FedMeNF: Privacy-Preserving Federated Meta-Learning for Neural Fields

Supplementary Material

A. Notations

Notation	Description
N	Number of clients
M	Number of participating clients in each communication round
R	Number of communication rounds
E	Number of outer loop steps
K	Number of inner loop steps
α^m	Weight proportional to the client m's dataset size (see Eq. (2))
λ_i	Inner loop learning rate
λ_o	Outer loop learning rate
T	Data within a task, *i.e.*, an image / a video / images and camera poses of a 3D object.
S	Support set of the task data
Q	Query set of the task data
B	Minibatch of the data (B_K is sampled from the query set Q, while the rest is sampled from the support set S)
θ	Global model (meta-learner) parameters
w	Local model (meta-learner) parameters
φ	Neural field parameters
D_{train}	Client's local training dataset
γ	Regularization coefficient

Table 5. Notations.

B. Related Work

B.1. Federated Meta-Learning

Traditional federated learning algorithms assume that decentralized local datasets all belong to the same task and aim to train a single global model. However, when data is heterogeneous, or clients require different tasks, a single global model often fails to perform well for all clients.

To address this challenge, federated meta-learning introduces the meta-learning approach to achieve personalization. Federated meta-learning combines federated learning and meta-learning to train a global meta-learner using data from multiple clients. For example, [8] proposed a method that combines the traditional FedAvg [41] algorithm with MAML [19] and Meta-SGD [39]. Similarly, [18] introduced an approach that integrates FedAvg [41] with HF-MAML [17].

In our work, we constructed baselines by combining not only FedAvg [41] but also state-of-the-art federated learning algorithms such as FedProx [38], Scaffold [30], FedNova [61], FedExP [28], FedACG [32] with meta-learning algorithms like MAML [19], FOMAML [19], Reptile [44], and meta-NSGD [69]. We then evaluated the performance of our FedMeNF against these baselines.

B.2. Meta-Learning for Neural Fields

Neural fields typically require training a separate neural network for each task (signal) and demand large amounts of data and computation. Additionally, the training process often converges slowly, which is a significant limitation [22, 58]. To address these challenges, recent research has focused on learning a generalized neural field that can handle multiple tasks efficiently.

For example, [58] used MAML [19] and Reptile [44] to train a meta-initialized neural field network. Instead of training neural fields for each task from scratch, they optimize the neural fields for each task from a meta-initialized neural field. This approach accelerates neural field optimization and achieves high performance even in few-shot scenarios. Inspired by this, we include MAML [19], FOMAML [19], and Reptile [44] as baselines and evaluate our method against them. [14] extended the work of [58] by introducing task-specific modulation vectors. During the inner loop, only the modulation vector is updated, while in the outer loop, the base network is updated separately from the modulation vector. This method reduces memory requirements by storing a modulation vector for each task with a single base network, rather than all neural field parameters. While our work focuses on achieving fast optimization and high performance with limited data, we did not include modulation-based meta-learning methods as baselines because they do not align directly with our objectives.

Recent research has also explored hypernetwork-based approaches for training generalized neural fields. [9, 31] employed transformers as hypernetworks to predict neural field parameters from task data, while [22] extended this approach using neural processes. These hypernetwork-based methods have shown superior generalization compared to gradient-based meta-learning approaches like [58]. However, they come with the drawback of requiring much larger models. Such large models are impractical in a federated learning setting where models are frequently communicated between the server and clients at every communication round. For this reason, we excluded hypernetwork-based methods from our baselines. In future work, we plan to explore ways to adapt these high-performing hypernetwork-based methods to federated learning environments, making them more efficient and scalable.

B.3. Federated Learning for Neural Fields

Federated learning methods for neural fields, such as those proposed in [24, 54, 55, 68], train a global neural field using local data from multiple clients without sharing the raw data. These methods aim to train a global neural field for a single scene collaboratively. This approach differs from ours, which focuses on training a global meta-learner that can quickly adapt to diverse tasks with minimal data.

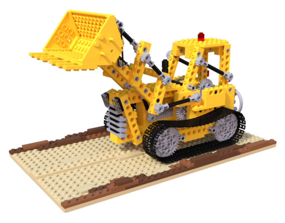
In this scenario, all clients have a subset of the same task data. For instance, if multiple clients have images of the same car, these methods aim to train a global neural field for that car using a federated learning approach. The trained global neural field is shared with all clients, including the server, and can be used to render images from new viewpoints. However, if the server or other clients render images using the same camera poses as those in a client's private dataset, they can reconstruct images that are very similar to the client's private images. This violates the core principle of data privacy, which is central to federated learning.

We experimentally demonstrate this privacy leakage. Using FedNeRF [24], we trained a global neural field for the Lego scene [43] with varying numbers of total clients: 5, 10, and 50. Our experimental setup follows [24] with two key differences: only five clients participated in training per round, and we assume the server has no access to any data and does not pre-train an initial NeRF. The dataset includes 100 training images with a resolution of 400 $\times$ 400 evenly distributed among clients and 200 test images.

As shown in Tab. 6 and Figs. 8 and 9, the trained global neural field can render images that are highly similar to the private images held by clients. These results show that we cannot train a global neural field for a single scene in a federated learning setup, which inherently carries a risk of privacy leakage. This approach should only be used with non-private data, such as public scenes.

Dataset	Lego [43]					
# of clients	PSNR$_p$($\downarrow$)	PSNR($\uparrow$)	SSIM$_p$($\downarrow$)	SSIM($\uparrow$)	LPIPS$_p$($\uparrow$)	LPIPS($\downarrow$)
5	24.00	**23.59**	0.829	**0.827**	0.173	**0.172**
10	23.66	23.40	0.823	0.823	0.177	0.176
50	**22.48**	22.03	**0.808**	0.804	**0.188**	0.191

Table 6. Results of various reconstruction quality metrics (PSNR, SSIM, LPIPS) and privacy metrics (PSNR$_p$, SSIM$_p$, LPIPS$_p$) on the Lego [43] dataset.


(a) Client's training image (PSNR$_p$)

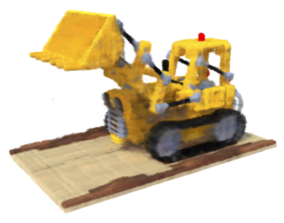
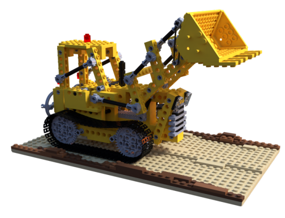

(b) Server-reconstructed image (23.46)

Figure 8. Qualitative results of reconstructing a client's private image on the server (# of clients = 5). A significant privacy leak arises as the server can reconstruct images that are highly similar to the client's original images.


(a) GT image of novel view (PSNR)

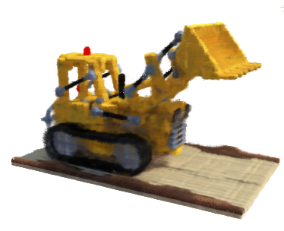

(b) Rendered image (23.53)

Figure 9. Qualitative results of novel view synthesis performed on the client with the trained global neural field (# of clients = 5).

