WateRF: Robust Watermarks in Radiance Fields for Protection of Copyrights

Supplementary Material

Overview

This supplementary material introduces further details and experimental results of our proposed method, WateRF.

- Section A introduces a use case of our method.
- Section B explains the details about the pre-trained watermark decoder.
- Section C provides further experimental results, including patch-wise loss, robustness results on 500 messages, and quantitative results of additional message bits of 32 and 48 bits.

A. Use case of WateRF

This section aims to provide more details about the use case of WateRF. As a use case, 3D content creators can train an initial NeRF model, denoted as F_{θ_0}, with one's own dataset, to create a 3D scene or content with NeRF model. After the creator has finished training the NeRF model, the creator can transfer the trained NeRF model into a watermarked model, F_θ, by fine-tuning the trained NeRF model with our proposed method. The fine-tuning process ensures that all rendered novel viewpoint images from the F_θ contain watermarks. Even if the F_θ is shared as an open source or misappropriated by an user, the owner(creator) of the NeRF model can extract the message that represents copyright from the 2D images, rendered by the user from the NeRF model at every viewpoint. Furthermore, the pre-training process of the watermark decoder, which includes a distortion layer, allows for effective extraction of the message from images that have been subjected to attacks aiming to erase the watermark.

B. Details on pre-training decoder

B.1. Architectures and implementation details

Architectures. We use a CNN-based encoder and decoder, similar to the architecture of HiDDeN [7]. The encoder is composed of 4 Conv-BN-ReLU (CBR) blocks with 64 filters, 3 × 3 kernels, stride 1, and padding 1. The decoder consists of 7 CBR blocks with 64 filters. After the last CBR block, an average pooling layer and L × L linear layer, where L is message length, follows. The distortion layer has crop, scaling, and JPEG compression. For more details, we refer the reader to the original paper [7].

Implementation details. We train the HiDDeN [7] on MS-COCO dataset [2] with images having resolution of 256 × 256. The optimization is performed with Lamb optimizer [6] for 300 epochs on a single GPU. The parameters of the distortion layer are set to 0.3 or 0.7 for crop and scaling. The JPEG compression is set to 50 or 80. Since we only need a decoder, we set the message loss parameter to 1 and the image loss parameter to 0. Thus during the training process, only the message loss will be minimized. This Hidden [7] training only needs to be performed once per bit.

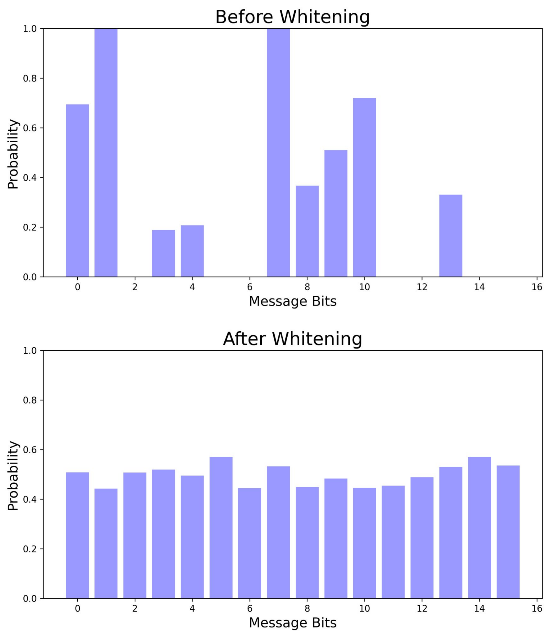

Figure 1. **Whitening effect.** These bar charts show the probability of decoding 1 from bits by a pre-trained decoder before and after whitening for NeRF datasets (Blender and LLFF).

