Is Vanilla MLP in Neural Radiance Field Enough for Few-shot View Synthesis?

Supplementary Material

In Sec. 7, we provide a detailed analysis of how mi-MLP works. Sec. 8 introduces more experimental details, including datasets, metrics, implementation details, and methods used for comparison. We also report the results of using dropout to avoid overfitting in Sec. 9 and the extensions of mi-MLP to the task of 3D generation in Sec. 10. The limitations and future works are illustrated in Sec. 11.

7. How mi-MLP Works?

As illustrated in Sec. 4.1.1, to mitigate the overfitting issue that usually occurs in few-shot view synthesis, we incorporate inputs into each layer of the MLP, which is denoted as:

$$\boldsymbol{f}_i = \phi_i(\boldsymbol{f}_{i-1}, \gamma_L(\boldsymbol{x})),\ \boldsymbol{f}_1 = \phi_1(\gamma_L(\boldsymbol{x})), \tag{13}$$

where ϕ_i is the i-th ($i \geq 2$) layer of the MLP, $\boldsymbol{f}_i$ is the corresponding output feature, $\boldsymbol{x}$ is the input 5D coordinate and $\gamma_L(\boldsymbol{x})$ represents the endoded input embeddings obtained by Eq. 2.

Intuitively, during the early stage of training, with a common MLP initialization, Eq. 13 encourages a smaller amplitude of gradient for the shallower layer compared to that for the deeper layer, where the deeper layers (*i.e.*, layers close to the outputs) are updated with large gradients while the shallower layers are updated with extremely small ones. **This implies that the model capacity is restricted at the start of training, which helps prevent the model from memorizing input views and thus avoids overfitting. However, as the number of network parameters remains unchanged, the total capacity of the model is preserved for more detailed rendering during the later stage of training.**

Specifically, assuming $\gamma_L(\boldsymbol{x}) \in \mathbb{R}^{d_1 \times 1}, \boldsymbol{f}_i \in \mathbb{R}^{d_2 \times 1}$, the bias vector and weight matrix of ϕ_i are $\boldsymbol{b}_i \in \mathbb{R}^{d_2 \times 1}$ and $\boldsymbol{w}_i = (\boldsymbol{w}_i^1, \boldsymbol{w}_i^2, \ldots, \boldsymbol{w}_i^{d_2})^T$ respectively, where $\boldsymbol{w}_i^j = (\boldsymbol{w}_i^{j0} \in \mathbb{R}^{1 \times d_1}, \boldsymbol{w}_i^{j1} \in \mathbb{R}^{1 \times d_2})^T$. Thus Eq. 13 is equivalent to

$$\phi_i^j(\gamma_L(\boldsymbol{x})) = \epsilon\{\boldsymbol{w}_i^{j0} \cdot \gamma_L(\boldsymbol{x}) + \boldsymbol{w}_i^{j1} \cdot \phi_{i-1}(\gamma_L(\boldsymbol{x})) + \boldsymbol{b}_i\}, \tag{14}$$

where ϕ_i^j is the j-th element of $\boldsymbol{f}_i$, ϵ denotes the activation fuction whose default setting is ReLU.

Assuming that the loss function is denoted as $\mathcal{L}$, then

$$\|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_i^{j0}}\|_1 = \|\frac{\partial \mathcal{L}}{\partial \phi_i^j} \cdot \frac{\partial \boldsymbol{w}_i^{j0} \cdot \gamma_L(\boldsymbol{x})}{\partial \boldsymbol{w}_i^{j0}}\|_1 = \|\frac{\partial \mathcal{L}}{\partial \phi_i^j}\|_1 \cdot \|\gamma_L(x)\|_1. \tag{15}$$

$$\|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_i^{j1}}\|_1 = \|\frac{\partial \mathcal{L}}{\partial \phi_i^j} \cdot \frac{\partial \boldsymbol{w}_i^{j1} \cdot \phi_{i-1}}{\partial \boldsymbol{w}_i^{j1}}\|_1 = \|\frac{\partial \mathcal{L}}{\partial \phi_i^j}\|_1 \cdot \|\phi_{i-1}\|_1. \tag{16}$$

$$\begin{aligned}\|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_{i-1}^{j0}}\|_1 &= \|\frac{\partial \mathcal{L}}{\partial \phi_i^j} \cdot \frac{\partial \boldsymbol{w}_i^{j1} \cdot \phi_{i-1}}{\partial \phi_{i-1}^j} \cdot \frac{\partial \phi_{i-1}^j}{\partial \boldsymbol{w}_{i-1}^{j0}}\|_1 \\ &= \|\frac{\partial \mathcal{L}}{\partial \phi_i^j}\|_1 \cdot \|\sum \boldsymbol{w}_i^{j1}\|_1 \cdot \|\gamma_L(x)\|_1.\end{aligned} \tag{17}$$

