

S^2 NeRF: Privacy-preserving Training Framework for NeRF

Bokang Zhang*

The Chinese University of Hong Kong, Shenzhen
bokangzhang@link.cuhk.edu.cn

Jinglan Yang

The Chinese University of Hong Kong, Shenzhen
jinglanyang@link.cuhk.edu.cn

Yanglin Zhang*

The Chinese University of Hong Kong, Shenzhen
yanglinzhang@link.cuhk.edu.cn

Lingying Huang

Nanyang Technological University
lingying.huang@ntu.edu.sg

Zhikun Zhang†

Zhejiang University
zhikun@zju.edu.cn

Junfeng Wu†

The Chinese University of Hong Kong, Shenzhen
junfengwu@cuhk.edu.cn

Abstract

Neural Radiance Fields (NeRF) have revolutionized 3D computer vision and graphics, facilitating novel view synthesis and influencing sectors like extended reality and e-commerce. However, NeRF's dependence on extensive data collection, including sensitive scene image data, introduces significant privacy risks when users upload this data for model training. To address this concern, we first propose a strawman solution: SplitNeRF, a training framework that incorporates *split learning* (SL) techniques to enable privacy-preserving collaborative model training between clients and servers without sharing local data. Despite its benefits, we identify vulnerabilities in SplitNeRF by developing two attack methods, Surrogate Model Attack and Scene-aided Surrogate Model Attack, which exploit the shared gradient data and few leaked scene images to reconstruct private scene information. To counter these threats, we introduce S^2 NeRF, secure SplitNeRF that integrates effective defense mechanisms. By introducing decaying noise related to the gradient norm into the shared gradient information, S^2 NeRF preserves privacy while maintaining a high utility of the NeRF model. Our extensive evaluations across multiple datasets demonstrate the effectiveness of S^2 NeRF against privacy breaches, confirming its viability for secure NeRF training in sensitive applications.¹

CCS Concepts

- Security and privacy; • Computing methodologies → Machine learning;

Keywords

Privacy-preserving Machine Learning; Split Learning; Neural Radiance Fields (NeRFs)

*Bokang and Yanglin are co-first authors.

†Corresponding authors.

¹The implementation can be found at <https://github.com/lucky9-cyou/S2-NeRF>.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0636-3/24/10

<https://doi.org/10.1145/3658644.3690185>

ACM Reference Format:

Bokang Zhang, Yanglin Zhang, Zhikun Zhang, Jinglan Yang, Lingying Huang, and Junfeng Wu. 2024. S^2 NeRF: Privacy-preserving Training Framework for NeRF. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA*. ACM, New York, NY, USA, 27 pages. <https://doi.org/10.1145/3658644.3690185>

1 Introduction

Neural radiance fields (NeRF) [20] represent a transformative development in the domain of novel view synthesis, catalyzing significant advancements in 3D computer vision and graphics. The impact of NeRF is wide-ranging, influencing various sectors such as extended reality, 3D generative AI, e-commerce, and robotics. NeRF's training procedure necessitates the collection of camera parameters, including position and orientation, alongside images captured from varied camera positions. This technology is increasingly incorporated into consumer devices, such as extended reality glasses, to model users' environments accurately. Presently, to facilitate NeRF model training, users are often required to upload images of their surroundings along with camera parameters to a cloud server managed by the service provider [16]. This process, while standard, allows the company unrestricted access to the uploaded images. Moreover, the trained NeRF models enable these companies to reconstruct user environments in extensive detail, posing significant privacy risks. Such exposure of personal data underscores a critical need for enhanced privacy safeguards in NeRF applications.

The privacy concerns associated with NeRF can be effectively addressed using *split learning* (SL), a methodology that suits the privacy requirements essential for training neural networks collaboratively between two parties [12, 37, 39]. This approach is particularly pivotal in *vertical federated learning* (VFL), where the data is vertically partitioned into two parties [7]. Within VFL, there are primarily two architectures: one without model splitting [9] and another that incorporates model splitting [39]. SL is categorized under VFL which includes model splitting. Within the framework of split learning, two distinct parties, the “user party” possessing the data inputs and the “label party” holding the labels, can cooperatively train a model comprising two sub-models—“user model” and “label model”. This setup ensures that neither party has to reveal their respective data inputs or labels to the other. The only information exchanged between the two parties involves the cut layer embeddings, used for forward computation, and the gradients, necessary

for the backward updates, thereby safeguarding the privacy of both the data and labels [39].

Recent research has uncovered several privacy attacks in SL settings, demonstrating that sensitive labels can be accurately inferred by adversarial parties across classification and regression tasks [5, 9, 17, 27, 46]. These studies highlight a significant risk in SL environments: the potential for an adversarial user party to reconstruct sensitive label information using access to partial models and shared gradient data, underscoring the need for enhanced security measures in SL systems.

To address such privacy concerns, privacy protection is primarily achieved through two categories of methods. The first involves the use of sophisticated cryptographic protocols, such as *secure multi-party computation* [22, 40] and *two-party computation* [23, 28, 45]. These approaches provide strong security guarantees by ensuring that the computation on private inputs is performed without revealing them to any party. The second category employs perturbation-based methods that introduce randomness into the data-sharing process to protect privacy. This includes obfuscating the shared gradient information among parties or directly perturbing the information that needs protection [1, 6, 11, 32, 34, 44, 48].

Our Contributions. In this paper, we introduce SplitNeRF, a NeRF training framework adapted to the SL methodology. In SplitNeRF, the client and server collaboratively train the NeRF model without exchanging local client data, addressing the inherent privacy concerns of traditional NeRF training approaches. To empirically assess the privacy risks of the SplitNeRF framework, we propose and implement two NeRF attack methods: Surrogate Model Attack and Scene-aided Surrogate Model Attack. We implement the Surrogate Model Attack method by integrating the model reconstruction attack [9] with the gradient matching technique [46]. We employ constructed dummy image data to enhance the surrogate model's ability to approximate the client's model. To explore the effectiveness of Surrogate Model Attack, we conduct extensive experiments utilizing various learning rate strategies across three popular datasets. These experiments demonstrate its capability to accurately restore the geometric outlines of the original scenes, albeit in black-and-white format. Moreover, the Scene-aided Surrogate Model Attack addresses scenarios where an attacker has access to a limited number of actual scene images, a common occurrence given the proliferation of social networks. Scene-aided Surrogate Model Attack leverages leaked image data to determine the corresponding camera poses, using this information to fine-tune the attack model trained by Surrogate Model Attack. This method effectively enhances the reconstructed black-and-white images to restore their color, achieving a Peak Signal Noise Ratio(PSNR) of 27.62 and a Learned Perceptual Image Patch Similarity(LPIPS) [52] of 0.19, demonstrating significant restoration quality even with just one leaked image.

To remedy these vulnerabilities, we develop S^2 NeRF, secure SplitNeRF that incorporates robust defense mechanisms. S^2 NeRF distorts the gradient information transmitted from the client to the server. By perturbing the gradients, it safeguards the client's scene privacy by disrupting the process of Surrogate Model Attack and Scene-aided Surrogate Model Attack. To ensure S^2 NeRF's applicability across various scenarios, S^2 NeRF adjusts the noise scale

based on the norm of the transmitted gradient and employs a decay strategy for the noise. The decay strategy is designed to balance privacy protection with the utility of the NeRF model during training. We have conducted comprehensive experiments to assess the effectiveness of S^2 NeRF against multiple attack strategies across three datasets. These experiments demonstrate that with appropriate configuration, S^2 NeRF can effectively resist attacks while maintaining high model utility across different dataset environments. This balance showcases the potential of S^2 NeRF to serve as a viable solution for secure NeRF training in diverse applications.

- **(C1) SplitNeRF.** To our knowledge, SplitNeRF is the first approach that tackles privacy issues in NeRF training by employing SL techniques. Within the SplitNeRF framework, the client and server collaboratively train the NeRF model without necessitating the transfer of local private data to the server.
- **(C2) Surrogate Model Attack and Scene-aided Surrogate Model Attack.** We develop two attack strategies, Surrogate Model Attack and Scene-aided Surrogate Model Attack against the SplitNeRF framework. Comprehensive experiments on three indoor NeRF datasets demonstrate that these attacks can effectively exploit vulnerabilities in the vanilla SplitNeRF setup, revealing significant privacy risks.
- **(C3) S^2 NeRF.** To remedy the vulnerabilities uncovered, we propose the secure SplitNeRF framework, which we call S^2 NeRF. Our extensive experiments have shown that S^2 NeRF effectively prevents attackers from attempting to recreate any part of the scene while maintaining the high utility of the NeRF model.

2 Preliminaries

2.1 NeRF in 3D Reconstruction

NeRF [20] uses *multilayer perception* (MLP) Θ_σ and Θ_c to map the 3D location $\mathbf{x} \in \mathbb{R}^3$ and viewing direction $\mathbf{d} \in \mathbb{R}^2$ to a RGB color $\mathbf{c} \in \mathbb{R}^3$ and a density value $\sigma \in \mathbb{R}^+$:

$$\begin{aligned} [\sigma, \mathbf{z}] &= \Theta_\sigma(\gamma_x(\mathbf{x})), \\ \mathbf{c} &= \Theta_c(\mathbf{z}, \gamma_d(\mathbf{d})), \end{aligned}$$

where γ_x and γ_d are high-frequency positional encoding for location and viewing direction, respectively, which are designed to make MLP better suited for modeling functions in low dimensions. It's able to overcome the spectral bias inherent in the NeRF model [36]. The intermediate variable \mathbf{z} is a feature output by the first MLP Θ_σ . To better represent the space and improve training efficiency, part of the MLPs Θ_σ can be replaced with a small neural network [24], which is augmented by a multi-resolution hash table of trainable feature vectors optimized through stochastic gradient descent.

For rendering a 2D image from the radiance fields Θ_σ and Θ_c , a numerical quadrature is used to approximate the volumetric projection integral. Formally, N_p points are sampled along a camera ray r with color and geometry values $\{(\mathbf{c}_r^i, \sigma_r^i)\}_{i=1}^{N_p}$. The RGB color value $\hat{\mathbf{C}}(r)$ is obtained using alpha composition [30]

$$\hat{\mathbf{C}}(r) = \sum_{i=1}^{N_p} T_r^i \left(1 - \exp(-\sigma_r^i \delta_r^i) \right) \mathbf{c}_r^i,$$

Table 1: Summary of the notations.

Notations	Descriptions
\mathbf{x}, \mathbf{d}	3D location and viewing direction
\mathbf{c}, σ	RGB Color and density of a 3D point
Θ_σ, Θ_c	MLPs that output σ and \mathbf{c}
$\gamma_{\mathbf{x}}, \gamma_{\mathbf{d}}$	High-frequency positional encoding from \mathbf{x}, \mathbf{d}
r	A camera ray from the camera position to a pixel
N_p	Number of sampled points in a ray
$\{(\mathbf{x}_r^i, \mathbf{d}_r^i)\}_{i=1}^{N_p}$	The set of 3D points sampled along a camera ray r
$\{(\mathbf{c}_r^i, \sigma_r^i)\}_{i=1}^{N_p}$	The set of 3D points' RGB color and density
N_r	Number of sampled pixels
$\hat{\mathbf{C}}(r), \mathbf{C}(r)$	RGB predictions and ground truth of ray r

where $T_r^i = \prod_{j=1}^{i-1} \left(\exp(-\sigma_r^j \delta_r^j) \right)$, and δ_r^i is the distance between adjacent sample points. The MLPs Θ_σ and Θ_c are optimized by minimizing the reconstruction loss between observations \mathbf{C} and predictions $\hat{\mathbf{C}}$ as

$$L_{recon} = \frac{1}{N_r} \sum_{m=1}^{N_r} \|\hat{\mathbf{C}}(r_m) - \mathbf{C}(r_m)\|_2^2, \quad (1)$$

where N_r is the number of sampled pixels. Given Θ_σ and Θ_c , novel views are synthesized by invoking volume rendering for each ray.

2.2 Two-party Split Learning

Split learning [12, 39] has been proposed to enable two parties, a user party and a label party, to collaboratively train a composite model with the vertically partitioned data (usually in federated learning [47]). In particular, the user party holds the feature of the training data while the label party holds the label. By the data location, the composite model is split into the user model and label model held by the user and label party respectively. More formally, for user party, denote the user's input data set $X = \{x_i, i \in [1, \dots, |D|]\}$, and the user model \mathcal{M}_{user} ; for the label party, denote the label party's label $Y = \{y_i, i \in [1, \dots, |D|]\}$ and label model \mathcal{M}_{label} . $D = \{(x_i, y_i)\}$ is the whole training dataset to use for training the composite model $\mathcal{M}_{user} \circ \mathcal{M}_{label}$. For example, in the practical advertisement conversion prediction, x_i can be a feature vector about a user and y_i indicates whether the user clicks on the advertisement. The final layer of the user model \mathcal{M}_{user} , where the composite model is split, is called the cut layer.

Split learning follows the conventional training process, consisting of two phases: 1) forward propagation and 2) backpropagation.

Forward Propagation. The user party computes the embedding of the cut layer $\mathcal{E}_i = \mathcal{M}_{user}(x_i)$. This embedding is used as input to the label model held in the label party to compute the prediction score of the whole composite model:

$$y_i^{predict} = \mathcal{M}_{label}(\mathcal{M}_{user}(x_i)) = \mathcal{M}_{label}(\mathcal{E}_i).$$

Then we evaluate the loss to capture the disagreement between the predictions and the ground truth: $L(y_i^{predict}, y_i)$.

Backpropagation. The label party first updates its label model's parameters by computing the gradient of loss L concerning the label model itself \mathcal{M}_{label} . To further help the user party to compute the updates for user model \mathcal{M}_{user} , the label party needs to compute

the gradient for the embedding \mathcal{E}_i at the cut layer by a chain rule, $\nabla_{\mathcal{E}_i} L$ (denoted as g for short), where

$$g \triangleq \nabla_{\mathcal{E}_i} L = \frac{\partial L(y_i^{predict}, y_i)}{\partial y_i^{predict}} \cdot \frac{\partial y_i^{predict}}{\partial \mathcal{E}_i}.$$

Then the label party sends this gradient to the user party. Finally, the user party continues to compute the gradient for the user model w.r.t. \mathcal{M}_{user} 's parameters:

$$\nabla_{\mathcal{M}_{user}} L = \nabla_{\mathcal{E}_i} L \cdot \frac{\partial \mathcal{E}_i}{\partial \mathcal{M}_{user}}.$$

As for the inference phase, the user party first computes the cut layer embedding $\mathcal{E}_i = \mathcal{M}_{user}(x_i)$ and sends it to the label party, which computes the final prediction.

2.3 Notations

Frequently used notations in SL and NeRF are summarised in Table 1.

3 The Strawman Solution: SplitNeRF

Cloud computing facilitates NeRF training between an edge device and a remote server. The edge device, which is resource-constrained and lacks support for powerful GPUs and extensive memory capacities, acts as the client of the training service provided by the server. In standard training configurations, the client needs to upload image data to the server, posing potential privacy risks. To address the privacy leakage concerns, we first propose the strawman solution: SplitNeRF, a split learning framework tailored for NeRF training in 3D reconstruction. Compared to traditional centralized NeRF training, SplitNeRF keeps the labels locally and sends the features to the server. In the context of NeRF training, the features (camera poses) are not considered private information.