C. Privacy Attack Scenarios

C.1. Membership Inference Attack

The objective of the Membership Inference Attack (MIA) [50] is for the server to determine whether a given data sample was in the training set of a target client's local meta-learner. Therefore, a successful attack would allow the server to identify the owner of a car in the data sample.

Setting. Our attacker model is implemented as a simple convolutional network consisting of two convolution layers and two fully connected layers. This model takes a pair of images as input: a target image and a synthesized image. The synthesized image is rendered by a specific local meta-learner using the same view as the target image. The attacker's task is to infer whether the target image belongs to the training set of the meta-learner used for synthesis. The attacker model outputs a binary prediction indicating whether the target image was included in the training set of the local meta-learner used to generate the corresponding synthesized image.

Dataset. To train and evaluate the attacker, we partition the 50 clients into a group of 40 shadow clients and a held-out group of 10 target clients. To construct the dataset for training and evaluating our attacker model, we partition the 50 clients into 40 shadow clients and 10 target clients. Each client holds a training set of four images of a unique car, totaling 200 unique target images (50 clients $\times$ 4 images/client). The 160 images from the shadow clients are used to generate the attacker's training set, while the 40 images from the target clients form the test set. For each target image, we generate a positive sample (membership label =

1) by pairing it with the image synthesized by its owner's local meta-learner. A negative sample (membership label = 0) is created by pairing the same target image with an image synthesized by a randomly selected client's local meta-learner within the same partition. This process results in a balanced dataset for the attacker model, comprising 320 training samples and 80 test samples.

Evaluation. We assess the attacker model's accuracy in correctly predicting the membership labels. Higher accuracy signifies a more effective attack and, consequently, a more significant privacy vulnerability.

C.2. Property Inference Attack

The objective of the Property Inference Attack (PIA) [20] is for the server to infer a global property of a local meta-learner's training set. Specifically, a successful attack allows the server to determine the vehicle category (e.g., Sedan, SUV, or Coupe) of the car that each client owns.

Setting. The attacker model for the PIA is a classifier, implemented with a simple convolutional network architecture similar to that for MIA. It is trained to take a synthesized image from an arbitrary client's meta-learner as input and predict the vehicle category of that client's car.

Dataset. Similar to the MIA setup, we construct the PIA dataset from the 200 unique target images provided by all clients. For each target image, we use its owner client's local meta-learner to synthesize a new image from the same viewpoint. The synthesized image is then labeled with the vehicle category of the client's car, constructing a single data sample for the attacker model. As a result, we obtain a total of 200 samples: 160 samples derived from the shadow clients, which form the training set, and 40 samples from the target clients, which form the test set. The distributions of the training and test sets are shown in Fig. 10.

Evaluation. We evaluate the attacker model's classification accuracy on the vehicle categories of the target clients' cars. High accuracy implies that the local meta-learner leaks information about the private properties (i.e., vehicle category) of a client's training data.

D. Implementation Details

Models. To maintain consistency with previous research [9, 14, 56, 58], we employ the widely adopted SIREN [51] architecture for image and video reconstruction and the simplified NeRF model [43] for novel view synthesis. The SIREN model consists of 6 layers with a hidden dimension of 128. The NeRF model consists of 6 layers with a hidden dimension of 256. We apply gradient clipping to the gradients (max norm value = 5). All experiments are run on a cluster of 64 NVIDIA TITAN RTX GPUs. Our code will be publicly available upon publication.

Hyperparameters. Table 7 summarizes the hyperparameters used in our experiments.

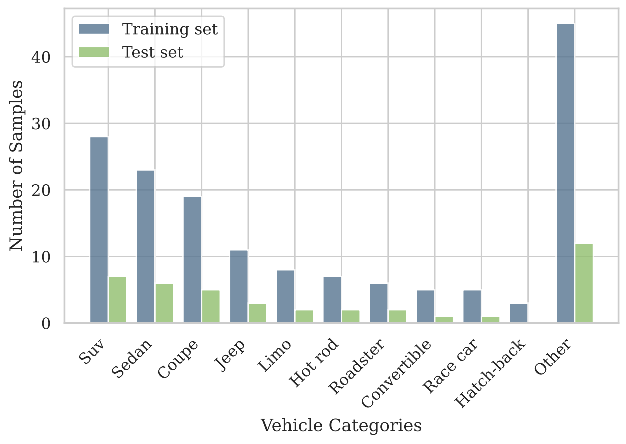


Figure 10. Distribution of the training and test sets for the Privacy Inversion Attack.

E. Datasets

We conduct experiments across the datasets of various modalities, including images, videos, and NeRF. To evaluate performance with limited local training data per client, we select scenarios in which each client has only one or very few tasks. For images, we use the cat category from the PetFace dataset [49], assuming each client has an average of 3.12 images of a unique cat instance. For videos, we use the GolfDB dataset [42], where each client has an average of 1.56 videos of only one person's golf swings. Since a task is defined as reconstructing a single image or video, each image or video corresponds to one task. Note that a client has images or videos of an individual, and the number of images or videos determines the number of tasks.

However, the definition of a task is slightly different for NeRF scenes. A task is defined as synthesizing a novel view of a 3D object, where the input views for test-time optimization (support set) and the new views for testing (query set) constitute one task. We use the Cars category from the ShapeNet dataset [6]. The training set for each client contains on average four input views of a single car with an equivalent amount allocated for testing. We also test the FaceScape dataset [65, 70] for human faces. Each client has an average of 10 training input views for a single facial expression and the same number of views for testing. Each client has one task (one 3D object) for NeRF scenes, and the number of input views follows a Dirichlet distribution.

E.1. Petface

We use the PetFace [49] dataset that contains animal face images of 257484 unique individuals across 13 animal families and 319 breed categories. In this work, we focus exclusively on the cat category within the dataset. We assume each client represents an individual pet owner and possesses images of a unique pet. The total number of training images across all clients is 156, and the total number of test images is 51. On average, each client has 3.12 training images and 1.02 test images. All images have a resolution of 224 × 224. The number of images available for each pet (client) varies, as shown in Fig. 11. The client's local training dataset is used in a federated meta-learning framework to collaboratively train a global meta-learner. Once the global meta-learner is trained, each client can use it to rapidly optimize the neural fields of new images, whether of their own pet, another client's pet, or out-of-distribution (OOD) pets.

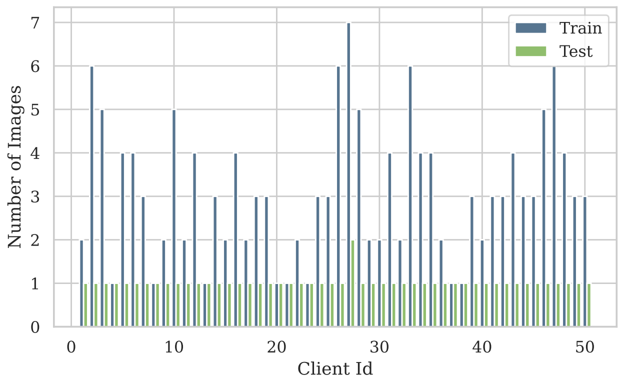


Figure 11. Distribution of the number of images for each client in the PetFace [49] dataset.