B.3. Decoder with whitening

After training HiDDeN [7], we only use a decoder to fine-tune the model. We conduct PCA whitening to the linear layer. The input of the pre-trained decoder is a vanilla image, where no watermark is applied in the fine-tuning process. We remove the bias of decoded message bits. Before whitening in Fig. 1, the decoded bits have a bias where the probabilities are close to 1 or 0 appear only in certain positions. However, after whitening in Fig. 1, the probabilities of each bit become close to 0.5, and the bias is removed. Thus we can efficiently fine-tune for random message bits.

Methods		TensoRF [1]				NeRF [5]			
DWT	$\mathcal{L}_{patch}$	Bit Acc(%)↑	PSNR ↑	SSIM ↑	LPIPS ↓	Bit Acc(%)↑	PSNR ↑	SSIM ↑	LPIPS ↓
–	–	95.54	20.22	0.747	0.1518	77.39	23.92	0.899	0.0619
✓	–	95.56	27.80	0.896	0.0584	94.05	22.91	0.882	0.0784
–	✓	72.68	**32.80**	0.944	0.0392	71.54	27.43	0.933	0.0394
✓	✓	**95.67**	32.79	**0.948**	**0.0334**	**94.24**	**28.81**	**0.954**	**0.0252**

Table 1. Ablation study to see the effect of our proposed two main methods: Discrete Wavelet Transform (DWT) and patch-wise loss $\mathcal{L}_{patch}$. We evaluate bit accuracy, PSNR, SSIM, and LPIPS for 16 bits and every NeRF dataset (Blender and LLFF).

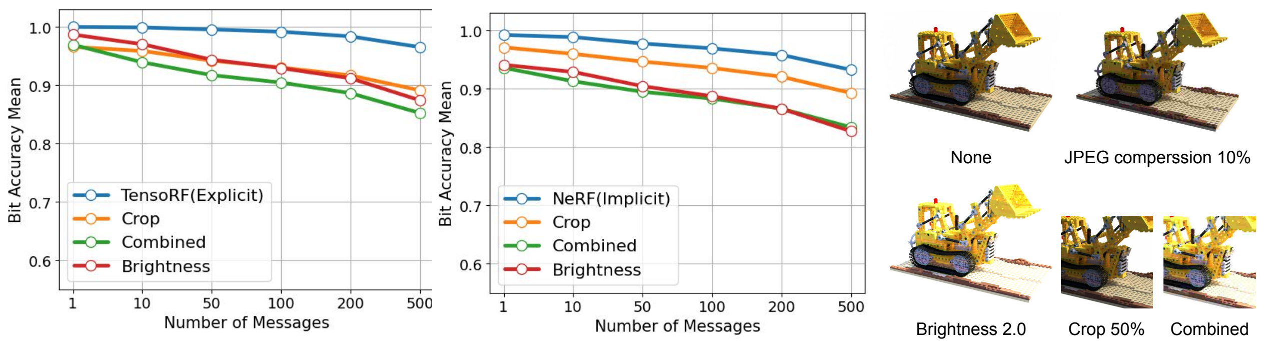

Figure 2. **Identification results.** Average of the bit accuracy for every 500 messages and representation type. The combined distortion consists of crop 50 %, JPEG compression 10 %, and brightness with factor 2.0. We show the results on 16 bits.

C. Additional results

C.1. Patch-wise loss

There exists a trade-off relationship between bit accuracy and rendering quality. In other words, the high bit accuracy brings the low rendering quality. Thus, it is necessary to adjust the trade-off. As shown in Tab. 1, our patch-wise loss improves the image quality while increasing bit accuracy.

C.2. The identification/detection experiment

This experiment is designed to verify if the NeRF model effectively fine-tunes for a single message and if the watermark survives combined distortion. For evaluation identification, we fine-tune 500 models with random messages. Each model generates 200 images at different viewpoints. For each of these 100K watermarked images, we decode the message and evaluate bit accuracy. In Fig. 2, we show that our methods effectively fine-tune NeRF for random messages.

C.3. Quantitative results for more bit lengths

We show results for more lengths of bits. The results of bit accuracy and reconstruction quality for 32 bits and 48 bits are shown in Tab. 2. We achieve state-of-the-art performance on longer message bits.