The SplitNeRF framework is illustrated in Figure 1. It assigns a portion of the NeRF model to the cloud server, therefore alleviating the computational burden on the client. During the SplitNeRF training process, the server does not directly gain access to the client's private image dataset but solely contributes computational resources. All sensitive image data is strictly retained on the client side. Within this framework, the NeRF model is partitioned into two components: the server model \mathcal{M}_{server} and the client model \mathcal{M}_{client} , by which the MLP layer responsible for generating density information must be retained on the client side. This setup enables the client to retain privacy-sensitive data locally, thus bolstering data privacy.

Following the standard NeRF training process and the virtue of split learning, SplitNeRF consists of two phases: 1) forward propagation and 2) backpropagation.

Forward Propagation. Initially, N_p points $\{(\mathbf{x}_r^i, \mathbf{d}_r^i)\}_{i=1}^{N_p}$ are sampled along a camera ray r on the client side. Subsequently, these points are transmitted to the server. The server then calculates the cut layer embeddings for these N_p points, denoted as $\{\mathcal{E}_r^i\}_{i=1}^{N_p}$. These embeddings are sent back to the client. Upon receiving the embeddings, the client uses them as inputs to its model to compute the final output labels, which include predicted density and color, represented as $\{(\mathbf{c}_r^i, \sigma_r^i)\}_{i=1}^{N_p}$. The client then calculates the

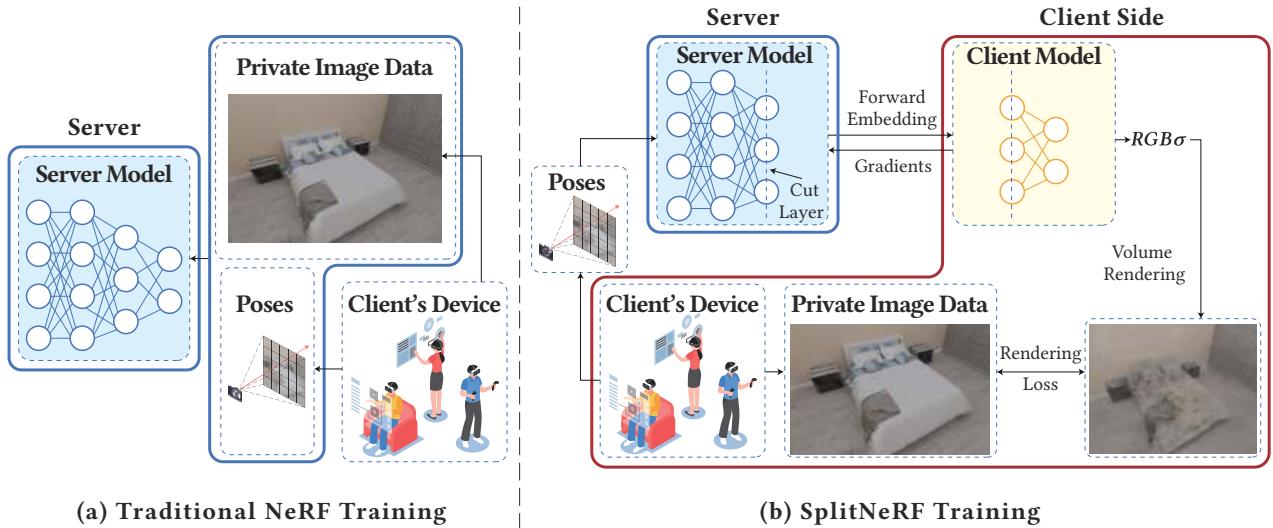


Figure 1: Overview of SplitNeRF framework. Traditional NeRF training requires the uploading of all training data, including scene images and corresponding camera poses, to a central server, presenting a substantial risk to privacy. We propose a split learning-based NeRF training framework, named SplitNeRF. The entire NeRF training model is divided into two parts, the server model and the client model. Specifically, clients are required only to send camera poses to the server, while keeping private image data local. During training, the client transmits a series of poses to the server, which then calculates and sends back the embeddings of these poses. The client proceeds to compute the colors and densities associated with these poses, followed by rendering and loss function calculations. The client then sends the gradients back to the server, thus completing the backpropagation process.

predicted RGB color $\hat{C}(r)$. The reconstruction loss between the predictions and the ground truth is then calculated using the local image data held by the client.

privacy? Next, we evaluate this privacy concern by devising attack methods against SplitNeRF to assess its vulnerabilities.

Backpropagation. After calculating the reconstruction loss, the client proceeds to update its model \mathcal{M}_{client} . To facilitate the server in completing updates to its model parameters \mathcal{M}_{server} , the client computes the shared gradients $g_r^i = \frac{\partial Loss}{\partial \mathcal{E}_r^i}$ for the cut layer embeddings using a chain rule. With these shared gradients, the server is then able to complete the backpropagation process for its network \mathcal{M}_{server} .

In the SplitNeRF inference, the client is tasked with rendering an image based on the 3D position and orientation of a camera. If the client's storage capacity permits, the server may transfer the entire set of server model parameters to the client, excluding itself from the inference process. Alternatively, if the client lacks sufficient local computing resources, the inference can be conducted interactively. In this scenario, the client begins by sampling the points required for rendering, based on the camera's position and orientation, and sends these points to the server. Then, similar to the forward propagation process, the client obtains the predicted pixels and proceeds to render the image.

Privacy Analysis. SplitNeRF enhances privacy compared to traditional NeRF training methods that require uploading local datasets to a cloud server. By keeping the image dataset local, SplitNeRF minimizes data exposure. Nevertheless, the server retains access to portions of the NeRF network parameters and receives gradient information during training. This raises an important question: *could the server exploit this information to compromise local data*

4 Privacy Risk Assessment of SplitNeRF

In this section, we comprehensively assess the privacy risks of SplitNeRF by designing two attacks to restore the client’s NeRF model to infer the client’s private information. First, we introduce Surrogate Model Attack against SplitNeRF, and then we introduce Scene-aided Surrogate Model Attack provided the attacker is additionally capable of getting access to a few leaked scene image data.

4.1 Surrogate Model Attack

4.1.1 Threat Model. We define the threat model for our attack in aspects of security research, including the attack setting and goal, and the attacker's capability and knowledge.

Attack Setting & Goal. Similar to numerous other studies in SL [9, 17], we employ the honest-but-curious model assumption. In this model, the attacker (server) adheres to the standard training protocol while attempting to deduce the private information and neural networks on the client side. In NeRF applications, the user’s network parameters are often as sensitive as the user’s stored image data, unlike in traditional deep-learning tasks, as the NeRF neural network serves as a comprehensive neural representation of the client’s scene. If an attacker gains access to the client model, it could render views of the client’s environment from any 3D poses by integrating the client model with the server model. Consequently,

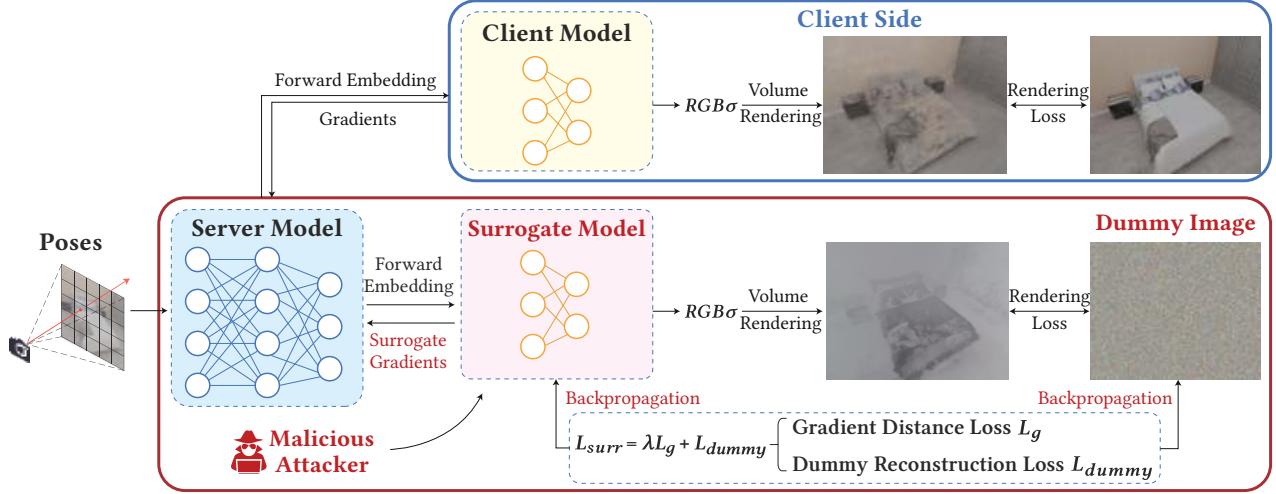


Figure 2: Overview of Surrogate Model Attack against SplitNeRF. The attacker (curious server) attempts to mimic the representation capabilities of the client model to generate a 3D description of the client’s private scenario. To achieve this, the attacker sets up a surrogate model along with dummy image data. The attacker utilizes backpropagation to optimize both the surrogate model and dummy image with the surrogate model loss L_{surr} , which includes gradient distance loss L_g and dummy reconstruction loss L_{dummy} .

the attacker’s main goal is to create a surrogate model that closely matches the client model.

Attackers’ Capabilities & Knowledge. Surrogate Model Attack operates under a black-box setting, where the attacker is unaware of both the client model parameters and the client model structure. In this scenario, the attacker has the flexibility to tailor its surrogate model, for example, by increasing its depth or width, to facilitate the attack. Unlike other SL scenarios [9, 46], obtaining even a small portion of complete training data is challenging in NeRF training due to the difficulty in accurately matching images to their corresponding poses. This is explored further in Section 4.2. Consequently, the attacker must construct the attack based solely on the server model parameters and the gradient information exchanged during training. The attacker has the discretion to determine the training approach for the surrogate model based on the available information.

4.1.2 Attack Methodology. Our proposed Surrogate Model Attack aims to approximate the representation capabilities of the client model through a learning-based method. Initially, we construct a surrogate model designed to approximate the client model and dummy image data to approximate the client’s image data. We then establish a loss function incorporating available information, such as gradients, and integrate additional machine learning objectives, such as model performance, as forms of regularization. Subsequently, we iteratively backpropagate the surrogate model and the dummy image data. This approach is detailed in Figure 2, illustrating our attack framework.

The surrogate model and the corresponding dummy image data for a given ray r are denoted as M_{surr} and $C(r)_{dummy}$, respectively. We first initialize M_{surr} and $C(r)_{dummy}$ and then calculate the predictions made by the surrogate model and the gradients at the cut layer (denoted as g_{surr}) following the SplitNeRF process. Next,

Algorithm 1: Surrogate Model Attack against SplitNeRF

```

1 Input: Training data(rays)  $\{r_m, m \in [1, \dots, |N_r|\}\}$ , Server
   model  $M_{server}$ , Number of training epochs  $T$ , surrogate
   model learning rate  $\eta_t$ 
2 Output: Inferred surrogate model  $M_{surr}$ 
3 Procedures:
4   /* Initialize dummy image data. */
5    $C_{dummy}^{(0)} \leftarrow [\mathcal{N}(0, 1)]^{|N_r|}$ 
6   Initialize the surrogate model  $M_{surr}^{(0)}$ .
7   for  $t = 0$  to  $T$  do
8     Compute  $L_g$  and  $L_{dummy}$  in (2) and (3)
9      $L_{surr} \leftarrow \lambda L_g + L_{dummy}$ ;
10    /* Optimize the surrogate model and dummy image data; */
11     $M_{surr}^{(t+1)} \leftarrow \text{AdamUpdate}(M_{surr}^{(t)})$ ;
12     $C_{dummy}^{(t+1)} \leftarrow \text{AdamUpdate}(C_{dummy}^{(t)})$ ;
13  end for
14 Return:  $M_{surr}^{(T+1)}$  as surrogate model

```

we introduce the loss function for optimizing the surrogate model and dummy image data.

Gradient Distance Loss. To reconstruct the private client’s network, we primarily utilize the shared gradient used in backpropagation. Specifically, the gradient g_{surr} produced by the surrogate model should closely approximate the original gradient g generated by the client model. To measure the closeness between these two gradients, we introduce the gradient distance loss, which quantifies the discrepancy between g and g_{surr} :

$$L_g = \mathcal{D}(g, g_{surr}), \quad (2)$$

where $\mathcal{D}(\cdot)$ denotes a distance function, e.g., the ℓ_2 norm function.

Dummy Reconstruction Loss. Considering that our learning objective involves high-dimensional spaces [46], it could be very likely to get different surrogate model parameters even if the gradient distance loss is close to 0. To constrain our learning space and guide the surrogate model to behave like the client model, we incorporate a dummy reconstruction loss as regularization. By incorporating this loss term, the surrogate model is compelled to function like the original NeRF model, particularly in scene reconstruction performance. Accordingly, we define the dummy reconstruction loss to ensure that the predictions of the ray r by the surrogate model M_{surr} closely match the dummy image data $C(r)_{dummy}$ once the surrogate model has converged.

Specifically, for a given ray r , the surrogate model M_{surr} first predicts the density and colors of the 3D sampled points along r and then computes the RGB color $\hat{C}(r)_{surr}$ at the pixel r . We then define the reconstruction loss as follows:

$$L_{dummy} = \frac{1}{N_r} \sum_{m=1}^{N_r} \|\hat{C}(r_m)_{surr} - C(r_m)_{dummy}\|_2^2, \quad (3)$$

where N_r represents the number of sampled pixels.

Additionally, we incorporate weighting parameters λ to balance the functionality of the components of the learning loss. In a nutshell, the learning loss for the surrogate model is given by

$$L_{surr} = \lambda L_g + L_{dummy}. \quad (4)$$

To minimize the above loss function over the surrogate model, we employ a gradient optimization algorithm, such as Adam [15], which facilitates the iterative updates to both M_{surr} and $C(r)_{dummy}$. The steps of updating M_{surr} and $C(r)_{dummy}$ are detailed in Algorithm 1.

Once the attacker has successfully trained a surrogate model as detailed in Algorithm 1, it can integrate it with the server model to assemble a complete NeRF model. This consolidated model enables the attacker to render the client’s scene from any camera pose.

4.2 Scene-aided Surrogate Model Attack

In this section, we explore a scenario where the attacker gains access to a limited amount of scene image data. Research has shown that fine-tuning a surrogate model with a small fraction of leaked data can significantly boost its performance [9, 46]. With the prevalent use of social networks, acquiring a few scene images has become progressively more accessible for potential attackers. However, accurately inferring the camera poses linked to these images poses a challenge. This hampers the attacker’s ability to effectively utilize the leaked images for developing the surrogate model, as precise pose data is essential for accurate NeRF training. Next, we clarify the threat model of Scene-aided Surrogate Model Attack and introduce the specific attack methodology.

4.2.1 Threat Model. The difference between the threat models of Scene-aided Surrogate Model Attack and Surrogate Model Attack is that Scene-aided Surrogate Model Attack has access to auxiliary information, which is a limited amount of scene image data. However, the corresponding camera poses for these images are not available.