E.2. GolfDB

GolfDB [42] is a video dataset that includes 1400 golf swing videos of professional golfers. Each video is categorized into one of three view types: face-on, down-the-line, or other. We use only the down-the-line videos with a resolution of 160 x 160 at 30 fps. We assume that each client represents an individual golfer and possesses only their own golf swing videos. Across all clients, the total number of training videos is 78, and the total number of test videos is 52. On average, each client has 1.56 training videos and 1.04 test videos. The number of videos available for each client is shown in Fig. 12. Each client can use the trained global meta-learner to rapidly optimize the neural field for a new golf swing video, regardless of who performed the swing.

E.3. ShapeNet

We conducted experiments on ShapeNet [6], which is broadly used for Neural Radiance Fields (NeRF). NeRF [43] is a method for rendering novel views of a 3D scene by predicting the color and density at the arbitrary 3D location and 2-dimensional viewing directions through volume rendering. This approach enables the synthesis of images from new viewpoints, a task referred to as novel view

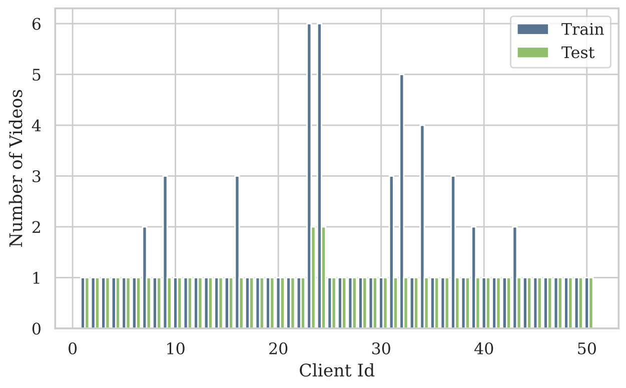
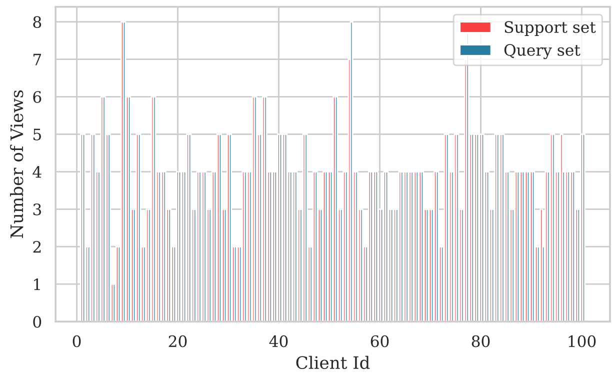


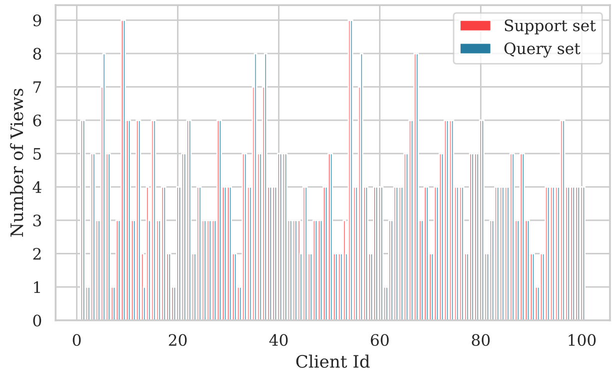
Figure 12. Distribution of the number of videos for each client in the GolfDB [42] dataset.

synthesis. In our experiments, we focused on private and personal objects that people typically possess in limited varieties, such as cars. From ShapeNet, we randomly selected a subset of 100 car objects, each represented by 50 images of 128×128 resolution along with their corresponding camera poses. The 100 car objects are allocated to 50 clients, with each client assigned a total of 2 car objects; one for the local training dataset to train a global meta-learner, and one for the local test dataset to evaluate novel view synthesis performance.

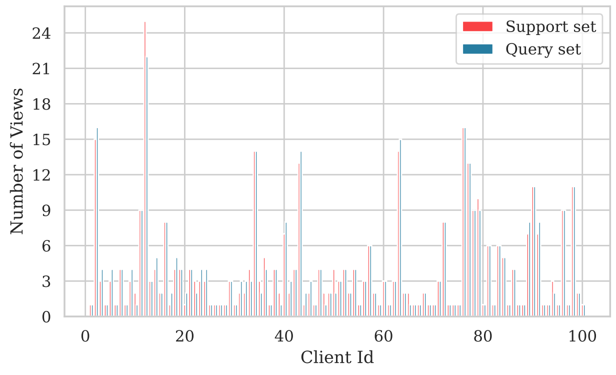
To mimic real-world scenarios where the number of available views per object is limited and varies, we restricted the input views for each car object using a Dirichlet distribution. Each car's data is divided into two parts. The support set is used for inner-loop optimization during training and test-time optimization during testing. The query set is used for outer-loop optimization during training and to evaluate the performance of novel view synthesis during testing. On average, each car object has four support images and 4 query images. Across all 100 car objects, this results in 400 support images distributed among clients. Using the Dirichlet distribution [7, 64], we redistributed the number of support images to simulate varying levels of heterogeneity, ensuring the size of the query set is proportional to the support set. To evaluate the generalization ability of our FedMeNF in non-identically distributed (non-IID) settings, we conduct experiments with different Dirichlet parameters ($\alpha = 10, 5.0, 1.0$). As α decreases, the distribution of support and query set sizes among clients becomes more heterogeneous, as illustrated in Fig. 13. Additionally, we test scenarios where the average number of support and query images per car is not fixed at 4. For example, we experimented with average sizes of 2 views and 8 views per car to examine FedMeNF's performance in different scenarios. The distributions of support and query set sizes for these settings are shown in Fig. 14. This comprehensive experimental setup allows us to evaluate FedMeNF's performance and robustness across a wide range of realistic, heterogeneous, and resource-constrained scenarios. The trained global meta-learner enables the rapid optimization of a neural field for any new car object even with a few images, allowing for the synthesis of high-quality novel views of that car.