$$\begin{aligned}\|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_{i-1}^{j1}}\|_1 &= \|\frac{\partial \mathcal{L}}{\partial \phi_i^j} \cdot \frac{\partial \boldsymbol{w}_i^{j1} \cdot \phi_{i-1}}{\partial \phi_{i-1}^j} \cdot \frac{\partial \phi_{i-1}^j}{\partial \boldsymbol{w}_{i-1}^{j1}}\|_1 \\ &= \|\frac{\partial \mathcal{L}}{\partial \phi_i^j}\|_1 \cdot \|\sum \boldsymbol{w}_i^{j1}\|_1 \cdot \|\phi_{i-2}\|_1.\end{aligned} \tag{18}$$

As a result,

$$\begin{aligned}&\|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_i}\|_1 / \|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_{i-1}}\|_1 = \frac{1}{d_2}\sum_{j=1}^{d_2} \|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_i^j}\|_1 / \|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_{i-1}^j}\|_1 \\ =&\frac{1}{d_2}\sum_{j=1}^{d_2} \|(\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_i^{j0}}\|_1 + \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_i^{j1}}\|_1) / (\|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_{i-1}^{j0}}\|_1 + \|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_{i-1}^{j1}}\|_1) \\ =&\frac{1}{d_2}\sum_{j=1}^{d_2} \frac{\|\gamma_L(\boldsymbol{x})\|_1 + \|\phi_{i-1}(\gamma_L(\boldsymbol{x}))\|_1}{\|\sum \boldsymbol{w}_i^{j1}\|_1 \cdot \{\|\gamma_L(\boldsymbol{x})\|_1 + \|\phi_{i-2}(\gamma_L(\boldsymbol{x}))\|_1\}}.\end{aligned} \tag{19}$$

Accordingly, during the early stage of training, if the MLP is initialized appropriately, where $\|\sum \boldsymbol{w}_i^{j1}\|_1 \in (0, 1]$ and $\|\phi_{i-1}(\gamma_L(\boldsymbol{x}))\|_1 \approx \|\sum \boldsymbol{w}_i^{j1}\|_1 \cdot \|\phi_{i-2}(\gamma_L(\boldsymbol{x}))\|_1$, then $\|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_i}\|_1 / \|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_{i-1}}\|_1 \geq 1$ holds true in a high probability.

In practice, we find that the default initialization provided by PyTorch for MLP can meet the requirements, where the weight matrix is uniformly initialized based on the dimension of the output feature. Specifically, taking $\boldsymbol{w}_i$ as an example, since the dimension of its output feature is d_2, each element of $\boldsymbol{w}_i$ is sampled from the following uniform distribution:

$$\boldsymbol{w}_i \sim \mathcal{U}(-\frac{1}{\sqrt{d_2}}, \frac{1}{\sqrt{d_2}}). \tag{20}$$

On account that $\mathbb{E}[\boldsymbol{w}_i] = 0$, $\|\sum \boldsymbol{w}_i^{j1}\|_1 \approx \|d_2 \cdot \mathbb{E}[\boldsymbol{w}_i]\|_1 \approx 0$, which demonstrates that $\|\sum \boldsymbol{w}_i^{j1}\|_1 \in (0, 1]$ holds.

Based on Eq. 14, $\phi_{i-1}(\gamma_L(\boldsymbol{x})) = \epsilon(\boldsymbol{w}_{i-1}^0 \cdot \gamma_L(\boldsymbol{x}) + \boldsymbol{w}_{i-1}^1 \cdot \phi_{i-2}(\gamma_L(\boldsymbol{x})) + \boldsymbol{b}_i)$, where $\boldsymbol{w}_{i-1}^0 \in \mathbb{R}^{d_2 \times d_1}, \boldsymbol{w}_{i-1}^1 \in \mathbb{R}^{d_2 \times d_2}$. For an easier illustration and demonstration, we omit the influence of ϵ, $\boldsymbol{b}_i$ and $\boldsymbol{w}_{i-1}^0$, thus

$$\|\phi_{i-1}(\gamma_L(\boldsymbol{x}))\|_1 \approx \|\boldsymbol{w}_{i-1}^1 \cdot \phi_{i-2}(\gamma_L(\boldsymbol{x}))\|_1. \tag{21}$$

Because each element in $\boldsymbol{w}_{i-1}^1$ is sampled from the same distribution, without loss of generality, we assume that