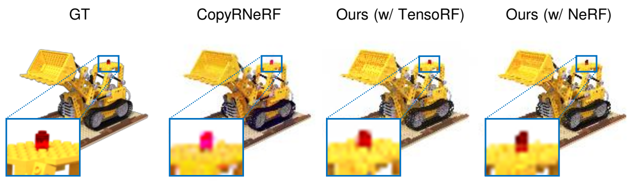

Figure 3. Comparisons on test-set views for scenes from Lego dataset. Our method preserves color value better than CopyRNeRF [3].

C.4. Qualitative results

In Fig. 3, We present an ablation study to show that our method can capture colors better than CopyRNeRF [3]. We provide all results rendered from original NeRF model, watermarked NeRF model (Fig. 4, 5, 6, 7), and the differences ($\times 10$) between the two images as an ablation study.

	32 bits				48 bits			
Methods	Bit Acc(%)↑	PSNR ↑	SSIM ↑	LPIPS ↓	Bit Acc(%)↑	PSNR ↑	SSIM ↑	LPIPS ↓
HiDDeN [7]+NeRF [5]	50.11	26.24	0.913	0.038	50.04	26.16	0.908	0.043
CopyRNeRF [3]	78.08	26.13	0.896	0.041	60.06	27.56	0.895	0.066
Ours (w/ NeRF [5])	86.81	27.20	0.944	0.033	70.43	28.35	0.925	0.037
Ours (w/ TensoRF [1])	**88.58**	**31.19**	**0.936**	**0.040**	**85.82**	**30.86**	**0.930**	**0.040**

Table 2. Bit accuracies and reconstruction qualities comparison with baselines. We show the results on 32 and 48 bits. The results are evaluated in the same way as baselines. The best performances are highlighted in **bold**.