(a) $\alpha = 10$

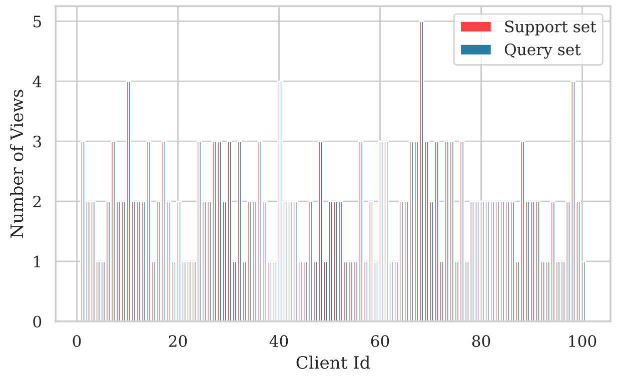


(b) $\alpha = 5.0$

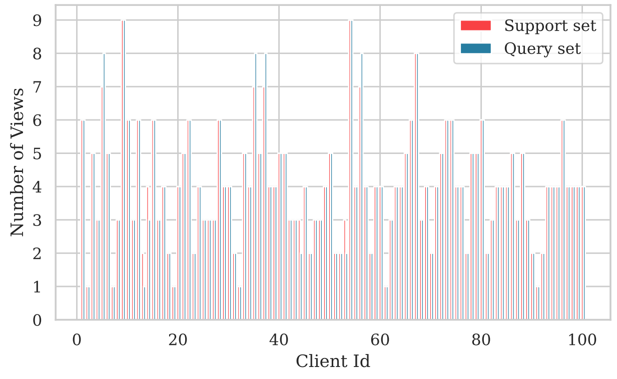


(c) $\alpha = 1.0$

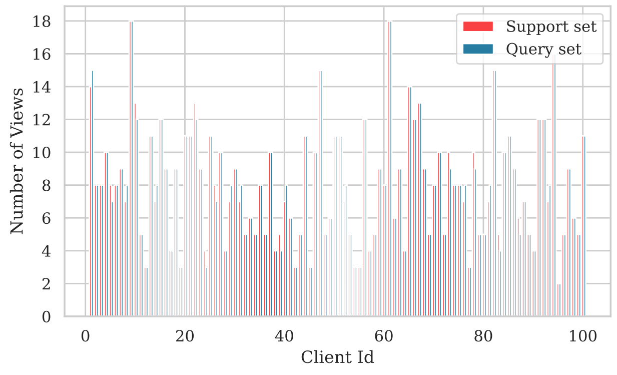
Figure 13. Distribution of the number of views for each car in the Cars [6] dataset under various Dirichlet parameters.



(a) Avg. number of views = 2



(b) Avg. number of views = 4



(c) Avg. number of views = 8

Figure 14. Distribution of the number of views for each car in the Cars [6] dataset under various avg. number of views ($\alpha = 5.0$).

E.4. FaceScape

We extended our evaluation to human faces, a critical area for privacy-preserving applications, using the FaceScape [65, 70] dataset, a more private and unique data type. This experiment aims to test the ability of our global meta-learner to handle diverse and privacy-sensitive datasets. The FaceScape dataset contains 3D facial data from 847 subjects, each with 20 expressions. These expressions include emotional states, such as neutral, smile, and anger, and dynamic facial movements, such as blinking or raising eyebrows. There are 120 images for each expression with a resolution of 512 x 512 and corresponding camera poses.

In our work, we randomly select 50 subjects from the dataset and treat each subject as a separate client. Each client is assigned data for two expressions, averaging 20 images per expression at a downsampled resolution of 128 x 128. Data for one expression is used for training, while the other, considered an unseen expression, is employed for testing.

For each expression, we assume an average of 10 support images used for inner-loop optimization during training or test-time optimization during testing. The number of support images for each expression, a total of 100 (clients) $\times$ 10 (images/client) $=$ 1000 images across all expressions, is reallocated using a Dirichlet distribution [7, 64]. Similarly, we assign an average of 10 query images per expression for outer-loop optimization during training or novel view synthesis evaluation during testing, maintaining the same ratio as the number of support images. The support and query images distribution across expressions is shown in Fig. 15.

Using the trained global meta-learner, each client can quickly optimize a neural field for a new expression or even an entirely new person with only a few input images. This allows for synthesizing novel views of the face with the desired expression. This capability has broad applications: in AR/VR, it can enable the creation of realistic avatars that dynamically mimic user expressions. In 3D animation, it simplifies the production of dynamic facial models. Personal digital assistants can also use this approach to deliver more engaging and personalized facial interactions.

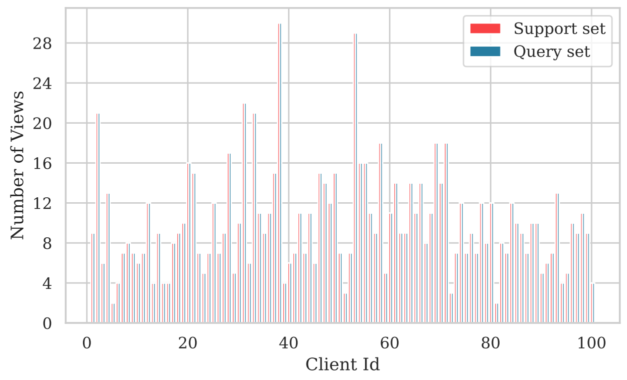


Figure 15. Distribution of the number of views for each expression in the FaceScape [7, 64] dataset ($\alpha = 5.0$).

F. Additional Metrics

We used not only PSNR but also SSIM and LPIPS as metrics to evaluate the performance of reconstruction or novel view synthesis for the trained neural field. Similarly, we extended the privacy metric, PSNR$_p$, defined in Sec. 4, to SSIM$_p$ and LPIPS$_p$. These metrics assess how much information about the client's private dataset is included in the global meta-learner. The privacy metric measures the similarity between data inferred by the meta-learner and the client's local data. The closer the similarity, the greater the privacy leakage, as it indicates the meta-learner has retained more specific information about the client's private data. Expanded results corresponding to Tab. 1 are presented in Tabs. 8 to 11.

G. Convergence

Figure 16 shows the convergence behavior of MAML [19] and our FedMeNF ($\gamma = 0.75$) with FedAvg [41] on the Cars [6] dataset. The inner loss represents the MSE loss during the inner loop, as defined in Eq. (8), while the outer loss corresponds to the MSE loss during the outer loop, as described in Eq. (9), for each round.

For the inner loss, MAML converges to a lower value compared to FedMeNF. This happens because the global meta-learner in MAML, used as the initialization for neural field parameters, is closer to the client's data.

On the other hand, the outer loss of FedMeNF is generally lower than that of MAML. This is because, according to Eqs. (9) and (10), FedMeNF's loss is calculated as

$$L(\varphi_K, B_K) - \gamma L(w_i, B_K),$$

which subtracts the global meta-learner's loss on the client's query set $L(w_i, B_K)$ from MAML's loss $L(\varphi_K, B_K)$. However, the outer loss of FedMeNF converges to a similar level to that of MAML.

One might incorrectly interpret this convergence as indicating that $L(w_i, B_K)$ approaches zero. However, this is not the case. The privacy metric PSNR$_p$ for FedMeNF converges to a non-zero value, meaning that $L(w_i, B_K)$ also converges to a non-zero value. Instead, the lower outer loss in FedMeNF implies that $L(\varphi_K, B_K)$ is smaller in FedMeNF compared to MAML. This result demonstrates that FedMeNF achieves a lower loss than MAML while preserving client data privacy. This also explains why FedMeNF's novel view synthesis metrics, such as PSNR, SSIM, and LPIPS, not only outperform MAML but also converge faster.

H. Proof of Proposition 1

Consider the local meta-optimization gradient in Eq. (9) where $\gamma = 0$:

$$g_M = \nabla_{w_i} L(\varphi_K, B_K). \tag{15}$$

Hyperparameter	PetFace [49]	GolfDB [42]	Cars [6]	FaceScape [65, 70]
Number of communication rounds (R)	1000			
Number of clients (N)	50			
Number of participating clients in each communication round (M)	5			
FedProx [38] μ	0.1	0.1	0.1 (Reptile: 0.0001)	
FedExP [28] ϵ	0.001			
FedACG [32] λ	0.2			
FedACG [32] β	0.1	0.1	0.1 (Reptile: 0.0001)	
meta-NSGD [69] ϵ	10.0			
meta-NSGD [69] λ	0.1			
Number of outer steps (E)	32	64	8	40
Number of inner steps (K)	1	1	8	8
Outer learning rate (λ_o)	0.01 (FOMAML: 0.05)	0.05	0.05 (Reptile:10)	
Inner learning rate (λ_i)	0.005	0.01	0.001	0.0001
Optimizer	SGD	SGD	AdamW	AdamW
Test-time optimization (TTO) steps	64	8192	8192	32768
Batch size	1024	2048	128 rays $\times$ 128 points	

Table 7. Hyperparameters.