$\sum w_i^{11} = \sum w_i^{21} = \cdots = \sum w_i^{d_2 1}$. Consequently,

$$\|\boldsymbol{w}_{i-1}^1 \cdot \phi_{i-2}(\gamma_L(\boldsymbol{x}))\|_1 = \|\sum \boldsymbol{w}_i^{j1}\|_1 \cdot \|\phi_{i-2}(\gamma_L(\boldsymbol{x}))\|_1. \tag{22}$$

According to Eq. 21 and Eq. 22, when the MLP is initialized by Eq. 20, Eq. 19 can be conveted into the following formulation:

$$\begin{aligned}
&\|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_i}\|_1 / \|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_{i-1}}\|_1 \\
=&\frac{1}{d_2}\sum_{j=1}^{d_2} \frac{\|\gamma_L(\boldsymbol{x})\|_1 + \|\phi_{i-1}(\gamma_L(\boldsymbol{x}))\|_1}{\|\sum \boldsymbol{w}_i^{j1}\|_1 \cdot \{\|\gamma_L(\boldsymbol{x})\|_1 + \|\phi_{i-2}(\gamma_L(\boldsymbol{x}))\|_1\}} \\
=&\frac{1}{d_2}\sum_{j=1}^{d_2} \frac{\|\gamma_L(\boldsymbol{x})\|_1 + \|\sum \boldsymbol{w}_i^{j1}\|_1 \cdot \|\phi_{i-2}(\gamma_L(\boldsymbol{x}))}{\|\sum \boldsymbol{w}_i^{j1}\|_1 \cdot \{\|\gamma_L(\boldsymbol{x})\|_1 + \|\phi_{i-2}(\gamma_L(\boldsymbol{x}))\|_1\}}.
\end{aligned} \tag{23}$$

Since $\|\sum \boldsymbol{w}_i^{j1}\|_1 \in (0, 1]$, $\|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_i}\|_1 / \|\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_{i-1}}\|_1 \geq 1$ holds, which demonstrates that with per-layer inputs incorporation, the amplitude of gradient of the shallow layer will be smaller than that of the deeper layer during the early stage of training.

8. Experimental Details

8.1. Datasets

We perform experiments on a wide range of benchmarks, *i.e.*, Blender [21], LLFF [20], and Shiny [46], to demonstrate the effectiveness of our proposed method.

Blender. The Blender dataset is comprised of 8 object-centric 360° inward-facing scenes, each containing 400 views. Following [12], when 8 input views are available, the images indexed $86, 93, 75, 26, 55, 73, 16, 2$ are selected as the input views; when 4 input views are available, the images indexed $26, 86, 2, 55$ are selected as the input views. For evaluation, the testing images are selected following [50]. The resolution for both training views and testing views is 400×400.

LLFF. The LLFF dataset consists of 8 real-world forward-facing scenes. Following [23], for each scene, every 8-th view is used as the holdout testing set and the training images are selected evenly from the remaining views. The resolution for both training views and testing views is 378×504. We report results when 3/6/9 input views are available.

Shiny. Similar to LLFF, the Shiny dataset also contains forward-facing scenes, while it is more complex due to its view-dependent effects such as reflection and refraction. We choose 6 scenes from the origional Shiny dataset and 2 scenes from the Shiny-extended dataset, where the resolution for each scene is 378×504. The training and testing images are sampled following [23].

8.2. Metrics

To measure the performance of our proposed method, we evaluate the quality of rendered novel views using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) [45], and Learned Perceptual Image Patch Similarity (LPIPS) [52]. Additionally, for an easier comparison, we also report the average score by calculating the geometric mean of MSE $= 10^{-\text{PSNR}/10}$, $\sqrt{1-\text{SSIM}}$ and LPIPS following [23].

8.3. Implementation details

We implement our approach using the nerf-pytorch codebase[1], with vanilla NeRF as our baseline. For Blender, the values for L_1, L_2, and L_3 are set to be 2, 6, and 10 respectively; for LLFF and Shiny, the values for L_1, L_2, and L_3 are set to be 2, 8, and 10 respectively. For L_{BR}, rays are sampled from extrapolated image space where $p_x \in [-H/2, H + H/2]$ and $p_y \in [-W/2, W + W/2]$. For sampling annealing, we set $N_{\max} = 256$, $N_{start} = 16$, and $\eta = 100$. The Color Branch C_θ and Density Branch D_θ both have 8 layers with 256 neurons per layer. We apply 50K training iterations for Blender, while 200K training iterations for LLFF and Shiny. All experiments are performed on a single NVIDIA RTX 3090 GPU with a batch size of 1024.