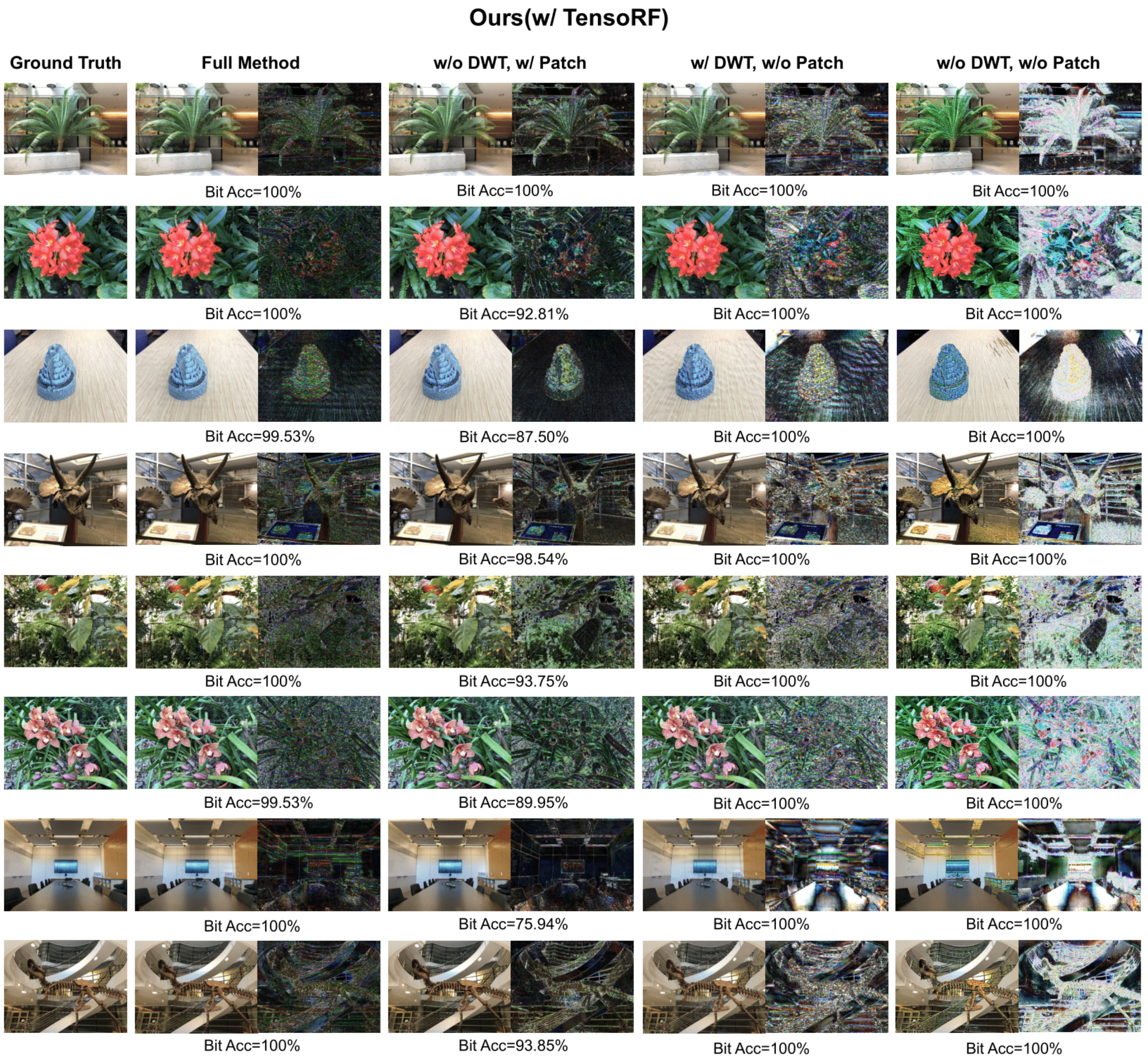

Figure 4. Reconstruction qualities of various rendering outputs using our method (with TensoRF[1]) for an ablation study on LLFF dataset[4]. We show the differences (×10) for the full method, our method without patch loss, and our method without frequency domain. The closer it is to white, the bigger the difference between the ground truth and the image. Our proposed WateRF can achieve a good balance between the rendering quality and bit accuracy. We show the results on 16 bits.