Then, the first-order approximation of the meta-gradient g_M is

$$g_M \approx g_K - \lambda_i \mathcal{I}_K, \text{ where } \mathcal{I}_K = \sum_{j=0}^{K-1} \nabla_{\varphi_0} \langle g_K, g_j \rangle . \tag{16}$$

Proof. We use the following definitions:

$$g_i = \nabla_{\varphi_0} L(\varphi_0, B_i) \tag{17}$$
$$g_K = \nabla_{\varphi_0} L(\varphi_0, B_K) \tag{18}$$
$$H_i = \nabla^2_{\varphi_0} L(\varphi_0, B_i) \tag{19}$$
$$\hat{g}_i = \nabla_{\varphi_i} L(\varphi_i, B_i) \tag{20}$$
$$\hat{g}_K = \nabla_{\varphi_K} L(\varphi_K, B_K) \tag{21}$$
$$\hat{H}_i = \nabla^2_{\varphi_i} L(\varphi_i, B_i) \tag{22}$$

According to Eq. (8),

$$\varphi_{j+1} = \varphi_j - \lambda_i \nabla_{\varphi_j} L(\varphi_j, B_j). \tag{23}$$

Then, we have

$$\varphi_j = \varphi_0 - \lambda_i \sum_{l=0}^{j-1} \nabla_{\varphi_l} L(\varphi_l, B_l), \tag{24}$$

$$\varphi_j - \varphi_0 = -\lambda_i \sum_{l=0}^{j-1} \nabla_{\varphi_l} L(\varphi_l, B_l) \tag{25}$$
$$= -\lambda_i \sum_{l=0}^{j-1} \hat{g}_l. \tag{26}$$

We assume $L(\varphi_j, B_j)$ is differentiable three times at φ_0 to apply Taylor theorem. By Taylor's theorem, we have

$$\begin{aligned} \hat{g}_j =& \nabla_{\varphi_j} L(\varphi_j, B_j) \quad (27)\\ =& \nabla_{\varphi_0} L(\varphi_0, B_j) \\ &+ \nabla^2_{\varphi_0} L(\varphi_0, B_j) \cdot (\varphi_j - \varphi_0) \\ &+ \frac{1}{2!} \nabla^3_{\varphi_0} L(\varphi_0, B_j) \cdot (\varphi_j - \varphi_0)^2 \\ &+ \cdots . \quad (28) \end{aligned}$$

Combining Eq. (28) and Eq. (26),

$$\begin{aligned} \hat{g}_j =& \nabla_{\varphi_0} L(\varphi_0, B_j) \\ &+ \nabla^2_{\varphi_0} L(\varphi_0, B_j) \cdot (\varphi_j - \varphi_0) \\ &+ O(\lambda_i^2) \quad (29)\\ =& g_j - \lambda_i H_j \sum_{l=0}^{j-1} \hat{g}_l + O(\lambda_i^2). \quad (30) \end{aligned}$$

We assume a learning rate λ_i is small enough for first-order approximation. Then, the first-order approximation of the $\hat{g}_j$ is

$$\hat{g}_j \approx g_j - \lambda_i H_j \sum_{l=0}^{j-1} \hat{g}_l. \tag{31}$$

Note that

$$\hat{g}_j \approx g_j + O(\lambda_i). \tag{32}$$

Similarly, we can have

$$\hat{H}_j \approx H_j + O(\lambda_i). \tag{33}$$

Combining Eq. (31) and Eq. (32), we have

$$\hat{g}_j \approx g_j - \lambda_i H_j \sum_{l=0}^{j-1} g_l + O(\lambda_i^2) \tag{34}$$

$$\approx g_j - \lambda_i H_j \sum_{l=0}^{j-1} g_l. \tag{35}$$

Now, let's calculate the meta-gradient g_M as follows.

$$g_M = \nabla_{w_i} L(\varphi_K, B_K) \tag{36}$$

$$= \nabla_{\varphi_0} L(\varphi_K, B_K) \qquad (\varphi_0 = w_i, \text{ ref. Algorithm 1}) \tag{37}$$

$$= \prod_{i=0}^{K-1} \nabla_{\varphi_i} \varphi_{i+1} \cdot \nabla_{\varphi_K} L(\varphi_K, B_K) \tag{38}$$

$$= \prod_{i=0}^{K-1} \nabla_{\varphi_i} (\varphi_i - \lambda_i \nabla_{\varphi_i} L(\varphi_i, B_i)) \cdot \nabla_{\varphi_K} L(\varphi_K, B_K) \tag{39}$$

$$= \prod_{i=0}^{K-1} (I - \lambda_i \nabla^2_{\varphi_i} L(\varphi_i, B_i)) \cdot \nabla_{\varphi_K} L(\varphi_K, B_K) \tag{40}$$

$$= \prod_{i=0}^{K-1} (I - \lambda_i \hat{H}_i) \cdot \hat{g}_K \tag{41}$$

$$= (I - \lambda_i \sum_{i=0}^{K-1} \hat{H}_i) \cdot \hat{g}_K + O(\lambda_i^2) \tag{42}$$

$$= (I - \lambda_i \sum_{i=0}^{K-1} H_i) \cdot \hat{g}_K + O(\lambda_i^2) \qquad (\text{using Eq. (33)}) \tag{43}$$

$$= (I - \lambda_i \sum_{i=0}^{K-1} H_i) \cdot (g_K - \lambda_i H_K \sum_{i=0}^{K-1} g_i) + O(\lambda_i^2) \qquad (\text{using Eq. (35)}) \tag{44}$$

$$= g_K - \lambda_i \sum_{i=0}^{K-1} (H_K g_i + g_K H_i) + O(\lambda_i^2) \tag{45}$$

$$= g_K - \lambda_i \sum_{i=0}^{K-1} \nabla_{\varphi_0} \langle g_K, g_i \rangle + O(\lambda_i^2) \tag{46}$$

Therefore, the first-order approximation of the g_M is

$$g_M \approx g_K - \lambda_i \sum_{i=0}^{K-1} \nabla_{\varphi_0} \langle g_K, g_i \rangle, \tag{47}$$

$$= g_K - \lambda_i \mathcal{I}_K. \tag{48}$$

□

I. Proof of Proposition 2

Let $\Delta L_{i+1} = L(w_{i+1}, B_K) - L(w_i, B_K)$. Then, the first-order approximation of ΔL_{i+1} is

$$\Delta L_{i+1} \approx -\lambda_o (\nabla_{w_i} L(w_i, B_K))^2 = -\lambda_o \cdot {g_K}^2 \leq 0. \tag{49}$$