4.2.2 Attack Methodology. Upon executing the Surrogate Model Attack as elaborated in Section 4.1, the attacker can utilize the surrogate model to generate and render 2D scene images captured from different camera poses. If the Surrogate Model Attack is effective, the images synthesized by the surrogate model can guide the attacker in deducing the actual poses of the scene images by comparing the rendered images with the real ones. Successful alignments between the images and their poses empower the attacker to refine the surrogate model. Therefore, the key lies in identifying the pose of the leaked images. We introduce our pose estimation method to address this challenge.

The attacker can perform a grid search method of camera poses to pinpoint the one that most closely matches the leaked image. This process involves dissecting the pose into two components: the 3D position and the camera viewing direction. Below, we delineate the search spaces for both components:

Search Space of 3D locations. NeRF typically normalizes three coordinate values within a 3D location to range between $[-1, 1]$ [20]. Consequently, the search space for 3D locations spans $[-1, 1]^3$. If the attacker’s search granularity is x , it traverses all three coordinates from $[-1, -1 + x, \dots, 1]$. This results in the attacker evaluating $(\lceil \frac{2}{x} \rceil + 1)^3$ potential 3D locations to find the one closest to the leaked image.

Search Space of Viewing Directions. For any 3D location, an omnidirectional view necessitates traversing 2π radians for the horizontal direction and π radians for the vertical direction. This traversal is typically guided by the Horizontal Field of View (HFOV) and Vertical Field of View (VFOV) of the camera used in the NeRF training. Denoting the camera’s HFOV as h and VFOV as v , the attacker uses polar and azimuth angles in the spherical coordinate system to define viewing directions. Hence, the search space for viewing directions spans $[0, 2\pi] \times [0, \pi]$. At each 3D location, the attacker incrementally adjusts the azimuth angles by $[0, h, \dots, 2\pi]$ and polar angles by $[0, v, \dots, \pi]$ to identify the viewing direction that aligns closest with the leaked image. The total number of viewing directions evaluated per location is $(\lceil \frac{2\pi}{h} \rceil + 1) * (\lceil \frac{\pi}{v} \rceil + 1)$.

For a leaked image, the attacker typically conducts at most $(\lceil \frac{2}{x} \rceil + 1)^3 * (\lceil \frac{2\pi}{h} \rceil + 1) * (\lceil \frac{\pi}{v} \rceil + 1)$ iterations to identify a camera pose. For instance, if NeRF training utilizes an Intel RealSense Depth Camera D435i² with an HFOV of 69° and a VFOV of 42° , and the granularity of the attack is set to 0.1, the total number of pose searches reaches approximately 388,962. After initially identifying a pose, the attacker can refine their search using finer granularity for more precise adjustments and potentially better alignment with the target pose. For example, employing a granularity of 0.01 for 3D locations and 1° for viewing directions enhances the precision of the pose adjustments. The attacker uses fidelity metrics, as detailed in Section 5.1, to gauge the closeness of the pose to the leaked image. This enables the automated identification of the pose that most precisely corresponds to the leaked image.

Discussion. The challenge of pose estimation is closely related to the inverting NeRF (iNeRF) problem, as discussed by Yen et al. [49]. This problem involves deducing the camera pose relative to a 3D

²<https://www.intelrealsense.com/depth-camera-d435i>

object or scene from a specific image. However, the images rendered by the attacker’s model in our scenario differ from standard color images, potentially reducing the effectiveness of the iNeRF’s optimization process, which is designed for full-color scenarios. We leave the integration of iNeRF for more efficient pose estimation of leaked images as a direction for future research.

5 Empirical Evaluation for Privacy Risks

In this section, we first assess the effectiveness of Surrogate Model Attack on the SplitNeRF framework, revealing significant privacy vulnerabilities within its architecture. Second, we execute the Scene-aided Surrogate Model Attack, leveraging a limited amount of leaked training image data to improve the fidelity of scene restoration. Lastly, we conduct ablation studies to investigate the impact of various attack strategies on the overall effectiveness of the attacks.

5.1 Experimental Setup

Datasets. We conduct our attack experiments using three well-established public NeRF datasets: the synthetic indoor dataset 3D-FRONT [10], Hypersim [33], and the real indoor dataset ScanNet [3]. 3D-FRONT provides an extensive collection of large-scale synthetic indoor scenes with intricate room layouts and textured furniture models. Hypersim is a realistic synthetic dataset tailored for indoor scene understanding, featuring a variety of rendered objects with 3D semantic annotations. ScanNet is a widely used real-world dataset primarily utilized for indoor 3D object detection, comprising over 1,500 scenes. Given the potential for indoor scenes to inadvertently disclose sensitive information, these datasets are deliberately chosen to enable a comprehensive assessment of privacy vulnerabilities within the SplitNeRF framework.

Fidelity Metrics. In place of visual contrasts between the attack-generated images and the original NeRF-rendered images, we introduce quantitative metrics to evaluate the attack’s efficacy. The attacker can generate both depth and color views from any camera pose using the surrogate model, making it feasible to evaluate the privacy breach based on the quality of these synthetic views. The Learned Perceptual Image Patch Similarity (LPIPS) [52] and Structural Similarity (SSIM) [41] metrics are used to assess the perceptual quality of images for privacy leakage evaluation [14]. Therefore, we utilize LPIPS and SSIM to define our fidelity metrics. Additionally, we conduct human evaluations to more realistically measure privacy breaches.

- **LPIPS-depth.** We utilize LPIPS-depth to quantify the perceptual similarity between the depth views rendered by the attacker and those from the original NeRF model.
- **LPIPS-gray.** Considering that the attacker’s synthetic views are frequently rendered solely in black and white, potentially affecting the synthetical scene’s structural assessment, we adjust the original NeRF’s color views to grayscale before proceeding with the comparison. This conversion enables a more targeted evaluation of scene details, decoupled from color influences. Consequently, we use LPIPS-gray to measure the similarity between their grayscale views.

- **SSIM-depth.** We utilize SSIM-depth to quantify the structural similarity between the depth views rendered by the attacker and those from the original NeRF model.
- **SSIM-gray.** Similarly to LPIPS-gray, we use SSIM-gray to measure the structural similarity between the grayscale views of the attacker and those from the original NeRF model.
- **Human-eval.** We present the attacker’s views (rendered in depth and color) alongside the real views to the annotators. The annotators are then asked to evaluate the severity of privacy disclosure, using a scale from 1 to 5, where 1 indicates no disclosure and 5 indicates complete disclosure. Each view pair is labeled by five independent annotators. We use Human-eval to report our Human evaluation results.

NeRF Model Implementation. We utilize Instant-NGP [24] through a third-party PyTorch implementation³ to implement our experiments. This version boosts the efficiency of feature map rendering compared to the original NeRF architecture. The model architecture comprises a hash encode network, a two-layer density MLP, and a three-layer color MLP. We employ the Adam optimizer [15] with an initial learning rate of 0.01, which decays exponentially to 0.1 of its original size by the final iteration. All experiments are executed on a server equipped with two NVIDIA RTX-4090 graphics cards and 256GB of memory.

Attack Setting. Figure 2 depicts the architecture of our attackers. We segment the NeRF network at the two-layer density MLP to specifically isolate output density information locally at the user side, thereby enhancing privacy. We mirror the client’s model architecture in the attacker’s surrogate model without loss of generality. This surrogate model consists of a one-layer density MLP and a three-layer color MLP. To address the issue of potentially disparate scales between the two loss terms, L_g and L_{dummy} , for the surrogate model, we dynamically adjust λ to maintain the loss ratio $\frac{\lambda L_g}{L_{dummy}}$ constant. We assess our attackers using three distinct learning rate decay schemes for training the surrogate model: $0.1^{\frac{t}{T}}$, $0.001^{\frac{t}{T}}$, $\frac{10}{t}$, where T is the number of total training epochs and t denotes the current epoch index.

5.2 Surrogate Model Attack Results

In this part, we evaluate the efficacy of Surrogate Model Attack with the loss ratio $\frac{\lambda L_g}{L_{dummy}} = 0.01$ and a learning rate decay scheme of $\frac{10}{t}$, referred to as the $\frac{10}{t}$ -attack. The experimental results for other attack configurations, involving different loss ratios and decay schemes, are elaborated in Section 5.4 as part of an ablation study aimed at delving deeper into the impact of different configurations on the attack’s performance.

Experimental Results. From Figure 3, it is evident that the $\frac{10}{t}$ -attack method enables partial reconstruction of the original images in terms of both depth and color views, although the color views are presented solely in grayscale. In conclusion, the $\frac{t}{10}$ -attack method can compromise the privacy of the scenes within the three datasets.

In the case of the 3D-FRONT and Hypersim datasets, the attack notably restores recognizable objects in various scenes, such as an

³A PyTorch CUDA extension implementation of Instant-NGP: <https://github.com/ashawkey/torch-ngp>

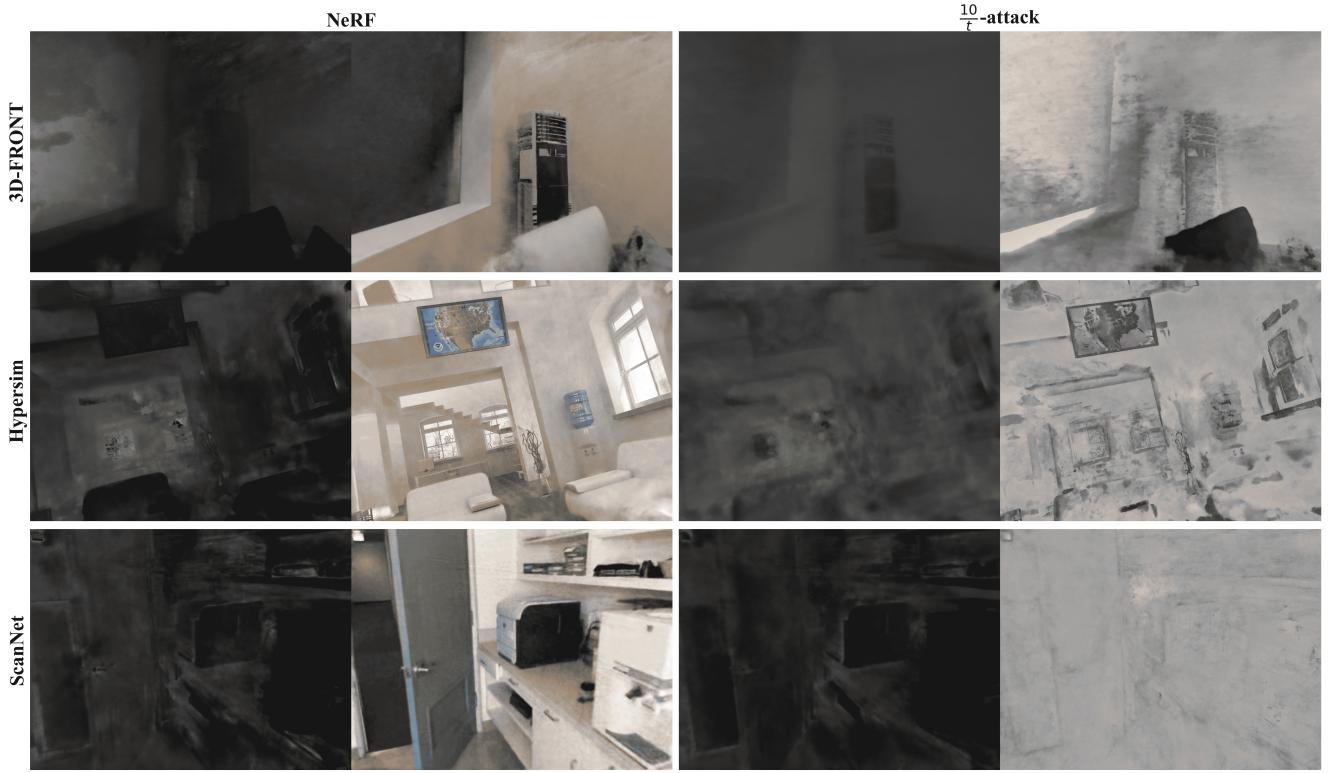


Figure 3: Surrogate Model Attack results on the three datasets, utilizing $\frac{10}{t}$ learning rate decay schemes, where t denotes the current learning epoch index. The attacker successfully restores partial outlines of the actual scenes across all datasets. Even though the views generated by the attacker are rendered in grayscale, it remains possible to distinguish indoor items and layout scenarios. The level of detail in the reconstructed scenes highlights a significant privacy violation, as it exposes sensitive information about the structure and contents of the scenes.

air conditioner and sofa in 3D-FRONT, and a water fountain and photo frame in Hypersim. This restoration underscores a significant breach of privacy. In the ScanNet dataset, while the color rendition produced by the attack lacks precision, the depth representation closely mirrors the original scene, enabling object identification. These outcomes indicate that the attacker’s model has effectively captured the spatial geometry of the scenes. Despite the synthetic scenes lacking true color fidelity, the attacker’s ability to reconstruct scene geometry signifies a substantial compromise in privacy.

5.3 Scene-aided Surrogate Model Attack Results

In this section, we evaluate the effectiveness of the Scene-aided Surrogate Model Attack against the SplitNeRF framework, specifically when the attacker has held just a single scene image of the trained scene.

Attack Setup. We execute the Scene-aided Surrogate Model Attack on a bedroom scene from the 3D-FRONT dataset. We randomly pick an image as the leaked image from the training data. Subsequently, we proceed with the pose estimation method as outlined in Section 4.2. This identified pose, alongside the corresponding training data, is then used to fine-tune the attacker’s surrogate model in Section 5.2. After Scene-aided Surrogate Model Attack, we render multiple poses from the attack model to visually compare with the original scene in Figure 4.

Results. As depicted in Figure 4, the Scene-aided Surrogate Model Attack effectively reconstructs the original scene with a notable Peak Signal to Noise Ratio(PSNR) of 27.62 and an LPIPS of 0.19. While the details in the scene generated by the attack may appear blurred, and color accuracy may deviate from the actual scene, the overall similarity to the real scene is remarkably close. This fidelity is apparent not only in the pose aligned with the leaked image but also in other poses, which closely mirror the original scene.

Comparison with Surrogate Model Attack. The grayscale views in Figure 3 resemble the depth views of the scene, which are derived from integrating the spatial density of points. Since the cut layer output embedding primarily relates to the density output, the attacker acquires the scene’s density information more readily.

With the incorporation of the scene image, the attacker gains supervision over color rendering, enabling more effective fine-tuning of the color network. Consequently, Scene-aided Surrogate Model Attack more accurately approximates the actual scene, significantly enhancing the fidelity of the reconstructed images.

5.4 Surrogate Model Attack Ablation Study

We now evaluate the impact of different attack methods concerning the attack performance against SplitNeRF.