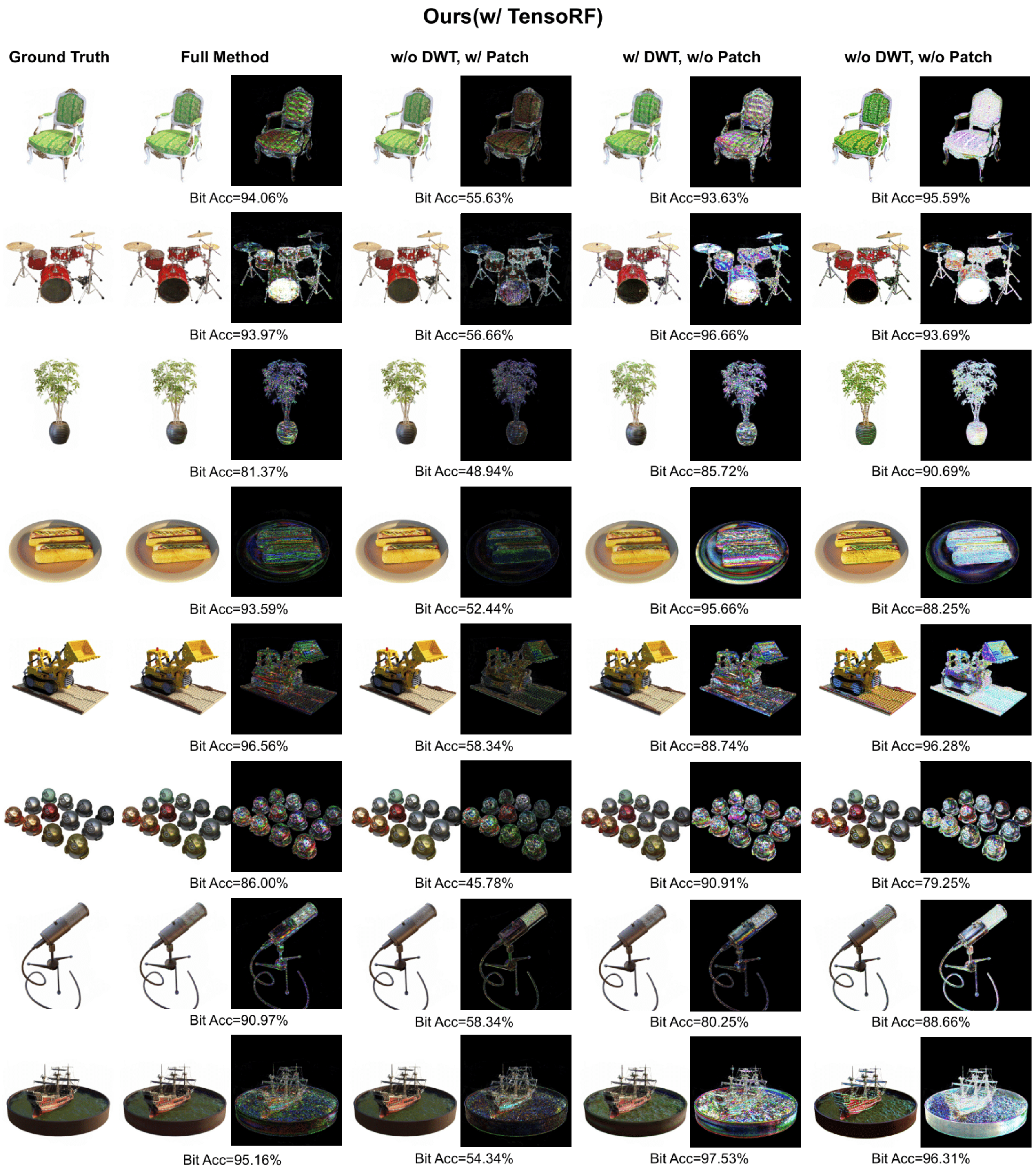

Figure 5. Reconstruction qualities of various rendering outputs using our method (with TensoRF[1]) for an ablation study on blender dataset[5]. We show the differences (×10) for the full method, our method without patch loss, and our method without frequency domain. The closer it is to white, the bigger the difference between the ground truth and the image. Our proposed WateRF can achieve a good balance between the rendering quality and bit accuracy. We show the results on 16 bits.

