG compression, with the compression parameter set 50 to 80.

Optimization. We train HiDDeN [10] on the MS-COCO dataset [4].The size of training images is set to 256×256. The number of bits k is set to 48, and a scaling factor of $\alpha = 0.3$. The optimization process took approximately a day and is performed on 8 GPUs using the Lamb optimizer [9]. Each GPU has a

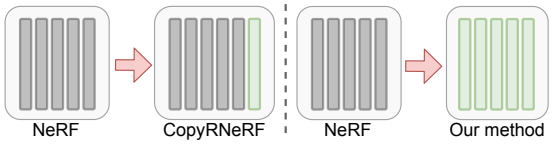

Fig. S3: Comparison between our method and CopyRNeRF [5]. To embed messages into NeRF, CopyRNeRF [5] modifies the representation scheme by appending additional message modules. Different from CopyRNeRF [5], our method is able to embed watermarks directly into the core representation of NeRF.

batch size of 64. We employ a cosine annealing schedule for the learning rate, starting with a linear warmup of 5 epochs to 1×10^{-2}, and then decaying to 1×10^{-6}.

D.2 Details on message distillation

With the watermarking base model $\mathcal{F}$, we then distill message patterns into NeRF during its creation.

For each scene, a 48-bit message $\mathbf{m}$ is randomly selected. In every training iteration, a camera pose corresponding to the training view is chosen, and a global progressive rendering strategy is used. The content loss $\mathcal{L}_{\text{local}}$ and the similarity loss $\mathcal{L}_{\text{inv}}$ between the rendering image and its corresponding ground truth are calculated. $\mathcal{L}_{\text{dis}}$ is calculated by the extracted messages set from these multi-layered representations and the select message $\mathbf{m}$. The distilling loss $\mathcal{L}_{\text{dis}}$ is the binary cross-entropy loss between embedded message $\mathbf{m}$ and the extracted message $\hat{\mathbf{m}}$. Additionally, we also find that only a few extra minutes are needed to achieve effective message embedding and high-quality scene representation. Thus, our message embedding also does not significantly change the time needed to create NeRF. After the creation of NeRF, the copyright message $\mathbf{m} \in \{0,1\}^{48}$ can be extracted from every view rendered by using our watermarking base model. The results show that our proposed method can achieve a good balance between bit accuracy and representation performance.

D.3 Robustness to image transformations

In Section 5.2, we evaluate the robustness of the watermark extraction to various distortions and transformations. These transformations simulate common image processing steps found in image editing software. The transformations are illustrated in Fig. S4. For the crop and resize transformation, the parameter is the ratio of the new area to the original area. For JPEG compression, the parameter is the quality factor, where a higher value (usually 90% or higher) represents high quality; value $(80\% - 90\%)$ and $(70\% - 80\%)$ represent medium quality and low quality, respectively. For brightness, contrast, saturation, and sharpness, the parameter represents the default factor used in the PIL and Torchvision [6] libraries. Text overlay is achieved through the AugLy library [8], which adds text to the image at a random position.

Table S1: Representation and bit accuracy trade-off during creation. our method sets $\lambda_3 = 0.001$ to achieve the best balance in NeRF representation and watermarking performance.

λ_3	0.1	0.01	0.001	0.0001
PSNR ↑	27.91	30.97	31.61	32.74
Bit acc. % ↑	95.15	94.15	93.41	72.60

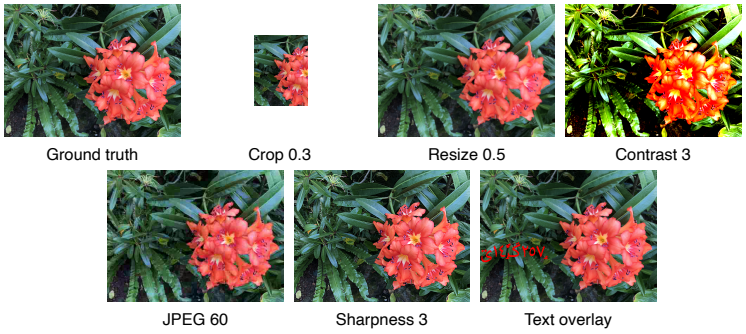

Fig. S4: Visual illustration of transformations evaluated in section 5.2

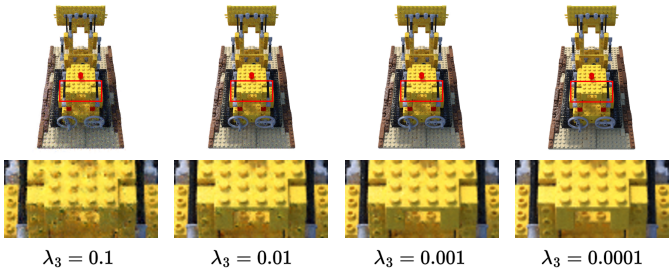

Fig. S5: Visual results of different balance values λ_3. As the decrease of λ_3, the quality of reconstructed scene representation becomes better.

E Additional experiment results

Effects of λ_3 in Eq. 9. Balance hyperparameter λ_3 is able to balance the message embedding and representation quality. We conduct experiments in evaluating the effectiveness of λ_3. We report the average PSNR and Bit accuracy on the messages extracted from test views. As shown in Table S1 and Fig. S5, a higher λ_3 leads to higher bit accuracy of the extracted message but lower quality of scene representation. In our experiment, we set λ_3 to 0.001 as it well balances message embedding and representation quality.

References

1. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: TensoRF: Tensorial radiance fields. In: ECCV (2022)
2. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR (2022)
3. Jia, Z., Fang, H., Zhang, W.: MBRS: Enhancing robustness of dnn-based watermarking by mini-batch of real and simulated jpeg compression. In: ACM MM (2021)

4. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV (2014)
5. Luo, Z., Guo, Q., Cheung, K.C., See, S., Wan, R.: CopyRNeRF: Protecting the copyright of neural radiance fields. In: ICCV (2023)
6. maintainers, T., contributors: TorchVision: Pytorch's computer vision library. `https://github.com/pytorch/vision` (2016)
7. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. TOG (2022)
8. Papakipos, Z., Bitton, J.: AugLy: Data augmentations for robustness (2022)
9. You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., Hsieh, C.J.: Large batch optimization for deep learning: Training bert in 76 minutes. In: ICLR (2020)
10. Zhu, J., Kaplan, R., Johnson, J., Fei-Fei, L.: HiDDeN: Hiding data with deep networks. In: ECCV (2018)