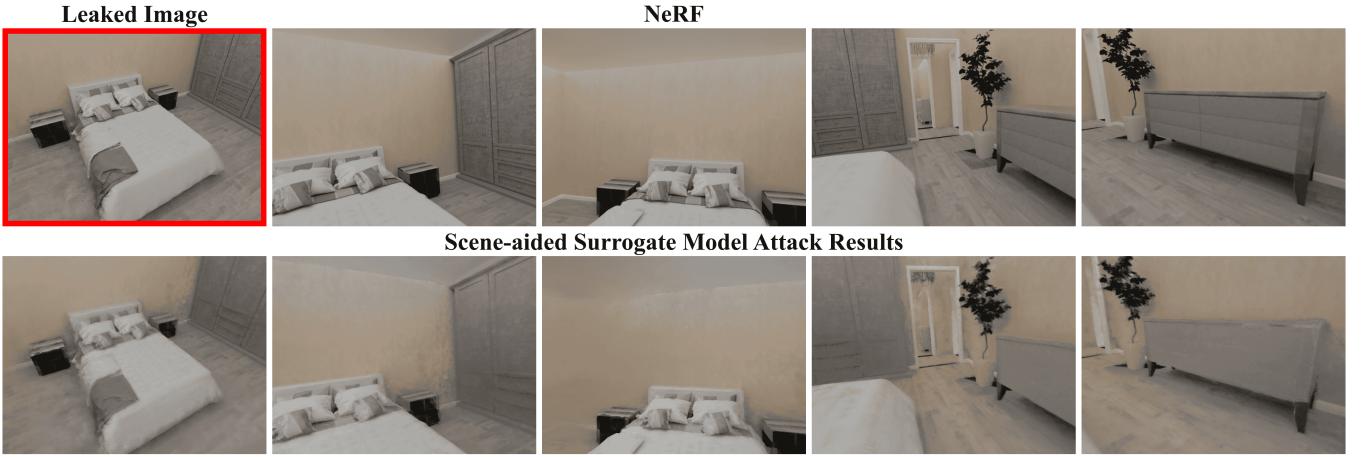


Figure 4: Scene-aided Surrogate Model Attack Results. The results show the attack’s ability to significantly restore high-quality, colored views from just one leaked picture, closely resembling the original scene. The fidelity of the attack extends beyond the specific pose of the leaked image, accurately replicating various other poses and effectively capturing the entire scene.

Learning Rate Decay Scheme in Surrogate Model Attack. The choice of a learning rate decay scheme plays a crucial role in NeRF training, and similarly, it is vital for training the surrogate model in Surrogate Model Attack. Initially, we adopt the same learning rate decay strategy used in the original NeRF model training: $0.1^{\frac{t}{T}}$. Additionally, we introduce two faster decaying schemes: $0.001^{\frac{t}{T}}$, $\frac{10}{t}^4$. We set the loss ratio to be 0.01.

Quantitative comparison results of these three attack strategies are presented in Table 2, with visual outcomes provided in Appendix A. The results suggest that the $\frac{10}{t}$ -attack generally outperforms the others. During the early stages of training, NeRF primarily focuses on learning geometric information, while later stages are devoted to enhancing color information and details. In the initial phases, attackers, who possess part of the geometry network and access gradients situated within this segment, can more effectively acquire geometric information. However, in the later stages of training, the gradients returned by the client mainly reflect updates to the color network. Since the supervision provided by the surrogate model over the color network updates is not enough, the effectiveness of the attack diminishes during this period. The $\frac{10}{t}$ -attack maintains effective attack performance because it decays more rapidly and seldom updates network parameters in the later stages of training.

Surrogate Model’s Loss Ratio. The effectiveness of the surrogate model in our attack framework is governed by the gradient loss L_g and the dummy reconstruction loss L_{dummy} . Consequently, the ratio of these losses, represented as $\frac{\lambda L_g}{L_{dummy}}$, dictates how heavily the surrogate model relies on gradient information during training.

To explore this dependency, we vary the loss ratio from 0.01 to 100 for the $\frac{10}{t}$ -attack. To thoroughly examine the impact of the two loss terms on attacks, we also conduct Surrogate Model Attack focusing solely on L_g (loss ratio ∞) or L_{dummy} (loss ratio 0). These attacks are relatively successful, with the human-eval metric generally higher than 4. Detailed visual outcomes are available

in Appendix A. Empirical results indicate that all chosen ratios can succeed in scene reconstruction in both depth and color views. Considering various attack metrics and visual results, the attack with a loss ratio of 0.01 performs better.

Different Structures of Surrogate Models. The default structure of the attacker’s surrogate model consists of a one-layer density MLP and a three-layer color MLP. Variations in surrogate model structure could potentially impact attack performance and defense effectiveness. Therefore, we conduct experiments with two additional surrogate model structures: a one-layer density MLP with a two-layer color MLP, and a one-layer density MLP with a four-layer color MLP. These structures represent different surrogate models and attacker capabilities. The results show that both structures can successfully attack SplitNeRF. More detailed results are presented in Appendix B.

Discussion about Fidelity Metrics. The LPIPS score, which measures perceptual similarity, ranges from 0 to 1. A score of 0 indicates that images are perceptually identical or extremely similar, while a score of 1 suggests significant perceptual differences.

The fidelity metric values exhibit significant variability across various scenarios. For instance, despite the visual recovery of many color view contours in the 3D-FRONT dataset in Figure 3, its LPIPS-gray value is high at 0.49 in Table 2. Conversely, the ScanNet dataset demonstrates a successful depth view attack, evidenced by a low LPIPS-depth value of 0.20. These examples highlight that, while the attack outcomes may visually seem effective, the fidelity metrics can vary significantly. Hence, to accurately assess attack effectiveness, comparisons must be made within the same scene. This can be further illustrated by examining the visuals and metrics detailed in Table 2, Table 3, and Appendix A.

It is also important to note that attack effectiveness does not uniformly reflect across all fidelity metrics. For example, in the ScanNet dataset in Figure 3, while the depth attack is successful with a low LPIPS-depth value of 0.20, the color views do not match this effectiveness, as indicated by higher values of LPIPS-gray. A score of around 0.7 in LPIPS-based metrics indicates an unsuccessful

⁴Since T tends to be large in NeRF training, often reaching values like 60,000, the scheme $\frac{10}{t}$ results in a faster decay rate in practice.

Table 2: Comparison of Surrogate Model Attack metrics for different learning rate decay strategies under the loss ratio 0.01. We utilize three distinct strategies: $\frac{10}{t}$, $0.1\frac{t}{T}$, $0.001\frac{t}{T}$, where t represents the current iteration and T denotes the total training iterations. We highlight the $\frac{10}{t}$ -attack with a red background and present the best results in bold. \uparrow (\downarrow) means a higher(lower) value is favored.

Dataset	Metric	Learning Rate Decay Schemes			Loss Ratio						
		$\frac{10}{t}$ -attack	$0.1\frac{t}{T}$ -attack	$0.001\frac{t}{T}$ -attack	0	0.01	0.1	1	10	∞	
3D-FRONT	LPIPS-depth \downarrow	0.44	0.47	0.49	0.22	0.44	0.44	0.41	0.44	0.46	0.46
	LPIPS-gray \downarrow	0.49	0.54	0.54	0.46	0.49	0.50	0.49	0.51	0.50	0.47
	SSIM-depth \uparrow	0.77	0.64	0.62	0.89	0.77	0.77	0.79	0.78	0.78	0.78
	SSIM-gray \uparrow	0.64	0.77	0.77	0.70	0.64	0.64	0.60	0.58	0.59	0.63
	Human-eval \uparrow	4.6	1.6	1.6	4.4	4.6	4.6	4.6	4.4	4.4	4.0
Hypersim	LPIPS-depth \downarrow	0.39	0.59	0.58	0.13	0.39	0.49	0.52	0.56	0.57	0.56
	LPIPS-gray \downarrow	0.41	0.65	0.65	0.47	0.41	0.43	0.46	0.47	0.47	0.48
	SSIM-depth \uparrow	0.82	0.22	0.26	0.96	0.82	0.78	0.78	0.78	0.78	0.78
	SSIM-gray \uparrow	0.72	0.80	0.80	0.78	0.72	0.69	0.66	0.65	0.66	0.61
	Human-eval \uparrow	5.0	3.2	3.2	4.6	5.0	4.4	4.8	4.8	4.8	5.0
ScanNet	LPIPS-depth \downarrow	0.20	0.22	0.22	0.22	0.20	0.44	0.50	0.56	0.63	0.65
	LPIPS-gray \downarrow	0.64	0.65	0.64	0.66	0.64	0.55	0.54	0.55	0.55	0.56
	SSIM-depth \uparrow	0.92	0.88	0.90	0.91	0.92	0.76	0.73	0.73	0.73	0.73
	SSIM-gray \uparrow	0.58	0.60	0.60	0.54	0.58	0.52	0.51	0.49	0.49	0.53
	Human-eval \uparrow	4.4	3.4	3.6	4.0	4.4	4.0	4.2	4.2	4.2	4.4

attack, emphasizing that success in one aspect does not guarantee overall effectiveness.

The same conclusion applies to SSIM-based metrics. The SSIM score, which measures structural similarity, ranges from 0 to 1. A score of 1 indicates that images are structurally identical or extremely similar, while a score of 0 suggests significant structural differences. In the ScanNet dataset (see Figure 3), while the depth attack is successful with a high SSIM-depth value of 0.92, the color views do not match this effectiveness, as indicated by low SSIM-gray values. Overall, the Human-eval metric is closer to the real privacy disclosure situation, although it is subject to human subjectivity. Specifically, for the results in Figure 3, the Human-eval metric remains high, consistently above 4.4 as shown in Table 2.

6 S²NeRF

Having identified significant privacy breaches within the Split-NeRF framework, our next step is to develop a version that ensures privacy. The privacy-preserving framework should satisfy the following objectives:

- **(G1) Defense Effectiveness.** The framework must effectively address the identified privacy vulnerabilities. Specifically, the framework needs to protect against attackers who utilize gradient information and potentially some leaked images.
- **(G2) Model Utility.** It is critical for the framework to maintain the utility of the NeRF model, ensuring it can truthfully and effectively represent the target scenes in 3D reconstruction.

NoisyLabelNeRF. Some prevalent defense methods against malicious attacks in SL involve protecting private label information

Table 3: Comparison of Surrogate Model Attack metrics for different surrogate model's loss ratio under $\frac{10}{t}$ -attack. A loss ratio of 0 indicates attacks solely on L_{dummy} , whereas a loss ratio of ∞ indicates attacks solely on L_g . \uparrow (\downarrow) means a higher(lower) value is favored.

Dataset	Metric	Loss Ratio						
		0	0.01	0.1	1	10	100	∞
3D-FRONT	LPIPS-depth \downarrow	0.22	0.44	0.44	0.41	0.44	0.46	0.46
	LPIPS-gray \downarrow	0.46	0.49	0.50	0.49	0.51	0.50	0.47
	SSIM-depth \uparrow	0.89	0.77	0.77	0.79	0.78	0.78	0.78
	SSIM-gray \uparrow	0.70	0.64	0.64	0.60	0.58	0.59	0.63
	Human-eval \uparrow	4.4	4.6	4.6	4.6	4.4	4.4	4.0
Hypersim	LPIPS-depth \downarrow	0.13	0.39	0.49	0.52	0.56	0.57	0.56
	LPIPS-gray \downarrow	0.47	0.41	0.43	0.46	0.47	0.47	0.48
	SSIM-depth \uparrow	0.96	0.82	0.78	0.78	0.78	0.78	0.78
	SSIM-gray \uparrow	0.78	0.72	0.69	0.66	0.65	0.66	0.61
	Human-eval \uparrow	4.6	5.0	4.4	4.8	4.8	4.8	5.0
ScanNet	LPIPS-depth \downarrow	0.22	0.20	0.44	0.50	0.56	0.63	0.65
	LPIPS-gray \downarrow	0.66	0.64	0.55	0.54	0.55	0.55	0.56
	SSIM-depth \uparrow	0.91	0.92	0.76	0.73	0.73	0.73	0.73
	SSIM-gray \uparrow	0.54	0.58	0.52	0.51	0.49	0.49	0.53
	Human-eval \uparrow	4.0	4.4	4.0	4.2	4.2	4.2	4.4

by adding noises to the labels [11, 32]. In a NeRF network, labels are the density and color values of points in 3D space, which are implicit and must be integrated to determine the pixel colors. We introduce a basic solution NoisyLabelNeRF, which protects privacy by adding noises to the image pixels. NoisyLabelNeRF can be outlined as follows: for each ray r , whose RGB ground truth value is $C(r)$, we apply

$$\langle C(r) \rangle = C(r) + \xi, \quad \xi \sim \mathcal{N}(\mathbf{0}, \sigma_l^2),$$

where $\mathcal{N}(\mathbf{0}, \sigma_l^2)$ represents a Gaussian random variable with variance σ_l . This altered RGB value $\langle C(r) \rangle$ is then utilized in the loss function during NeRF training, enabling successful backpropagation. While NoisyLabelNeRF safeguards the privacy of the image data, the model's privacy during NeRF training may remain vulnerable. The introduction of noise in the image dataset can obscure the network parameters, but it also unavoidably diminishes the model's utility. Moreover, it may not be able to defend against attacks that exploit gradient sharing. Consideration of more advanced privacy-preserving methods is essential to address these limitations and vulnerabilities.

S²NeRF. The S²NeRF framework entails obfuscating gradient information, which malicious entities could potentially leverage [17, 55]. This is achieved by introducing randomly generated noise, including Gaussian or Laplace distributed ones, to obscure the genuine gradient, thereby impeding access to the valuable information during the gradient sharing [1, 13, 21, 46, 48]. In the defense mechanism, the noise applied to the gradients diminishes in intensity relative to the training iteration index, with the noise scale decreasing exponentially as training advances. In our algorithm, Gaussian noise is introduced to the shared gradients of each training batch g_1, \dots, g_B ,

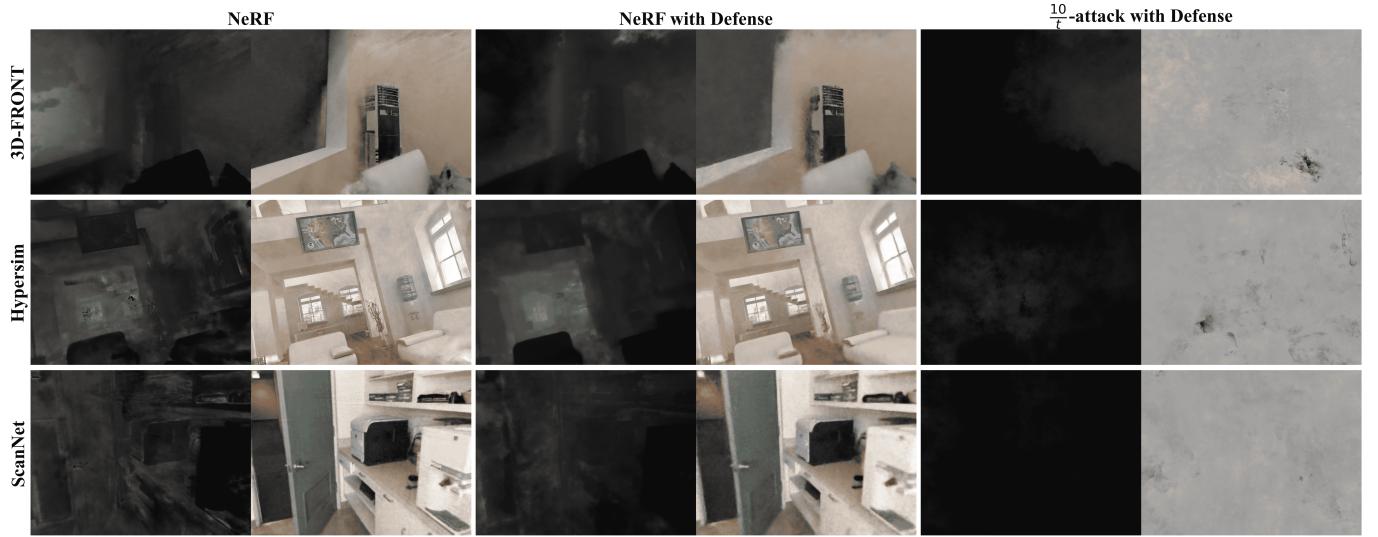


Figure 5: The defense results of S^2 NeRF with the configure $c = 1.2, r = 0.0001$ under $\frac{10}{t}$ -attack, where c and r denote the noise scale and noise decay ratio. S^2 NeRF achieves a balanced trade-off between privacy protection and model utility. In particular, the attacker can not recover any useful information, while the NeRF with defense can still reflect the appearance of the scene, albeit with some noise in the views.

where B represents the batch size, in the following manner:

$$\langle g_i \rangle = g_i + \eta_i, \quad \eta_i \sim \mathcal{N}(\mathbf{0}, \sigma_t^2 I_d),$$

Here, $\sigma_t = c \max_{i=1,\dots,B} \|g_i\| r^{\frac{t}{T}}$, where d denotes the dimension of the gradients and I_d is the d -by- d identity matrix. The magnitude of σ_t is regulated as follows: the coefficient c acts as a scaling factor determining the noise magnitude, while r governs the decay rate. Consequently, the noise intensity not only diminishes as training progresses but also adjusts to the volume of the gradients. This adaptive approach enables the noise addition to dynamically adapt to various training scenarios.