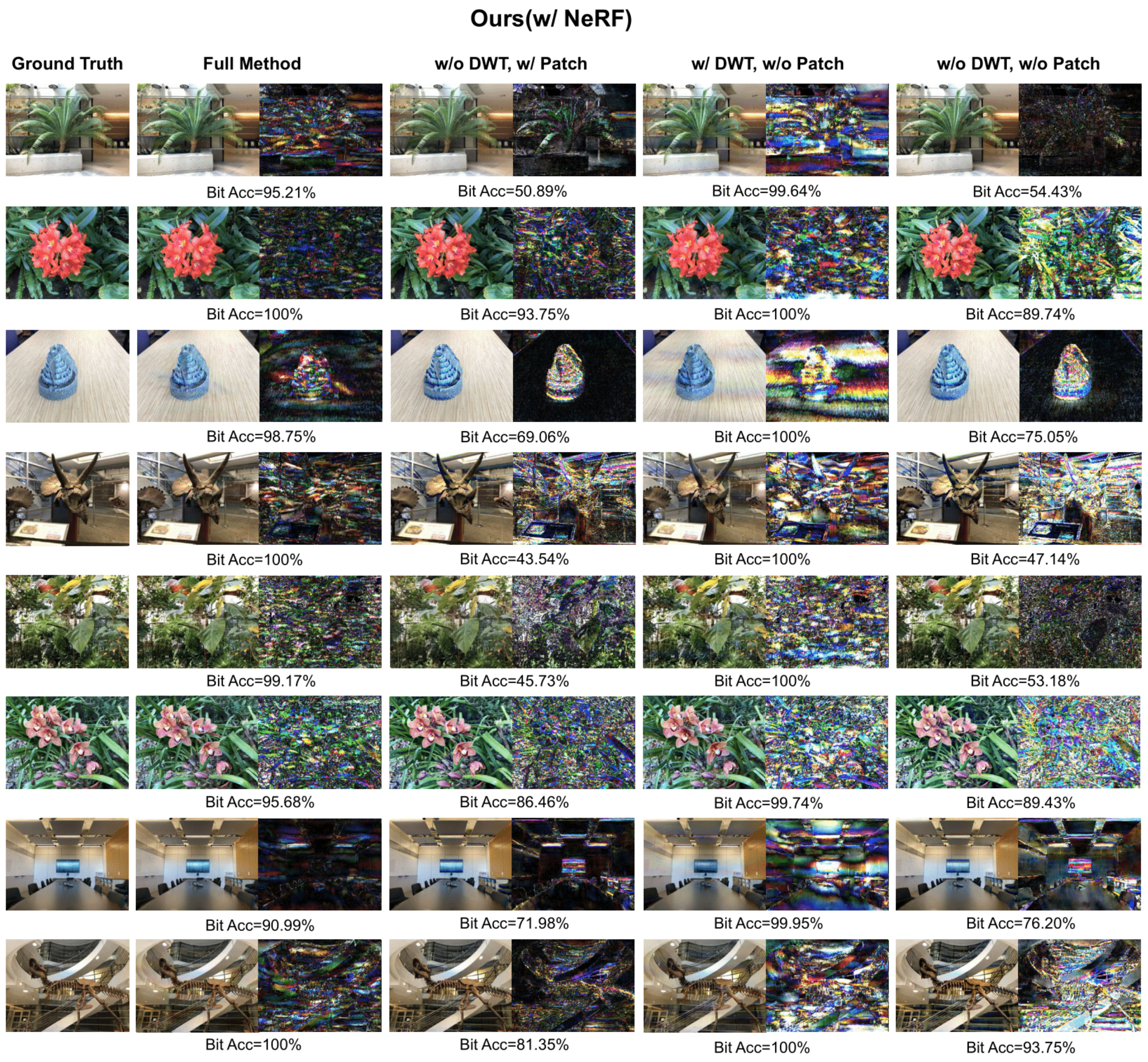

Figure 6. Reconstruction qualities of various rendering outputs using our method (with Vanilla NeRF[5]) for an ablation study on LLFF dataset[4]. We show the differences (×10) for the full method, our method without patch loss, and our method without frequency domain. The closer it is to white, the bigger the difference between ground truth and the image. Our proposed WateRF can achieve a good balance between the rendering quality and bit accuracy. We show the results on 16 bits.