Proof. According to Eq. (9) where $\gamma = 0$,

$$w_{i+1} = w_i - \lambda_o \nabla_{w_i} L(\varphi_K, B_K), \tag{50}$$

$$w_{i+1} - w_i = -\lambda_o \nabla_{w_i} L(\varphi_K, B_K). \tag{51}$$

By Taylor's theorem, we have

$$L(w_{i+1}, B_K) = L(w_i, B_K) + L'(w_i, B_K) \cdot (w_{i+1} - w_i) + \frac{1}{2!} L''(w_i, B_K) \cdot (w_{i+1} - w_i)^2 + \cdots. \tag{52}$$

$$= L(w_i, B_K) + L'(w_i, B_K) \cdot (-\lambda_o \nabla_{w_i} L(\varphi_K, B_K)) + O(\lambda_o^2) \qquad (\text{using Eq. (51)}). \tag{53}$$

Since $\varphi_0 = w_i$ according to Algorithm 1,

$$L'(w_i, B_K) = L'(\varphi_0, B_K). \tag{54}$$

Then, using Eq. (18),

$$L'(w_i, B_K) = g_K. \tag{55}$$

Combining Eq. (53), Eq. (55), and Proposition 1

$$\Delta L_{i+1} = L(w_{i+1}, B_K) - L(w_i, B_K) \tag{56}$$

$$= g_K \cdot (-\lambda_o \nabla_{w_i} L(\varphi_K, B_K)) + O(\lambda_o^2) \tag{57}$$

$$\approx -\lambda_o \cdot g_K \cdot (g_K - \lambda_i \mathcal{I}_K) + O(\lambda_o^2) \tag{58}$$

$$= -\lambda_o \cdot {g_K}^2 + O(\lambda_i \lambda_o) + O(\lambda_o^2). \tag{59}$$

Therefore, the first-order approximation of the ΔL_{i+1} is

$$\Delta L_{i+1} \approx -\lambda_o \cdot {g_K}^2 \leq 0. \tag{60}$$

□

J. Proof of Proposition 3

Consider the gradient of privacy-preserving loss L_{pp}:

$$g_{pp} = \nabla_{w_i} L_{pp}(\gamma, w_i, \varphi_K, B_K) \tag{61}$$

$$= \nabla_{w_i} (L(\varphi_K, B_K) - \gamma L(w_i, B_K)). \tag{62}$$

Then, the first-order approximation of g_{pp} and ΔL_{i+1} are

$$g_{pp} \approx (1 - \gamma) \cdot g_K - \lambda_i \mathcal{I}_K, \tag{63}$$

$$\Delta L_{i+1} \approx -\lambda_o (1 - \gamma)(g_K)^2 \leq 0. \tag{64}$$

Proof. Using Proposition 1 and Eq. (18), we have

$$\begin{aligned} g_{pp} =& \nabla_{w_i}(L(\varphi_K, B_K) - \gamma L(w_i, B_K)) \quad (65)\\ \approx& g_M - \gamma g_K \quad (66)\\ =& (1-\gamma)\cdot g_K - \lambda_i \mathcal{I}_K. \quad (67) \end{aligned}$$

According to Eq. (9) and using Eq. (67),

$$w_{i+1} = w_i - \lambda_o \nabla_{w_i}(L(\varphi_K, B_K) - \gamma L(w_i, B_K)), \quad (68)$$

$$\begin{aligned} w_{i+1} - w_i =& -\lambda_o \nabla_{w_i}(L(\varphi_K, B_K) - \gamma L(w_i, B_K)) \quad (69)\\ \approx& -\lambda_o((1-\gamma)\cdot g_K - \lambda_i \mathcal{I}_K). \quad (70) \end{aligned}$$

Combining Eq. (52) and Eq. (70), we have

$$\begin{aligned} L(w_{i+1}, B_K) =& L(w_i, B_K)\\ &+ L'(w_i, B_K)\cdot(w_{i+1} - w_i)\\ &+ O(\lambda_o^2) \quad (71)\\ \approx& L(w_i, B_K)\\ &+ L'(w_i, B_K)\cdot(-\lambda_o((1-\gamma)\cdot g_K - \lambda_i \mathcal{I}_K))\\ &+ O(\lambda_o^2). \quad (72) \end{aligned}$$

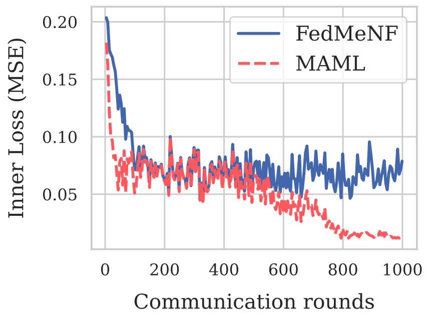
Combining Eq. (72), Eq. (55), and Proposition 1

$$\begin{aligned} \Delta L_{i+1} =& L(w_{i+1}, B_K) - L(w_i, B_K) \quad (73)\\ \approx& g_K\cdot(-\lambda_o((1-\gamma)\cdot g_K - \lambda_i \mathcal{I}_K)) + O(\lambda_o^2) \quad (74)\\ =& -\lambda_o(1-\gamma)\cdot {g_K}^2 + O(\lambda_i\lambda_o) + O(\lambda_o^2). \quad (75) \end{aligned}$$

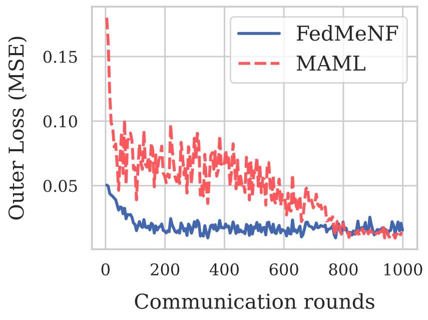
Therefore, the first-order approximation of the ΔL_{i+1} is