8.4. Methods Used for Comparison

To demonstrate the superiority of our proposed method, we compare it against several baselines, as well as prior-based and regularization-based methods. For baselines, we choose vanilla NeRF [21], Mip-NeRF [2], and Ref-NeRF [39], which are all representative methods for novel view synthesis. For prior-based methods, following [50], we choose SRF [7], PixelNeRF [51], and MVSNeRF [4], where a large dataset is utilized to incorporate learned priors. We also report the fine-tuning results of these methods on LLFF and Shiny, which are able to obtain better performance. For regularization-based methods, we choose DietNeRF [12], InfoNeRF [14], RegNeRF [23], MixNeRF [33], as well as FreeNeRF [50], which are state-of-the-art methods for few-shot view synthesis to date. Notably, since the pre-trained network used in RegNeRF for appearance regularization is not provided, we only report the results of RegNeRF trained without it.

[1] https://github.com/yenchenlin/nerf-pytorch

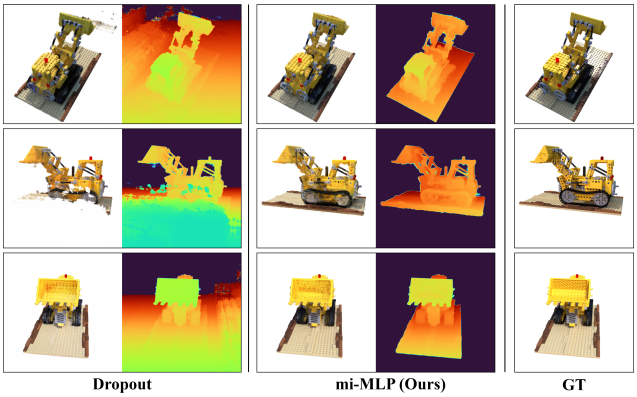


Figure 9. The rendered novel views and estimated depth map by Dropout [36] and our proposed method respectively.

Method	PSNR↑	SSIM↑	LPIPS↓	Average↓
DietNeRF [12]	23.83	0.859	0.117	0.056
Dropout [36]	23.79	0.875	0.108	0.054
mi-MLP (Ours)	25.50	0.887	0.087	0.043

Table 6. The quantitative results of DietNeRF [12], Dropout [36], and our proposed mi-MLP respectively.

9. Results of Using Dropout to Avoid Overfitting

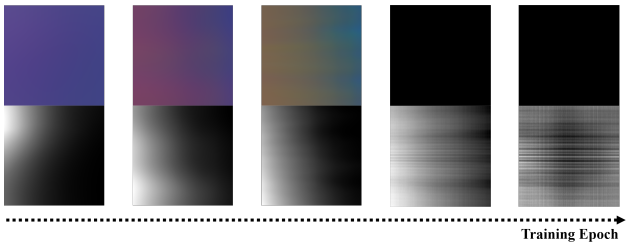
As explained in Sec. 5.2, we demonstrate the effectiveness of our mi-MLP by comparing it with the classical representative technique, *i.e.*, Dropout [36], which is used to avoid the overfitting problem.

Concretely, as shown in Fig. 9, for a testing scene randomly selected from the Blender dataset, a direct application of Dropout to the NeRF MLP leads to severe artifacts as well as unreasonable geometry. In contrast, our proposed method can achieve both photorealistic renderings and clear depth estimation. Quantitatively, as demonstrated in Tab. 6, Dropout obtains a comparable performance with DietNeRF [12], while a great performance improvement can be witnessed by using our mi-MLP.

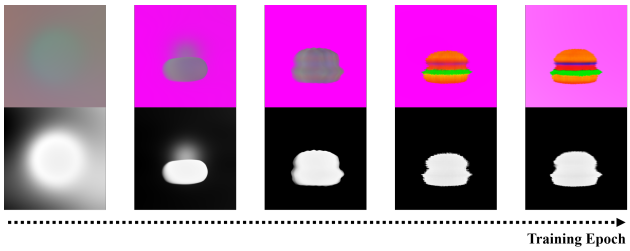
10. Extensions to 3D Generation

As an important topic in computer vision and graphics, 3D generation can be viewed as an extreme case of few-shot view synthesis, where only one reference image or a textual description is available. To demonstrate the potential of our proposed method, we extend it to the task of Text-to-3D. Specifically, we take the public-available stable-dreamfusion[2] as an example, where we replace the network structure with our proposed mi-MLP.

[2]A reimplemented version of DreamFusion [26] by Stable Diffusion



(a) The training process of original stable-dreamfusion.



(b) The training process of combining stable-dreamfusion with our proposed mi-MLP.