In the realm of deep learning, it has been established that under certain conditions, decaying noise does not harm the eventual convergence of neural network training [43]. Moreover, our attack experiments have unveiled that the surrogate model has the capacity to absorb significant information in the initial phases of training, emphasizing the crucial necessity for intensive noise injection during these early stages.

Different Noises’ Influence. In S^2 NeRF, the noise only needs to satisfy zero mean, with variance proportional to the norm of the gradient. In the domain of privacy-preserving deep learning, Gaussian noise is commonly used to meet this requirement [1, 42].

6.1 Algorithm Analysis

In this section, we first theoretically analyze the communication and computation complexity of S^2 NeRF, and then empirically evaluate the communication overhead and running time. Recall that the dimension of the cut layer is d . We denote the number of parameters of the client model as P , and the total training iterations as T .

Communication Complexity. The communication cost can be broken as: (1) the user uploads the sampled N_p points along N_r rays to the server($O(TN_r N_p)$ messages); (2) the server shares the cut

layer embeddings with the user($O(TN_r N_p d)$ messages); (3) The user returns the cut layer gradients to the server($O(TN_r N_p d)$ messages). Hence, the total communication overhead is $O(TN_r N_p d)$.

User Computation Complexity. The user’s computation cost can be broken as: (1) forward propagation and backpropagation computation of the client model($O(TN_r N_p P)$ complexity); (2) perform the noise adding($O(TN_r N_p d)$ complexity). Therefore, the user’s computation cost adds up to $O(TN_r N_p P)$.

Empirical Evaluation. For our main evaluations, we adopt the default setting of Instant-NGP [24] as the standard version. Specifically, the standard version uses $N_r = 4096$ and $N_p = 512$, with the client model consisting of a one-layer density MLP followed by a three-layer color MLP, each with hidden dimensions of 64. We use the standard version to better validate SplitNeRF’s privacy issues and S^2 NeRF’s security and utility. To reduce communication and computation stress, we design a light version of S^2 NeRF. The light version uses $N_r = 512$ and $N_p = 128$, with the client model consisting of a one-layer density MLP followed by a two-layer color MLP, each with hidden dimensions of 32. The communication overhead of a single iteration is 8 Megabytes. The user running time per iteration is 0.021, 0.019, and 0.021 seconds on the 3D-FRONT, Hypersim, and ScanNet datasets, respectively. We show in Appendix E that the light version S^2 NeRF still provides stable defense effectiveness and acceptable utility.

7 Empirical Evaluation for S^2 NeRF

In this section, we first assess the defense effectiveness and model utility of S^2 NeRF. Second, we conduct ablation studies to explore how various configurations of S^2 NeRF influence both defense effectiveness and model utility.

Table 4: S^2 NeRF ablation study under $\frac{10}{t}$ -attack with 0.01 loss ratio. We evaluate different noise scales c and decay ratio r . Experimental results indicate that S^2 NeRF with $c = 1.2, r = 0.0001$ achieves a better trade-off between privacy and utility. We use PSNR and LPIPS to compare utility and use LPIPS-depth, LPIPS-gray, SSIM-depth, SSIM-gray, and Human-eval to measure privacy. We report the utility results without defense, using GT as the baseline. We highlight $c = 1.2, r = 0.0001$ method in the red ground. ↑ (↓) means a higher(lower) value is favored.

Dataset	Metric	S^2 NeRF Configuration											
		$c = 0.6$ $r = 0.0001$	$c = 0.6$ $r = 0.001$	$c = 0.6$ $r = 1$	$c = 1.2$ $r = 0.0001$	$c = 1.2$ $r = 0.001$	$c = 1.2$ $r = 1$	$c = 2.4$ $r = 0.0001$	$c = 2.4$ $r = 0.001$	$c = 2.4$ $r = 1$	$c = 4.8$ $r = 0.0001$	$c = 4.8$ $r = 0.001$	GT
3D-FRONT	LPIPS-depth ↑	0.30	0.35	0.50	0.42	0.43	0.60	0.42	0.45	0.56	0.52	0.52	-
	LPIPS-gray ↑	0.46	0.49	0.44	0.47	0.50	0.45	0.58	0.50	0.46	0.45	0.52	-
	SSIM-depth ↓	0.83	0.81	0.59	0.53	0.75	0.47	0.80	0.71	0.40	0.13	0.55	-
	SSIM-gray ↓	0.74	0.73	0.75	0.72	0.73	0.74	0.63	0.72	0.74	0.74	0.70	-
	Human-eval ↓	2.0	1.6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-
	PSNR ↑	21.04	20.95	19.70	20.83	20.82	18.13	20.64	20.25	17.32	20.00	19.42	21.24
	LPIPS ↓	0.42	0.47	0.71	0.45	0.49	0.77	0.51	0.61	0.80	0.65	0.67	0.39
Hypersim	LPIPS-depth ↑	0.41	0.51	0.52	0.51	0.54	0.60	0.60	0.60	0.66	0.61	0.67	-
	LPIPS-gray ↑	0.55	0.56	0.56	0.56	0.57	0.54	0.67	0.71	0.73	0.70	0.74	-
	SSIM-depth ↓	0.84	0.63	0.77	0.63	0.55	0.69	0.79	0.79	0.76	0.79	0.75	-
	SSIM-gray ↓	0.79	0.79	0.79	0.78	0.79	0.79	0.71	0.70	0.65	0.70	0.64	-
	Human-eval ↓	2.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-
	PSNR ↑	18.69	18.65	18.14	18.66	18.59	17.79	18.64	18.53	17.79	18.59	18.20	18.77
	LPIPS ↓	0.44	0.49	0.75	0.48	0.59	0.81	0.54	0.67	0.81	0.61	0.76	0.36
ScanNet	LPIPS-depth ↑	0.54	0.55	0.62	0.57	0.56	0.56	0.59	0.59	0.58	0.55	0.58	-
	LPIPS-gray ↑	0.69	0.68	0.65	0.67	0.67	0.64	0.71	0.65	0.63	0.67	0.66	-
	SSIM-depth ↓	0.71	0.54	0.30	0.49	0.59	0.44	0.43	0.38	0.34	0.43	0.42	-
	SSIM-gray ↓	0.60	0.60	0.60	0.60	0.60	0.60	0.61	0.59	0.59	0.60	0.60	-
	Human-eval ↓	1.2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-
	PSNR ↑	19.63	19.51	17.94	19.59	19.44	17.48	19.51	19.22	17.08	19.42	18.81	19.89
	LPIPS ↓	0.52	0.58	0.80	0.56	0.62	0.82	0.60	0.72	0.82	0.67	0.77	0.42

7.1 Experimental Setup

The experimental setup for implementing the NeRF model and the datasets remains consistent with the details provided in Section 5.1.

Defense Metrics. In the design of S^2 NeRF, we mainly consider two aspects of performance, defense effectiveness and model utility.

- **Defense Effectiveness.** We use the fidelity metrics(LPIPS-depth, LPIPS-gray, SSIM-depth, SSIM-color) in Section 5.1 to measure the decline of the attack effect, and thus to reflect the defense effectiveness.
- **Model Utility.** We use PSNR and LPIPS to measure the NeRF model utility of S^2 NeRF.

Defense Setting. To evaluate the defense robustness of S^2 NeRF, we expose it to three different attacks using varying learning rate schemes ($\frac{10}{t}, 0.1\frac{t}{f}, 0.001\frac{t}{f}$) and a loss ratio of 0.01. We specifically showcase the $\frac{10}{t}$ -attack in Section 7.2 and Section 7.3, because our experiments in Section 5.4 confirm it as the most potent attack configuration. The results of the $0.1\frac{t}{f}$ -attack and $0.001\frac{t}{f}$ -attack are similar to the $\frac{10}{t}$ -attack’s result, and these are detailed in Appendix C.

7.2 Effectiveness

Results. The defense outcomes of S^2 NeRF in Figure 5 are configured with $c = 1.2, r = 0.0001$. We observed a significant reduction in the attack’s effectiveness: the images generated by the attacker contained little to no useful information. The defense effectiveness is more obvious when compared to the successful attack depicted in Figure 3.

Regarding model utility, the quality of NeRF generation with S^2 NeRF does experience a slight reduction. However, this decrease remains within acceptable limits. While the clarity of object boundaries may be somewhat diminished, the overall structure of the scene is still rendered clearly, preserving the essential visual integrity of the model.

Optimal Defense Setting. An ideal defense setting should demonstrate the capability to thwart attacks while maintaining acceptable model utility effectively. Additionally, the defense parameters should be versatile and applicable across various scenarios.

In Figure 5, we confirm that this defense setting provides strong defensive effectiveness and maintains good model utility across three distinct datasets. Despite variations in the parameters of the NeRF network models trained in different scenarios, S^2 NeRF’s reliance on gradient norms enhances its generalizability.

Defense for Scene-aided Surrogate Model Attack. In Section 4.2, the success of Scene-aided Surrogate Model Attack is heavily dependent on the effectiveness of the preceding Surrogate Model Attack. Scene-aided Surrogate Model Attack specifically needs a successful Surrogate Model Attack model to accurately determine the poses closest to the leaked images. Our defense results demonstrate a robust capability to neutralize the Surrogate Model Attack, making it impossible to extract any useful information. With depth views rendered entirely black and color views appearing gray, Surrogate Model Attack is unable to provide the necessary insights for Scene-aided Surrogate Model Attack to function effectively. Consequently, Scene-aided Surrogate Model Attack is ineffective under this condition, affirming that S²NeRF provides a strong defense against both the Surrogate Model Attack and Scene-aided Surrogate Model Attack.

7.3 S²NeRF Ablation Study

In this section, we evaluate the performances of various defense configurations within the S²NeRF framework. Our objective is to identify the defense settings that offer the best balance between robust defense effectiveness and acceptable model utility.

S²NeRF Results. For S²NeRF, we explore noise scales c at set $\{0.6, 1.2, 2.4, 4.8\}$ and decay ratio r at $\{0.0001, 0.001, 1\}$, where $r = 1$ indicates adding noise with no decay. The quantitative results are documented in Table 4, with visual results available in Appendix C.

Through our evaluations, we find that $c = 1.2, r = 0.0001$ offers a balanced defense across all three datasets. This setting provides better model utility compared to the non-decaying defense mechanism($r = 1$). A lower noise scale though yielding higher model utility, results in considerable privacy compromises, which are unacceptable. For example, while the $c = 0.6, r = 0.0001$ setting achieves a PSNR of 21.04 in 3D-FRONT dataset, it also allows the attacker to reach an SSIM-depth of 0.83, indicating a significant privacy breach.

Conversely, the setup of $c = 1.2, r = 0.0001$ maintains both high utility and robust defense effectiveness. Specifically, this setup achieves a PSNR of 20.83 in 3D-FRONT, a decrease of only 2% from the ground truth of 21.24, and provides a strong privacy guarantee, as indicated by a low SSIM-depth value of 0.53. On the other hand, a higher noise scale($c = 2.4$) enhances defense effectiveness but at the cost of reduced model utility. An intuitive comparison of these effects is detailed in Appendix C.

NoisyLabelNeRF Results. For NoisyLabelNeRF, we explore varying the noise scale σ_l at levels of $\{0.5, 1, 2, 4, 8\}$. The quantitative results are presented in Table 4, and the visual results are available in Appendix C.

Overall, NoisyLabelNeRF has proven to be ineffective. Even with high noise levels, it fails to provide a stable defense against attacks. For instance, at a noise scale of $\sigma_l = 8$, although the model utility significantly decreases (PSNR of 19.17 and LPIPS of 0.68 on 3D-FRONT), the attacker's LPIPS-depth remains low at 0.29, indicating that the defense does not sufficiently obscure the information from the attacker. More detailed visual effects demonstrating these outcomes can be viewed in Appendix D.

Table 5: NoisyLabelNeRF results under $\frac{10}{t}$ -attack with 0.01 loss ratio. We report the utility results without defense, using GT as the baseline. NoisyLabelNeRF proves ineffective: it fails to defend against attacks when the noise level is high and significantly compromises model utility. \uparrow (\downarrow) means a higher(lower) value is favored.