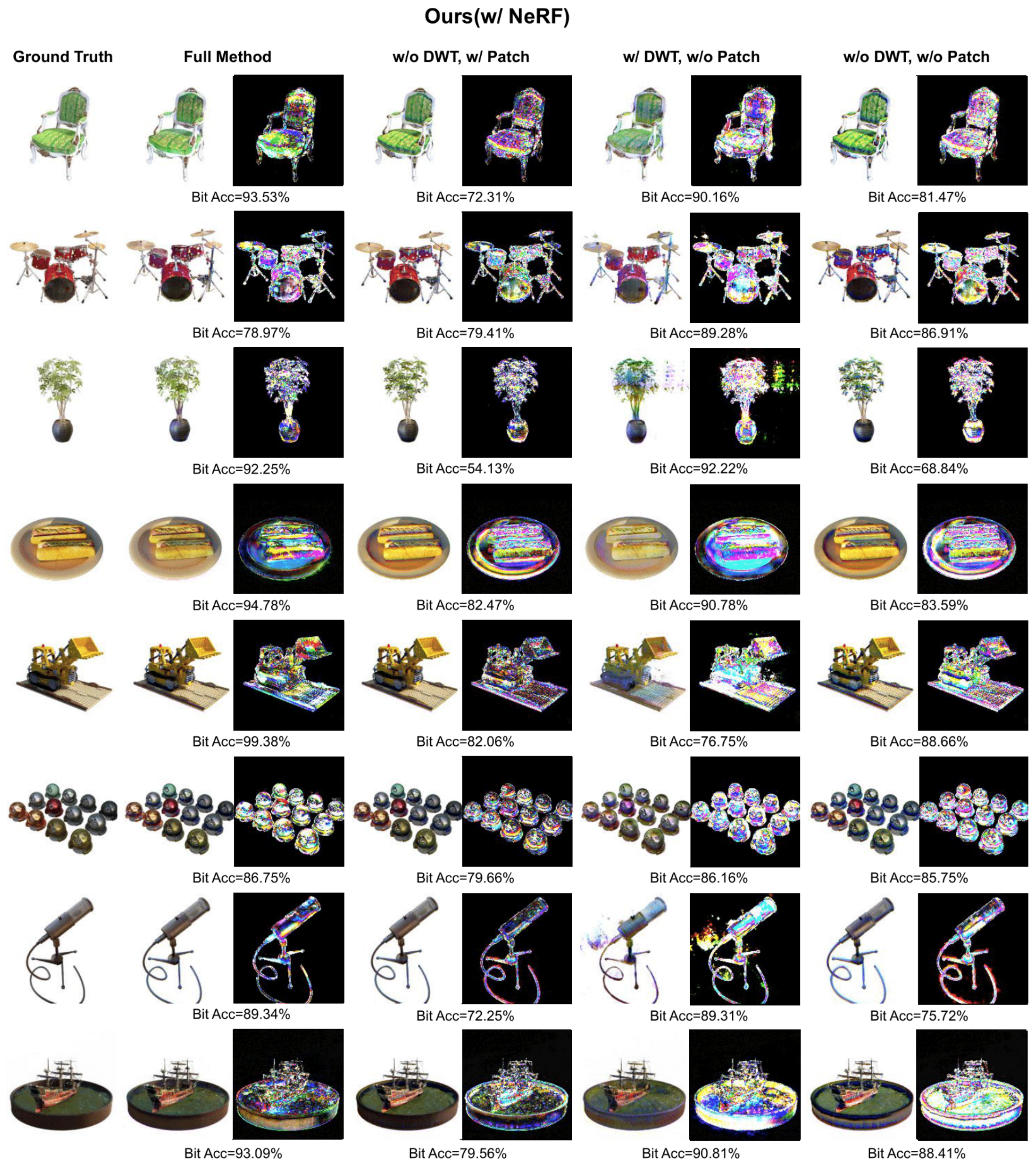

Figure 7. Reconstruction qualities of various rendering outputs using our method (with Vanilla NeRF[5]) for an ablation study on blender dataset[5]. We show the differences (×10) for the full method, our method without patch loss, and our method without frequency domain. The closer it is to white, the bigger the difference between the ground truth and the image. Our proposed WateRF can achieve a good balance between the rendering quality and bit accuracy. We show the results on 16 bits.

References

[1] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, pages 333–350. Springer, 2022. 2, 3, 4

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014. 1

[3] Ziyuan Luo, Qing Guo, Ka Chun Cheung, Simon See, and Renjie Wan. Copyrnerf: Protecting the copyright of neural radiance fields. In *ICCV*, pages 22401–22411, 2023. 2, 3

[4] 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)*, 38(4):1–14, 2019. 3, 5

[5] 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*, 65(1):99–106, 2021. 2, 3, 4, 5, 6

[6] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *ICLR*, 2020. 1

[7] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *ECCV*, pages 657–672, 2018. 1, 3