Figure 10. Given the input textual description, *i.e.*, "a hamburger", (a) illustration of the training process of original stable-dreamfusion, (b) as well as the training process of modified stable-dreamfusion whose network structure is replaced by our proposed mi-MLP.

As shown in Fig. 10, combined with mi-MLP, the stable-dreamfusion is more robust to different textual descriptions, where reasonable results can be generated. In contrast, sometimes the original stable-dreamfusion falls into a degradation solution, generating nothing but pure colors. Such an observation shows that mi-MLP is beneficial to generating diverse 3D assets, which opens up a new direction for future research.

11. Limitations and Future Works

Our proposed method aims to realize the task of few-shot view synthesis from the perspective of network structure, where state-of-the-art performance can be achieved. However, for objects with complex textures or thin structures, consistency across different views is hardly guaranteed since we impose no constraints on unknown novel views. To solve this problem, future works include introducing additional regularization terms or utilizing learned priors for better novel view synthesis.

Blender-4									
Scene	**chair**	**drums**	**ficus**	**hotdog**	**lego**	**materials**	**mic**	**ship**	**Average**
PSNR↑	23.37	15.17	19.42	24.22	20.31	19.42	20.65	20.49	20.38
SSIM↑	0.871	0.711	0.840	0.887	0.835	0.834	0.903	0.742	0.828
LPIPS↓	0.121	0.258	0.129	0.121	0.160	0.124	0.109	0.238	0.157
Average↓	0.058	0.161	0.083	0.053	0.084	0.083	0.066	0.102	0.084
Blender-8									
Scene	**chair**	**drums**	**ficus**	**hotdog**	**lego**	**materials**	**mic**	**ship**	**Average**
PSNR↑	27.75	19.85	21.49	29.85	25.50	22.33	27.15	23.69	24.70
SSIM↑	0.936	0.844	0.876	0.948	0.887	0.864	0.949	0.777	0.885
LPIPS↓	0.055	0.110	0.108	0.051	0.087	0.084	0.045	0.159	0.087
Average↓	0.028	0.076	0.064	0.022	0.043	0.056	0.026	0.068	0.046

Table 7. The quantitative results for scenes in the Blender dataset with 4/8 input views available.

LLFF-3									
Scene	**fern**	**flower**	**fortress**	**horns**	**leaves**	**orcids**	**room**	**trex**	**Average**
PSNR↑	21.43	19.61	23.38	17.40	16.42	15.70	23.17	20.86	19.75
SSIM↑	0.675	0.580	0.580	0.486	0.494	0.470	0.880	0.741	0.614
LPIPS↓	0.273	0.311	0.274	0.424	0.389	0.356	0.161	0.211	0.300
Average↓	0.103	0.130	0.093	0.176	0.184	0.191	0.064	0.095	0.125
LLFF-6									
Scene	**fern**	**flower**	**fortress**	**horns**	**leaves**	**orcids**	**room**	**trex**	**Average**
PSNR↑	24.44	23.92	27.49	23.36	19.46	17.56	29.36	22.94	23.57
SSIM↑	0.803	0.800	0.839	0.810	0.738	0.558	0.922	0.831	0.788
LPIPS↓	0.164	0.130	0.116	0.179	0.177	0.280	0.098	0.164	0.163
Average↓	0.063	0.061	0.043	0.071	0.100	0.148	0.031	0.069	0.069
LLFF-9									
Scene	**fern**	**flower**	**fortress**	**horns**	**leaves**	**orcids**	**room**	**trex**	**Average**
PSNR↑	25.89	25.94	29.26	25.37	20.95	18.50	29.93	25.33	25.15
SSIM↑	0.846	0.854	0.883	0.862	0.788	0.615	0.934	0.889	0.834
LPIPS↓	0.135	0.111	0.092	0.147	0.161	0.259	0.093	0.125	0.140
Average↓	0.051	0.047	0.033	0.054	0.084	0.131	0.028	0.049	0.055

Table 8. The quantitative results for scenes in the LLFF dataset with 3/6/9 input views available.

Shiny-3									
Scene	**cake**	**crest**	**food**	**giants**	**pasta**	**room**	**seasoning**	**tools**	**Average**
PSNR↑	20.41	14.31	16.06	18.98	14.78	22.37	19.78	19.28	18.24
SSIM↑	0.439	0.251	0.353	0.509	0.442	0.569	0.506	0.730	0.475
LPIPS↓	0.395	0.579	0.467	0.407	0.370	0.375	0.456	0.276	0.415
Average↓	0.139	0.264	0.210	0.153	0.209	0.112	0.149	0.119	0.165

Table 9. The quantitative results for scenes in the Shiny dataset with 3 input views available.