$$\Delta L_{i+1} \approx -\lambda_o(1-\gamma){g_K}^2 \leq 0. \quad (76)$$

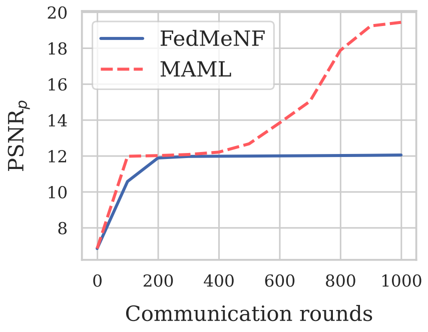
□



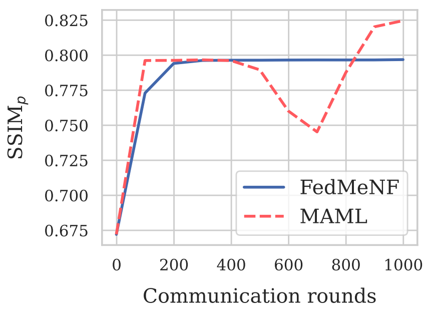
(a) Inner loss



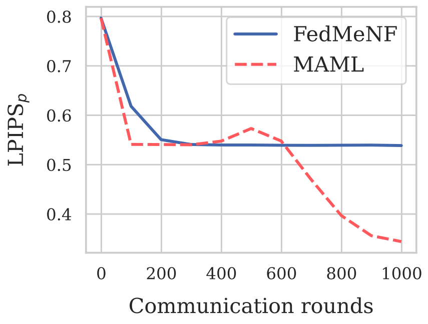
(b) Outer loss



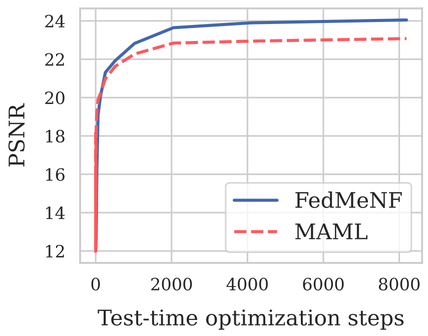
(c) PSNR_p



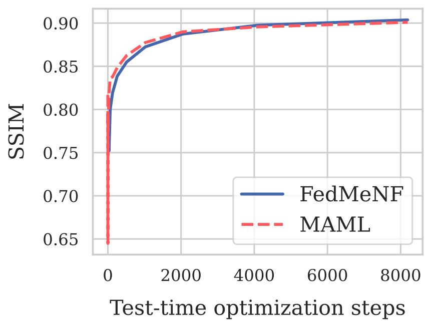
(d) SSIM_p



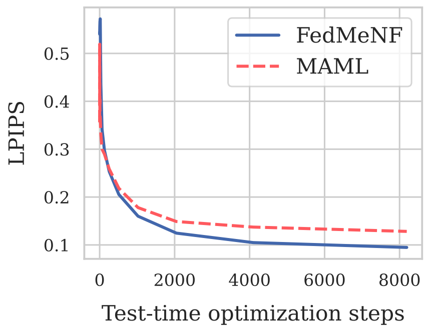
(e) LPIPS_p



(f) PSNR



(g) SSIM



(h) LPIPS

Figure 16. Convergence of MAML [19] and our FedMeNF ($\gamma = 0.75$) with FedAvg [41] on the Cars dataset [6].

Modality		Image					
Dataset		PetFace [49]					
Method \ Metric		PSNR$_p$(↓)	PSNR(↑)	SSIM$_p$(↓)	SSIM(↑)	LPIPS$_p$(↑)	LPIPS(↓)
Local		-	22.29	-	0.572	-	0.716
FedAvg [41]	+ MAML	16.57	27.39	0.544	0.734	0.534	0.419
	+ FOMAML	18.52	23.15	0.573	0.618	0.421	0.415
	+ Reptile	17.39	22.52	0.505	0.585	0.731	0.614
	+ meta-NSGD	12.49	5.15	0.395	0.003	0.923	1.566
	+ **Ours**	14.77	27	0.526	0.723	0.606	0.441
FedProx [38]	+ MAML	16.58	27.49	0.545	0.737	0.532	0.415
	+ FOMAML	18.53	24.32	0.578	0.647	0.423	0.4
	+ Reptile	17.37	22.5	0.503	0.582	0.733	0.619
	+ meta-NSGD	12.49	5.15	0.395	0.003	0.923	1.57
	+ **Ours**	14.44	27.08	0.517	0.727	0.613	0.438
Scaffold [30]	+ MAML	16.71	27.66	0.547	0.743	0.53	0.408
	+ FOMAML	18.56	23.98	0.578	0.641	0.421	0.401
	+ Reptile	17.37	22.47	0.503	0.582	0.736	0.619
	+ meta-NSGD	12.49	5.13	0.395	0.003	0.923	1.571
	+ **Ours**	14.93	27.19	0.512	0.729	0.592	0.434
FedNova [61]	+ MAML	16.52	27.51	0.54	0.737	0.548	0.414
	+ FOMAML	18.5	24.1	0.571	0.635	0.432	0.434
	+ Reptile	17.38	22.52	0.505	0.585	0.731	0.615
	+ meta-NSGD	12.49	5.14	0.395	0.003	0.923	1.57
	+ **Ours**	14.94	27.15	0.526	0.728	0.582	0.433
FedExP [28]	+ MAML	16.57	27.39	0.544	0.734	0.534	0.419
	+ FOMAML	18.52	23.15	0.573	0.618	0.421	0.415
	+ Reptile	17.39	22.64	0.505	0.586	0.731	0.614
	+ meta-NSGD	12.49	5.15	0.395	0.003	0.923	1.566
	+ **Ours**	14.65	26.86	0.504	0.72	0.619	0.449
FedACG [32]	+ MAML	16.63	27.5	0.547	0.738	0.526	0.415
	+ FOMAML	18.48	24.04	0.572	0.638	0.43	0.416
	+ Reptile	17.38	22.63	0.505	0.587	0.727	0.612
	+ meta-NSGD	12.49	5.15	0.395	0.003	0.923	1.571
	+ **Ours**	14.71	26.97	0.524	0.723	0.606	0.441

Table 8. Results of various reconstruction quality metrics (PSNR, SSIM, LPIPS) and privacy metrics (PSNR$_p$, SSIM$_p$, LPIPS$_p$) on the PetFace dataset [49].

Modality		Video					
Dataset		GolfDB [42]					
Method \ Metric		PSNR$_p$(↓)	PSNR(↑)	SSIM$_p$(↓)	SSIM(↑)	LPIPS$_p$(↑)	LPIPS(↓)
Local		-	26.92	-	0.796	-	0.27
FedAvg [41]	+ MAML	21.21	29.68	0.657	0.867	0.385	0.131
	+ FOMAML	20.82	28.57	0.634	0.843	0.445	0.166
	+ Reptile	19.89	27.22	0.562	0.811	0.66	0.245
	+ meta-NSGD	10.96	4.85	0.42	0.003	0.936	1.485
	+ **Ours**	17.31	28.89	0.579	0.855	0.53	0.167
FedProx [38]	+ MAML	21	29.35	0.652	0.862	0.406	0.146
	+ FOMAML	21.28	28.79	0.642	0.85	0.434	0.154
	+ Reptile	19.79	27.21	0.559	0.811	0.665	0.242
	+ meta-NSGD	10.96	4.84	0.42	0.003	0.936	1.486
	+ **Ours**	15.68	28.96	0.546	0.856	0.543	0.164
Scaffold [30]	+ MAML	21.23	29.11	0.658	0.857	0.403	0.157
	+ FOMAML	21.27	28.78	0.648	0.849	0.427	0.161
	+ Reptile	20	27.23	0.568	0.812	0.651	0.24
	+ meta-NSGD	10.96	4.84	0.42	0.003	0.936	1.486
	+ **Ours**	16.21	29.23	0.56	0.861	0.528	0.149
FedNova [61]	+ MAML	21.56	29.63	0.669	0.868	0.378	0.127
	+ FOMAML	20.12	28.52	0.616	0.842	0.496	0.156
	+ Reptile	19.92	27.27	0.563	0.814	0.662	0.235
	+ meta-NSGD	10.96	4.84	0.42	0.003	0.936	1.485
	+ **Ours**	15.71	28.98	0.549	0.856	0.517	0.161
FedExP [28]	+ MAML	21.21	29.68	0.657	0.867	0.385	0.131
	+ FOMAML	20.82	28.57	0.634	0.843	0.445	0.166
	+ Reptile	19.89	27.22	0.562	0.811	0.66	0.245
	+ meta-NSGD	10.96	4.85	0.42	0.003	0.936	1.485
	+ **Ours**	15.38	28.1	0.53	0.834	0.54	0.204
FedACG [32]	+ MAML	21.09	29.51	0.657	0.866	0.398	0.132
	+ FOMAML	21.12	28.7	0.637	0.847	0.445	0.167
	+ Reptile	19.83	27.23	0.561	0.812	0.658	0.24
	+ meta-NSGD	10.96	4.84	0.42	0.003	0.936	1.485
	+ **Ours**	15.94	28.99	0.535	0.858	0.584	0.165

Table 9. Results of various reconstruction quality metrics (PSNR, SSIM, LPIPS) and privacy metrics (PSNR$_p$, SSIM$_p$, LPIPS$_p$) on the GolfDB dataset [42].