Dataset	Metric	The Scale of the Noise					GT
		0.5	1	2	4	8	
3D-FRONT	LPIPS-depth \uparrow	0.25	0.19	0.20	0.25	0.29	-
	LPIPS-gray \uparrow	0.41	0.47	0.42	0.48	0.48	-
	SSIM-depth \downarrow	0.89	0.95	0.94	0.90	0.90	-
	SSIM-gray \downarrow	0.76	0.73	0.76	0.74	0.73	-
	Human-eval \downarrow	4.0	3.8	3.6	3.0	2.0	-
	PSNR \uparrow	20.48	19.93	20.42	20.14	19.17	21.24
	LPIPS \downarrow	0.40	0.47	0.51	0.64	0.68	0.39
Hypersim	LPIPS-depth \uparrow	0.22	0.25	0.33	0.42	0.52	-
	LPIPS-gray \uparrow	0.60	0.58	0.55	0.57	0.62	-
	SSIM-depth \downarrow	0.93	0.92	0.89	0.86	0.80	-
	SSIM-gray \downarrow	0.80	0.80	0.80	0.79	0.76	-
	Human-eval \downarrow	4.0	3.8	3.6	3.0	2.2	-
	PSNR \uparrow	18.69	18.72	18.66	18.66	18.48	18.77
	LPIPS \downarrow	0.41	0.47	0.56	0.69	0.80	0.36
ScanNet	LPIPS-depth \uparrow	0.29	0.33	0.43	0.47	0.44	-
	LPIPS-gray \uparrow	0.65	0.65	0.69	0.70	0.74	-
	SSIM-depth \downarrow	0.91	0.90	0.84	0.80	0.81	-
	SSIM-gray \downarrow	0.59	0.59	0.60	0.58	0.53	-
	Human-eval \downarrow	5.0	5.0	3.8	3.0	1.0	-
	PSNR \uparrow	19.71	19.58	19.48	19.50	18.76	19.89
	LPIPS \downarrow	0.50	0.55	0.61	0.65	0.75	0.42

8 Discussion

Privacy Assessment on Reconstructed Images. The Surrogate Model Attack aligns with the goals of traditional reconstruction attacks that intercept gradients from a model to recreate images, potentially exposing sensitive details of the original images [8, 55]. Our evaluation assesses whether these reconstructed images compromise the privacy of the original scenes, a concern highlighted in recent studies [35]. Common metrics like PSNR, LPIPS, and structural similarity index(SSIM) are traditionally used to measure pixel-level image similarity. High similarity scores typically indicate a successful reconstruction attack and greater model vulnerability to privacy breaches. Conversely, low scores suggest reduced privacy risks. However, the effectiveness of these metrics in accurately reflecting human perceptions of privacy is questionable due to the subjective nature of privacy [35].

To address these challenges in NeRF scenarios, we introduce two fidelity metrics based on LPIPS: LPIPS-depth, and LPIPS-gray; and two fidelity metrics based on SSIM: SSIM-depth, and SSIM-gray. These metrics aim to better align with human perception, though they may still contain inaccuracies. At the same time, we also conduct human evaluations, adding the Human-eval metric to assess the level of privacy disclosure more realistically.

The Trade-off Between Utility and Privacy. Balancing utility (model performance) and privacy is crucial in privacy-enhancing

technologies. A key challenge is how to effectively add noise for privacy without sacrificing training performance. Our defense approach, which incorporates a noise decay mechanism, offers a better trade-off than traditional noise injection, yet there is potential for further improvement. Previous strategies have aimed to mitigate the impact of noise on model utility [26, 38, 42, 50, 56]. However, such methods are less effective for NeRF, where parameters directly represent a scene’s 3D structure and are inherently sensitive. To enhance NeRF’s utility while preserving privacy, a promising avenue is enabling the client side to efficiently fine-tune models after noise addition, representing a specific direction for research in privacy-preserving NeRF training.

S^2 NeRF for Other NeRF Architectures. In our paper, we adopt Instant-NGP [24], a popular and efficient NeRF framework. Instant-NGP is widely used, as evidenced by state-of-the-art implementations like Zip-NeRF [2], which are based on its structure. Our proposed SplitNeRF can also be applied to other MLP-based NeRF models, such as vanilla NeRF [20] and D-NeRF [31]. Surrogate Model Attack, Scene-aided Surrogate Model Attack, and S^2 NeRF can be utilized with these similar SplitNeRF architectures.

Reconstruction Attacks against Split Learning. Reconstruction attacks [8, 53, 55] typically necessitate access to full model parameters and gradients to reconstruct training data. However, our attacks focus on recovering client model parameters and training data, which differs significantly. Although some studies [18, 51] have reduced the optimization space using GANs and over-parameterized networks, the optimization space for client model parameters remains extensive. In our paper, our attack methods leverage server model parameters and gradients effectively, demonstrating relative strength.

The Practical Deployment Challenges of S^2 NeRF. The main challenges of deploying S^2 NeRF come from two aspects, one is the frequent transmission of high-dimensional features and back-propagated gradients over bandwidth-limited wireless channels, and the other is the difficulty of training deep models on resource-constrained edge networks. Model pruning [4, 19] directly eliminates the number of parameters of the neural network. Sparsification or quantization techniques can help compress intermediate embeddings and gradients, thereby reducing communication overhead [25, 29, 54]. For instance, the work [54] achieves 34.96 \times compression with only a 2.9% accuracy decrease on CIFAR-100 datasets. These solutions facilitate the deployment of S^2 NeRF.

9 Related Work

Information Leakage in Split Learning. Label leakage under split learning, initially investigated in the context of advertisement conversion prediction [17], involves an attack that leverages the differences in shared gradients between positive and negative data samples to achieve a high AUC score for inferring labels. However, this approach is restricted to binary classification. The Unsplit attack [5] employs a gradient-matching strategy to minimize the mean square error between the original gradients and those of a surrogate model to infer labels. Additionally, Fu et al. [9] show that a malicious server can utilize a partially trained model to launch model completion attacks with few labeled data. The classic gradient matching [55] utilizes a learning-based method to infer

the data and labels by minimizing the distance of gradients under the white-box setting (known model). Similarly, the work [46] has a stronger attack setting to focus on the black-box regression problem setting, where the attacker does not know the target model.

Above all, all the previous works mainly utilize the difference of the gradients for the binary or multi-classification problem, or regression problem, which are not fit for our NeRF problem. To the best of our knowledge, we are the first to study the leakage against the split learning under the NeRF problem.

Privacy protection in Split Learning. To address these privacy concerns, protection is primarily achieved through two categories of methods. The first involves using advanced cryptographic protocols, such as *secure multiparty computation*[22, 40] and *two-party computation*[23, 28, 45]. The second category employs perturbation-based methods that introduce randomness into the data-sharing process to protect privacy. This includes obfuscating the shared gradient information among parties or directly perturbing the information that needs protection [1, 6, 11, 32, 34, 44, 48]. For example, Sun et al. [34] focus on adding noise to gradients as a means to safeguard the underlying data from potential privacy breaches. Our proposed S^2 NeRF belongs to the perturbation defend methods, as we add noises to the gradients to protect privacy.

10 Conclusion

In this paper, we propose the pioneering split-learning NeRF framework SplitNeRF to address the NeRF training data privacy issues. Concretely, we design an algorithm framework where the client and server can train NeRF collaboratively, while the client does not need to transfer local private scene data to the server. While the SplitNeRF framework appears inherently private, our sequentially devised Surrogate Model Attack and Scene-aided Surrogate Model Attack compromise the privacy of SplitNeRF. The experimental results demonstrate that the SplitNeRF framework still has a serious privacy breach. We further propose S^2 NeRF, secure SplitNeRF, which can strike a trade-off between model utility and privacy protection in various scenarios. Extensive experiments on three widely used public NeRF datasets illustrate the effectiveness of S^2 NeRF.

Acknowledgments

We would like to thank our shepherd and the anonymous reviewers for their insightful comments. This work was supported in part by NSFC under grant 62336005; in part by the Shenzhen Science and Technology Program JCYJ20210324120011032; in part by the Guangdong Provincial Key Laboratory of Big Data Computing, the Chinese University of Hong Kong, Shenzhen.

References

- [1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.
- [2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2023. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 19697–19705.
- [3] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. 2017. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5828–5839.

- [4] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. 2020. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proc. IEEE* 108, 4 (2020), 485–532.
- [5] Ege Erdogan, Alptekin Küpcü, and A Ercument Çicek. 2022. Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society*. 115–124.
- [6] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2468–2479.
- [7] Siwei Feng and Han Yu. 2020. Multi-participant multi-class vertical federated learning. *arXiv preprint arXiv:2001.11154* (2020).
- [8] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333.
- [9] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. 2022. Label inference attacks against vertical federated learning. In *31st USENIX security symposium (USENIX Security 22)*. 1397–1414.
- [10] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengye Sun, Rongfei Jia, Binqiang Zhao, et al. 2021. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10933–10942.
- [11] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. 2021. Deep learning with label differential privacy. *Advances in neural information processing systems* 34 (2021), 27131–27145.
- [12] Otkrist Gupta and Ramesh Raskar. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications* 116 (2018), 1–8.
- [13] Lingying Huang, Junfeng Wu, Dawei Shi, Subhrakanti Dey, and Ling Shi. 2024. Differential Privacy in Distributed Optimization with Gradient Tracking. *IEEE Trans. Automat. Control* (2024).
- [14] Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. 2021. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in neural information processing systems* 34 (2021), 7232–7241.
- [15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [16] Chunlin Li, Ruofan Liang, Hanru Fan, Zhenghen Zhang, Sankeerth Durvasula, and Nandita Vijaykumar. 2024. DISORF: A Distributed Online NeRF Training and Rendering Framework for Mobile Robots. *arXiv preprint arXiv:2403.00228* (2024).
- [17] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. 2021. Label Leakage and Protection in Two-party Split Learning. In *International Conference on Learning Representations*.
- [18] Zhiuhang Li, Jiaxin Zhang, Luyang Liu, and Jian Liu. 2022. Auditing privacy defenses in federated learning via generative gradient leakage. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10132–10142.
- [19] Zheng Lin, Guanqiao Qu, Xianhao Chen, and Kaibin Huang. 2024. Split learning in 6g edge networks. *IEEE Wireless Communications* (2024).
- [20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.
- [21] Meisam Mohammady, Shangyu Xie, Yuan Hong, Mengyuan Zhang, Lingyu Wang, Makan Pourzandi, and Mourad Debbabi. 2020. R2dp: A universal and automated approach to optimizing the randomization mechanisms of differential privacy for utility metrics with no known optimal distributions. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 677–696.
- [22] Payman Mohassel and Peter Rindal. 2018. ABY3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 35–52.
- [23] Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 19–38.
- [24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages. <https://doi.org/10.1145/3528223.3530127>
- [25] Yongjeong Oh, Jaeho Lee, Christopher G Brinton, and Yo-Seb Jeon. 2023. Communication-Efficient Split Learning via Adaptive Feature-Wise Compression. *arXiv preprint arXiv:2307.10805* (2023).
- [26] Nicolas Papernot, Martin Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2016. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755* (2016).
- [27] Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. 2021. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2113–2129.
- [28] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalamé. 2021. {ABY2.0}: Improved {Mixed-Protocol} secure {Two-Party} computation. In *30th USENIX Security Symposium (USENIX Security 21)*. 2165–2182.
- [29] Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668* (2018).
- [30] Thomas Porter and Tom Duff. 1984. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. 253–259.
- [31] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10318–10327.
- [32] Haoze Qiu, Fei Zheng, Chaochao Chen, and Xiaolin Zheng. 2023. Defending Label Inference Attacks in Split Learning under Regression Setting. *arXiv preprint arXiv:2308.09448* (2023).
- [33] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. 2021. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10912–10922.
- [34] Jiankai Sun, Xin Yang, Yuanshun Yao, Junyuan Xie, Di Wu, and Chong Wang. 2023. Dpauc: Differentially private auc computation in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 15170–15178.
- [35] Xiaoxiao Sun, Nidham Gazagnadou, Vivek Sharma, Lingjuan Lyu, Hongdong Li, and Liang Zheng. 2024. Privacy Assessment on Reconstructed Images: Are Existing Evaluation Metrics Faithful to Human Perception? *Advances in Neural Information Processing Systems* 36 (2024).
- [36] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nitin Raghavan, Utkarsh Singh, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems* 33 (2020), 7537–7547.
- [37] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. 2022. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8485–8493.
- [38] Florian Tramer and Dan Boneh. 2020. Differentially private learning needs better features (or much more data). *arXiv preprint arXiv:2011.11660* (2020).
- [39] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564* (2018).
- [40] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. 2020. Falcon: Honest-majority maliciously secure framework for private deep learning. *arXiv preprint arXiv:2004.02229* (2020).
- [41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.
- [42] Zihao Wang, Rui Zhu, Dongrui Zhou, Zhikun Zhang, John Mitchell, Haixu Tang, and XiaoFeng Wang. 2024. DPAdapter: Improving Differentially Private Deep Learning through Noise Tolerance Pre-training. *arXiv preprint arXiv:2403.02571* (2024).
- [43] Rachel Ward, Xiaoxia Wu, and Leon Bottou. 2020. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. *Journal of Machine Learning Research* 21, 219 (2020), 1–30.
- [44] Shangyu Xie and Yuan Hong. 2022. Differentially private instance encoding against privacy attacks. In *In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*.
- [45] Shangyu Xie, Meisam Mohammady, Han Wang, Lingyu Wang, Jaideep Vaidya, and Yuan Hong. 2021. A generalized framework for preserving both privacy and utility in data outsourcing. *IEEE transactions on knowledge and data engineering* 35, 1 (2021), 1–15.
- [46] Shangyu Xie, Xin Yang, Yuanshun Yao, Tianyi Liu, Taiqing Wang, and Jiankai Sun. 2023. Label inference attack against split learning under regression setting. *arXiv preprint arXiv:2301.07284* (2023).
- [47] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.
- [48] Xin Yang, Jiankai Sun, Yuanshun Yao, Junyuan Xie, and Chong Wang. 2022. Differentially private label protection in split learning. *arXiv preprint arXiv:2203.02073* (2022).
- [49] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. 2021. inferf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1323–1330.
- [50] Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. 2021. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. *arXiv*

- preprint arXiv:2102.12677 (2021).*
- [51] Chi Zhang, Zhang Xiaoman, Ekanut Sotthiwat, Yanyu Xu, Ping Liu, Liangli Chen, and Yong Liu. 2023. Generative gradient inversion via over-parameterized networks in federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5126–5135.
 - [52] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.
 - [53] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610* (2020).
 - [54] Fei Zheng, Chaochao Chen, Lingjuan Lyu, and Binhu Yao. 2023. Reducing communication for split learning by randomized top-k sparsification. *arXiv preprint arXiv:2305.18469* (2023).
 - [55] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems* 32 (2019).
 - [56] Yuqing Zhu, Xiang Yu, Manmohan Chandraker, and Yu-Xiang Wang. 2020. Private-knn: Practical differential privacy for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11854–11862.

A Additional Experimental Results of Attack Ablation Study

Different Learning Rate Decay Schemes’ Results. Figure 6 presents the outcomes of the $0.1^{\frac{t}{T}}$ -attack and $0.001^{\frac{t}{T}}$ -attack, both employing a loss ratio of 0.01. Generally, these attacks are slightly less effective compared to the $\frac{10}{t}$ attacks. They reconstruct some aspects of the original scene’s depth perspective, such as parts of the sofa in the 3D-FRONT dataset, the water dispenser and picture frame in Hypersim, and the door outline in ScanNet. However, the level of detail and accuracy in these reconstructions falls short of what is achieved with the $\frac{10}{t}$ -attack.

Different Surrogate Model’s Loss Ratios’ Results. Figure 11 displays the performance of the $\frac{10}{t}$ -attack across various loss ratios, ranging from $[0, 0.01, 0.1, 1, 10, 100, \infty]$. These attacks effectively capture scene information across different loss ratio settings. Notably, as the loss ratio increases from low to high, the attack’s focus shifts from predominantly depth views to color views, illustrating a variation in attack impact influenced by the degree of model dependency on gradient information. Overall, under the $\frac{10}{t}$ -attack, varying loss ratios consistently deliver potent attack outcomes, underscoring the effectiveness and robustness of this attack strategy.