Modality		3D (NeRF)					
Dataset		Cars [6]					
Method \ Metric		$\text{PSNR}_p(\downarrow)$	PSNR($\uparrow$)	$\text{SSIM}_p(\downarrow)$	SSIM($\uparrow$)	$\text{LPIPS}_p(\uparrow)$	LPIPS($\downarrow$)
Local		-	17.13	-	0.844	-	0.319
FedAvg [41]	+ MAML	19.73	23.08	0.827	0.901	0.343	0.128
	+ FOMAML	19.73	23.66	0.827	0.905	0.343	0.112
	+ Reptile	19.96	21.98	0.839	0.892	0.311	0.154
	+ meta-NSGD	6.85	10.62	0.672	0.767	0.798	0.566
	+ **Ours**	12.15	24.05	0.798	0.904	0.537	0.095
FedProx [38]	+ MAML	19.67	22.94	0.826	0.9	0.346	0.129
	+ FOMAML	19.67	23.73	0.826	0.907	0.346	0.11
	+ Reptile	19.95	22.11	0.839	0.891	0.311	0.155
	+ meta-NSGD	6.85	10.91	0.672	0.779	0.798	0.547
	+ **Ours**	12.14	23.98	0.798	0.902	0.537	0.096
Scaffold [30]	+ MAML	19.07	24.21	0.811	0.911	0.372	0.092
	+ FOMAML	19.07	24.47	0.811	0.912	0.373	0.086
	+ Reptile	19.81	22.6	0.936	0.896	0.322	0.137
	+ meta-NSGD	6.85	10.63	0.672	0.762	0.798	0.559
	+ **Ours**	14.4	24.34	0.798	0.91	0.514	0.086
FedNova [61]	+ MAML	19.72	23.63	0.827	0.907	0.344	0.109
	+ FOMAML	19.72	23.27	0.827	0.903	0.344	0.121
	+ Reptile	19.96	22.7	0.838	0.897	0.311	0.135
	+ meta-NSGD	6.85	10.6	0.672	0.763	0.798	0.561
	+ **Ours**	12.14	24.12	0.798	0.903	0.537	0.096
FedExP [28]	+ MAML	19.81	22.87	0.829	0.899	0.339	0.14
	+ FOMAML	19.81	22.81	0.829	0.899	0.339	0.141
	+ Reptile	20.91	22.03	0.851	0.891	0.267	0.161
	+ meta-NSGD	6.85	10.61	0.672	0.765	0.798	0.571
	+ **Ours**	12.17	24.05	0.798	0.905	0.537	0.093
FedACG [32]	+ MAML	19.9	22	0.832	0.891	0.328	0.155
	+ FOMAML	19.9	21.95	0.832	0.89	0.328	0.155
	+ Reptile	20.13	22.26	0.842	0.894	0.299	0.148
	+ meta-NSGD	6.85	10.59	0.672	0.761	0.798	0.57
	+ **Ours**	10.93	22.45	0.779	0.881	0.601	0.133

Table 10. Results of various reconstruction quality metrics (PSNR, SSIM, LPIPS) and privacy metrics (PSNR_p, SSIM_p, LPIPS_p) on the Cars dataset [6].

Modality		3D (NeRF)					
Dataset		FaceScape [65, 70]					
Method \ Metric		PSNR$_p$(↓)	PSNR(↑)	SSIM$_p$(↓)	SSIM(↑)	LPIPS$_p$(↑)	LPIPS(↓)
Local		-	23.67	-	0.772	-	0.178
FedAvg [41]	+ MAML	21.31	28.59	0.658	0.904	0.421	0.053
	+ FOMAML	21.24	28.65	0.658	0.905	0.416	0.051
	+ Reptile	21.92	28.24	0.684	0.895	0.368	0.056
	+ meta-NSGD	7.72	11.29	0.191	0.461	0.934	0.742
	+ **Ours**	15.16	27.88	0.268	0.894	0.665	0.061
FedProx [38]	+ MAML	21.31	28.59	0.658	0.904	0.421	0.053
	+ FOMAML	21.24	28.65	0.658	0.905	0.416	0.051
	+ Reptile	21.89	28.21	0.683	0.895	0.37	0.056
	+ meta-NSGD	7.72	11.29	0.191	0.461	0.934	0.742
	+ **Ours**	15.16	27.88	0.268	0.894	0.665	0.061
Scaffold [30]	+ MAML	21.09	28.51	0.651	0.902	0.44	0.053
	+ FOMAML	21.01	28.51	0.65	0.903	0.435	0.053
	+ Reptile	21.79	28.14	0.677	0.893	0.384	0.058
	+ meta-NSGD	7.72	11.29	0.191	0.461	0.933	0.741
	+ **Ours**	13.94	27.53	0.195	0.888	0.748	0.065
FedNova [61]	+ MAML	21.37	28.59	0.661	0.903	0.421	0.053
	+ FOMAML	21.3	28.62	0.66	0.905	0.416	0.052
	+ Reptile	21.98	28.17	0.686	0.894	0.369	0.057
	+ meta-NSGD	7.72	11.29	0.191	0.461	0.933	0.743
	+ **Ours**	15.45	27.86	0.282	0.894	0.648	0.061
FedExP [28]	+ MAML	21.33	27.64	0.659	0.886	0.419	0.065
	+ FOMAML	21.26	27.66	0.659	0.887	0.415	0.064
	+ Reptile	21.93	28.27	0.685	0.896	0.367	0.058
	+ meta-NSGD	7.72	11.29	0.191	0.461	0.933	0.742
	+ **Ours**	13.69	26.12	0.191	0.843	0.766	0.103
FedACG [32]	+ MAML	22.28	28.02	0.7	0.897	0.327	0.055
	+ FOMAML	22.24	28.03	0.699	0.9	0.327	0.052
	+ Reptile	22.07	28.32	0.691	0.896	0.356	0.055
	+ meta-NSGD	7.72	11.29	0.191	0.461	0.934	0.742
	+ **Ours**	14.83	27.46	0.365	0.876	0.669	0.074

Table 11. Results of various reconstruction quality metrics (PSNR, SSIM, LPIPS) and privacy metrics (PSNR$_p$, SSIM$_p$, LPIPS$_p$) on the FaceScape dataset [65, 70].