B Additional Experimental Results of Surrogate Models with Different Structures

Attack Experiments. We conduct attack experiments with two additional surrogate model structures: a one-layer density MLP with a two-layer color MLP (referred to as the two-layer color MLP attack), and a one-layer density MLP with a four-layer color MLP (referred to as the four-layer color MLP attack). In each attack, we utilize $\frac{10}{t}$ learning rate decay schemes and the loss ratio of 0.01. The results of these two attacks are shown in Table 6 and Figure 8.

The results indicate that both attacks are highly effective. Specifically, the attacker’s view successfully reveals the outline of the original scene. The human-eval metric is generally high, with most values exceeding 4.5, indicating serious privacy disclosure.

$S^2\text{NeRF}$ ’s Defense Effectiveness. Based on the configuration of $c = 1.2, r = 0.0001$, we evaluate the defense effectiveness of $S^2\text{NeRF}$ against the two-layer color MLP attack and the four-layer color

MLP attack. The defense results for these two attacks are shown in Table 7 and Figure 9.

The results demonstrate that $S^2\text{NeRF}$ can effectively defend against attacks from different surrogate model structures. Visually, neither attack can recover any private information, with the attack views appearing nearly completely black or gray. From the perspective of attack metrics, the effectiveness of the attacks is significantly reduced, with the human-eval score dropping from about 4.5 to around 1.

C Additional Experimental Results of $S^2\text{NeRF}$ Ablation Study

$S^2\text{NeRF}$ ’s Defense Effectiveness against Other Attacks. Figure 7 demonstrates the defense effectiveness of $S^2\text{NeRF}$ against the $0.1^{\frac{t}{T}}$ -attack and $0.001^{\frac{t}{T}}$ -attack. The results clearly indicate that the attackers are unable to access any sensitive information about the scene, showcasing the robustness and universality of $S^2\text{NeRF}$ ’s defensive capabilities. This effectiveness is particularly pronounced when compared to the outcomes of successful attacks depicted in Figure 6, highlighting the strength of $S^2\text{NeRF}$ ’s defense mechanisms.

Experimental Results of Different Configures of $S^2\text{NeRF}$. The performance of $S^2\text{NeRF}$ under various noise scales(c) and decay ratios(r) is documented across multiple figures and tables: Figure 12, Figure 13, Figure 14, Figure 15, Table 8, and Table 9.

Overall, configurations with a decay rate of 0.0001 exhibit better NeRF utility. However, lower noise scale configs($c = 0.6$) are less effective in defending, allowing attackers to recover parts of the scene’s outline still. For instance, in the 3D-FRONT dataset, as shown in Figure 12 and Figure 13, attackers are able to discern outlines of sofas and air conditioners. On the other hand, a larger noise decay ratio r (i.e., 0.001, 1) leads to slower noise decay, which, while resisting attacks, significantly diminishes model utility. The scene outlines become blurred, and the rendered images exhibit considerable noise. Consequently, a configuration of $S^2\text{NeRF}$ with $c = 1.2, r = 0.0001$ strikes the optimal balance, effectively countering attacks while preserving good model utility.

D Additional Experimental Results of NoisyLabelNeRF

Figure 16 and Figure 17 display the performance of NoisyLabelNeRF under various noise scales σ_l . The figures clearly demonstrate that NoisyLabelNeRF is ineffective. Even with high noise levels($\sigma_l = 4, 8$), the attacker is still able to successfully recover scene information. For example, under the Hypersim dataset, the outline of the picture frame and the room remain discernible despite the noise.

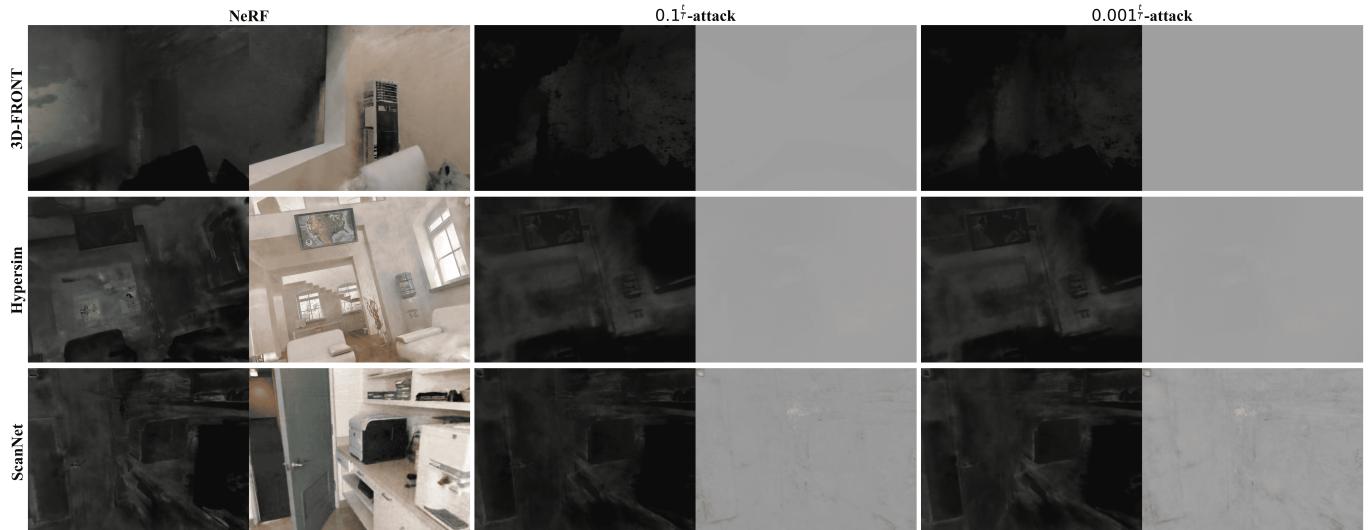
Additionally, the model’s utility significantly deteriorates as the noise scale increases, leading to distorted scene colors that render the results unacceptable. This performance starkly contrasts with our proposed $S^2\text{NeRF}$, underscoring its superior effectiveness and the inadequacies of NoisyLabelNeRF in providing robust defense while maintaining acceptable model utility.

Table 6: Attack experiment results of surrogate models with different structures. We use LPIPS-depth, LPIPS-gray, SSIM-depth, SSIM-gray, and Human-eval to measure attack effectiveness. \uparrow (\downarrow) means a higher(lower) value is favored.

Surrogate Model Structures	Dataset	Metric				
		LPIPS-depth \downarrow	LPIPS-gray \downarrow	SSIM-depth \uparrow	SSIM-gray \uparrow	Human-eval \uparrow
Two-layer color MLP	3D-FRONT	0.26	0.54	0.87	0.58	5.0
	Hypersim	0.40	0.42	0.83	0.72	5.0
	ScanNet	0.20	0.63	0.92	0.59	4.6
Four-layer color MLP	3D-FRONT	0.45	0.54	0.78	0.66	3.4
	Hypersim	0.48	0.45	0.79	0.68	5.0
	ScanNet	0.19	0.65	0.93	0.58	4.6

Table 7: S^2 NeRF defense results against surrogate models with different structures. We use LPIPS-depth, LPIPS-gray, SSIM-depth, SSIM-gray, and Human-eval to measure S^2 NeRF’s defense effectiveness. \uparrow (\downarrow) means a higher(lower) value is favored.

Surrogate Model Structures	Dataset	Metric				
		LPIPS-depth \uparrow	LPIPS-gray \uparrow	SSIM-depth \downarrow	SSIM-gray \downarrow	Human-eval \downarrow
Two-layer color MLP	3D-FRONT	0.35	0.47	0.78	0.71	1.0
	Hypersim	0.43	0.59	0.82	0.80	1.2
	ScanNet	0.59	0.68	0.57	0.56	1.2
Four-layer color MLP	3D-FRONT	0.71	0.73	0.53	0.39	1.0
	Hypersim	0.63	0.69	0.77	0.67	1.0
	ScanNet	0.56	0.70	0.56	0.62	1.2

**Figure 6:** Comparison of Surrogate Model Attack across the depth and color views on the three datasets, utilizing 0.1^{-t} and 0.001^{-t} learning rate decay schemes.

E Experimental Results of Light Version S^2 NeRF

The defense outcomes of the light version S^2 NeRF in Figure 10 are configured with $c = 1.2$ and $r = 0.0001$. Although the utility results

for the light version are slightly worse than those of the standard version shown in Figure 5, they still maintain high availability. Additionally, the light version S^2 NeRF remains resistant to attacks.

Table 8: Defense method ablation study under $0.1^{\frac{t}{T}}$ -attack. We evaluate different noise scales c and decay ratio r . Experimental results indicate that $S^2\text{NeRF}$ with $c = 1.2, r = 0.0001$ achieves a better trade-off between privacy and utility. We use PSNR and LPIPS to compare utility and use LPIPS-depth, LPIPS-gray, SSIM-depth, SSIM-gray, and Human-eval to measure privacy. We report the utility results without defense, using GT as the baseline. We highlight $c = 1.2, r = 0.0001$ method in the red ground. ↑ (↓) means a higher(lower) value is favored.

Dataset	Metric	$S^2\text{NeRF}$ Configuration											
		$c = 0.6$ $r = 0.0001$	$c = 0.6$ $r = 0.001$	$c = 0.6$ $r = 1$	$c = 1.2$ $r = 0.0001$	$c = 1.2$ $r = 0.001$	$c = 1.2$ $r = 1$	$c = 2.4$ $r = 0.0001$	$c = 2.4$ $r = 0.001$	$c = 2.4$ $r = 1$	$c = 4.8$ $r = 0.0001$	$c = 4.8$ $r = 0.001$	GT
3D-FRONT	LPIPS-depth ↑	0.46	0.41	0.60	0.35	0.49	0.65	0.46	0.57	0.71	0.61	0.58	-
	LPIPS-gray ↑	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	-
	SSIM-depth ↓	0.74	0.81	0.26	0.77	0.71	0.41	0.65	0.63	0.28	0.46	0.42	-
	SSIM-gray ↓	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	-
	Human-eval ↓	2.0	1.6	1.0	1.4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-
	PSNR ↑	21.06	20.84	18.88	21.05	20.75	19.05	20.82	20.55	17.08	20.08	19.13	21.24
	LPIPS ↓	0.42	0.47	0.75	0.43	0.51	0.79	0.48	0.60	0.83	0.69	0.73	0.39
Hypersim	LPIPS-depth ↑	0.49	0.51	0.66	0.52	0.53	0.64	0.49	0.61	0.67	0.53	0.63	-
	LPIPS-gray ↑	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	-
	SSIM-depth ↓	0.69	0.75	0.54	0.46	0.74	0.59	0.73	0.65	0.37	0.56	0.51	-
	SSIM-gray ↓	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	-
	Human-eval ↓	2.0	1.0	1.0	1.6	1.0	1.0	1.2	1.0	1.0	1.0	1.0	-
	PSNR ↑	18.75	18.66	18.12	18.72	18.62	17.81	18.63	18.57	17.81	18.49	18.38	18.77
	LPIPS ↓	0.43	0.49	0.77	0.47	0.58	0.80	0.56	0.70	0.80	0.63	0.75	0.36
ScanNet	LPIPS-depth ↑	0.52	0.56	0.60	0.55	0.55	0.57	0.51	0.55	0.57	0.51	0.57	-
	LPIPS-gray ↑	0.74	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	-
	SSIM-depth ↓	0.41	0.35	0.48	0.38	0.43	0.36	0.62	0.59	0.24	0.66	0.41	-
	SSIM-gray ↓	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.62	0.62	-
	Human-eval ↓	1.2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-
	PSNR ↑	19.66	19.44	17.94	19.61	19.39	17.25	19.51	19.12	17.35	19.40	18.65	19.89
	LPIPS ↓	0.51	0.58	0.80	0.55	0.67	0.80	0.60	0.72	0.82	0.67	0.76	0.42



Figure 7: The results of $S^2\text{NeRF}$ under Surrogate Model Attack utilizing $0.1^{\frac{t}{T}}$ and $0.001^{\frac{t}{T}}$ learning rate decay schemes.

Table 9: Defense method ablation study under $0.001^{\frac{f}{T}}$ -attack. We evaluate different noise scales c and decay ratio r . Experimental results indicate that S²NeRF with $c = 1.2, r = 0.0001$ achieves a better trade-off between privacy and utility. We use PSNR and LPIPS to compare utility and use LPIPS-depth, LPIPS-gray, SSIM-depth, SSIM-gray, and Human-eval to measure privacy. We report the utility results without defense, using GT as the baseline. We highlight $c = 1.2, r = 0.0001$ method in the red ground. ↑ (↓) means a higher(lower) value is favored.

Dataset	Metric	S ² NeRF Configuration										GT
		$c = 0.6$ $r = 0.0001$	$c = 0.6$ $r = 0.001$	$c = 0.6$ $r = 1$	$c = 1.2$ $r = 0.0001$	$c = 1.2$ $r = 0.001$	$c = 2.4$ $r = 1$	$c = 2.4$ $r = 0.0001$	$c = 2.4$ $r = 0.001$	$c = 4.8$ $r = 1$	$c = 4.8$ $r = 0.0001$	
3D-FRONT	LPIPS-depth ↑	0.34	0.30	0.59	0.31	0.41	0.62	0.38	0.46	0.57	0.44	0.51
	LPIPS-gray ↑	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	-
	SSIM-depth ↓	0.82	0.86	0.65	0.83	0.76	0.24	0.74	0.77	0.57	0.64	0.62
	SSIM-gray ↓	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	-
	Human-eval ↓	3.4	2.6	1.0	1.8	1.0	1.0	1.6	1.0	1.0	1.0	-
	PSNR ↑	21.22	20.92	19.54	21.14	20.96	17.77	20.66	20.21	18.13	20.15	18.99
	LPIPS ↓	0.41	0.48	0.72	0.45	0.51	0.80	0.49	0.61	0.79	0.62	0.72
Hypersim	LPIPS-depth ↑	0.38	0.56	0.68	0.49	0.53	0.67	0.50	0.56	0.69	0.50	0.63
	LPIPS-gray ↑	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	-
	SSIM-depth ↓	0.83	0.69	0.52	0.76	0.75	0.49	0.76	0.66	0.41	0.74	0.69
	SSIM-gray ↓	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	-
	Human-eval ↓	2.4	2.2	1.0	1.6	1.0	1.0	1.2	1.0	1.0	1.0	-
	PSNR ↑	18.65	18.66	18.08	18.70	18.61	17.83	18.69	18.51	17.79	18.65	18.38
	LPIPS ↓	0.42	0.49	0.76	0.47	0.58	0.79	0.52	0.67	0.80	0.62	0.75
ScanNet	LPIPS-depth ↑	0.49	0.55	0.57	0.58	0.54	0.59	0.50	0.54	0.63	0.52	0.53
	LPIPS-gray ↑	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	-
	SSIM-depth ↓	0.49	0.33	0.46	0.29	0.47	0.21	0.70	0.57	0.57	0.56	0.66
	SSIM-gray ↓	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	-
	Human-eval ↓	1.2	1.2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-
	PSNR ↑	19.73	19.54	18.04	19.64	19.22	17.46	19.51	19.10	17.03	19.27	18.78
	LPIPS ↓	0.50	0.57	0.79	0.55	0.66	0.80	0.60	0.73	0.82	0.68	0.78



Figure 8: Surrogate Model Attack results of different surrogate model structures: two-layer color MLP attack and four-layer color MLP attack.

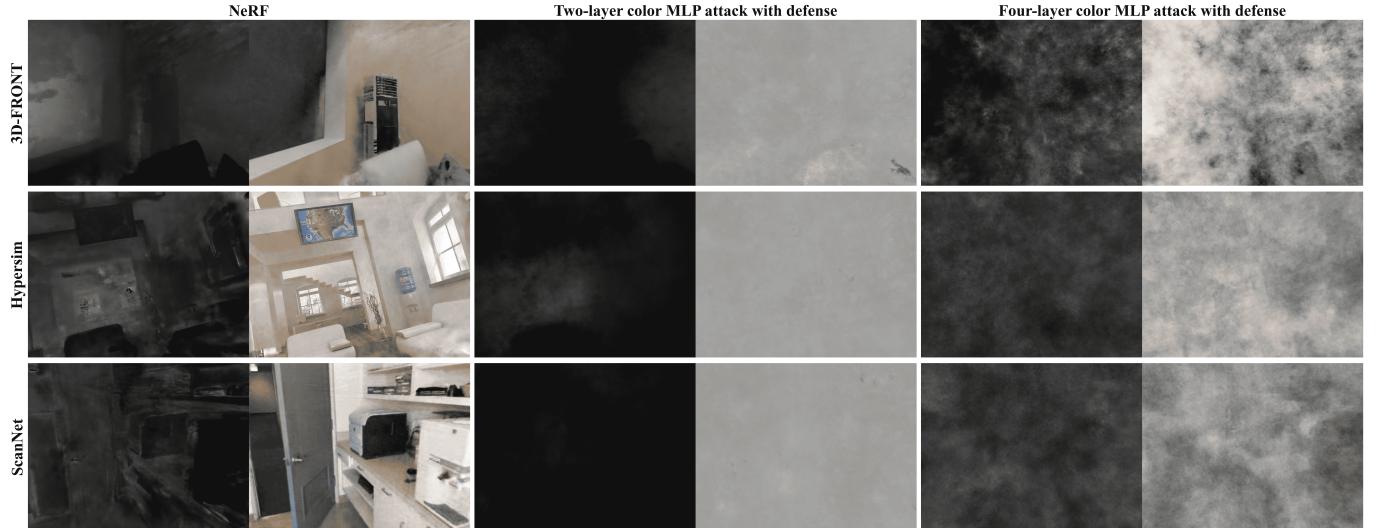


Figure 9: Defense results of S^2 NeRF with the configure $c = 4.8, r = 0.001$ under two-layer color MLP attack and four-layer color MLP attack.



Figure 10: The light version S^2 NeRF Results. The light version S^2 NeRF maintains stable defense effectiveness and acceptable utility.



Figure 11: Comparison of Surrogate Model Attack across the depth and color views on the three datasets, utilizing different loss ratios. A loss ratio of 0 indicates attacks solely on L_{dummy} , whereas a loss ratio of ∞ indicates attacks solely on L_g .



Figure 12: Experimental S^2 NeRF Results with the configuration $c = 0.6, r = 0.0001, 0.001, 1$.

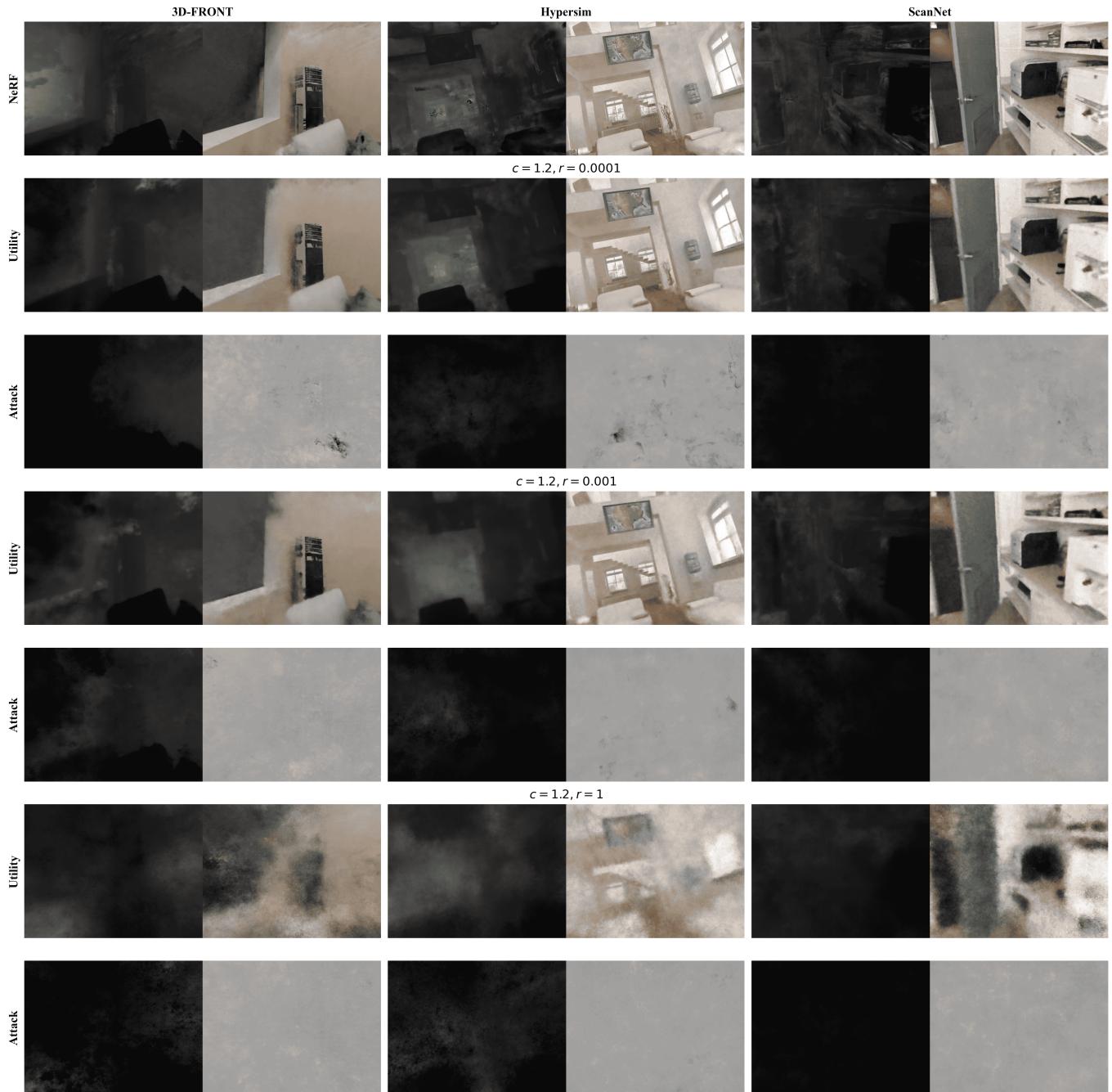


Figure 13: Experimental S^2 NeRF Results with the configuration $c = 1.2, r = 0.0001, 0.001, 1$.

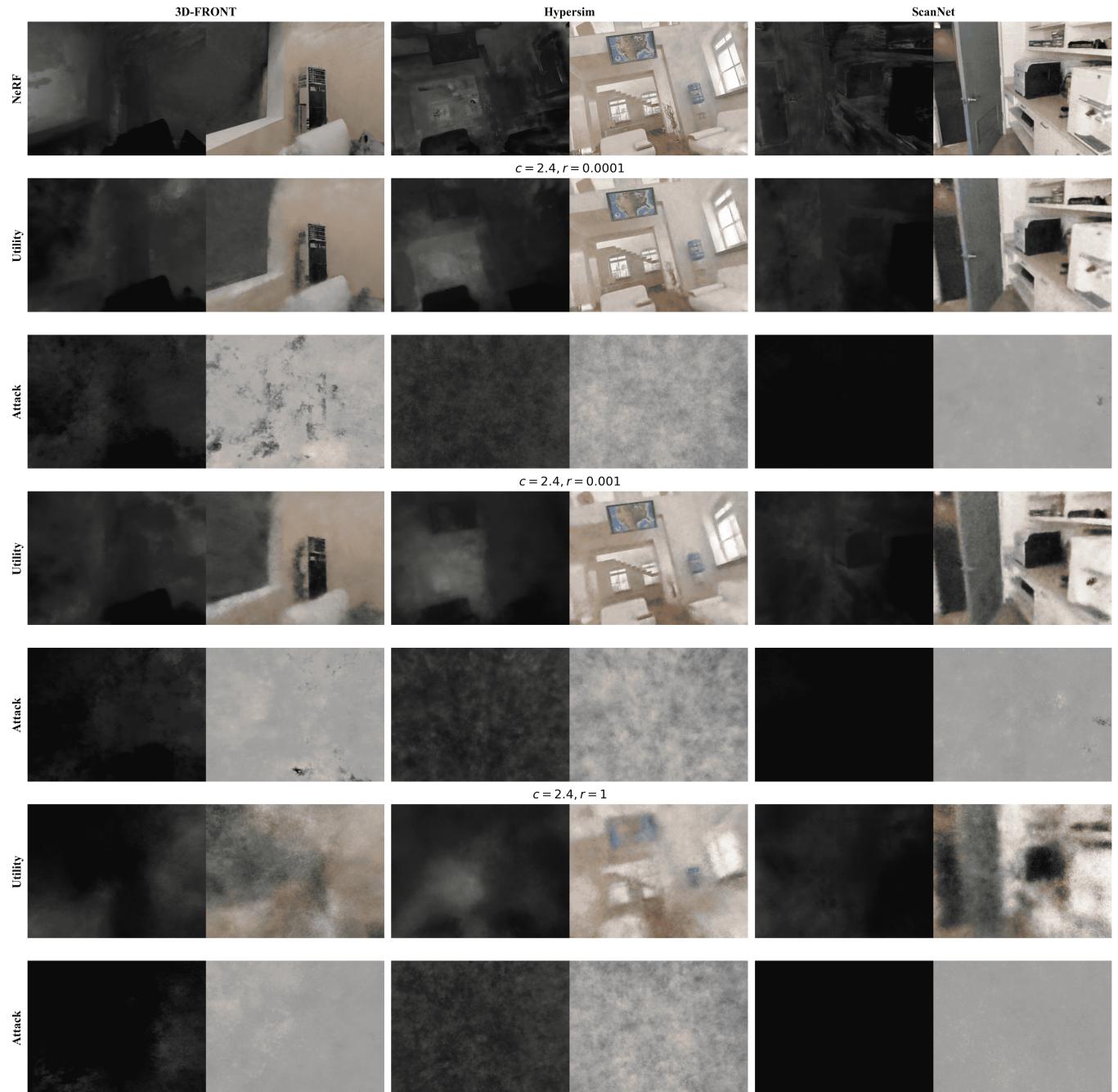


Figure 14: Experimental $S^2\text{NeRF}$ Results with the configuration $c = 2.4, r = 0.0001, 0.001, 1$.



Figure 15: Experimental S^2 NeRF Results with the configuration $c = 4.8, r = 0.0001, 0.001, 1$.



Figure 16: NoisyLabelNeRF Results with $\sigma_l = 0.5, 1, 2$.

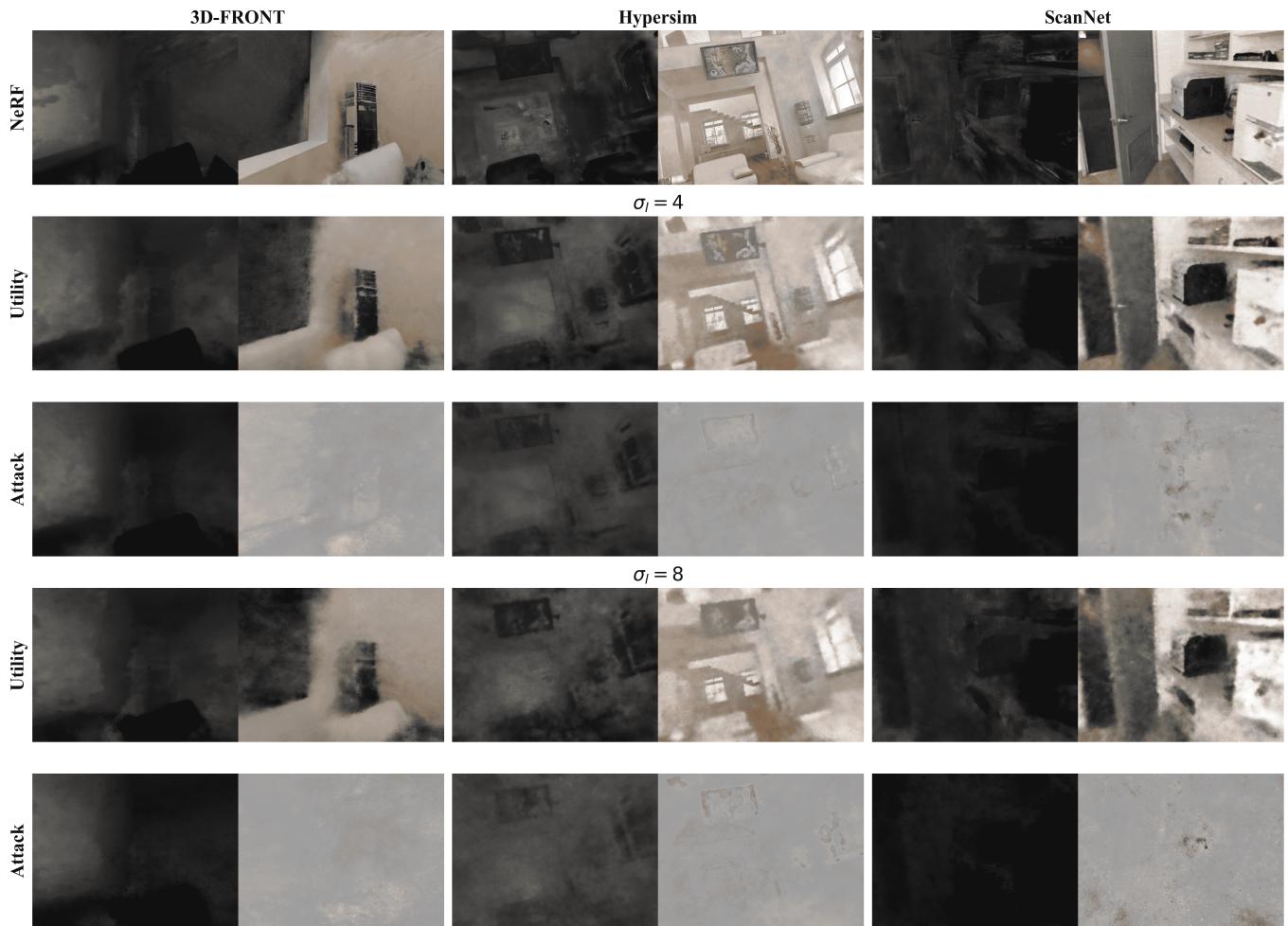


Figure 17: NoisyLabelNeRF Results with $\sigma_l = 4, 8$.