Grounding and Enhancing Grid-based Models for Neural Fields

Supplementary Material

7. Proofs to theorems

One of our major contributions is to reveal the power of NTK theories [3, 17, 46] to grid-based models. The main paper introduces several claims based on our introduced grid tangent kernel (GTK). We introduce proofs in this section and provide discussions and interpretations of claims in grid-based unbounded radiance fields.

7.1. Settings

We present our notations in the Table 6. Following previous works [38, 46], we build up the analysis framework for grid-based models in a supervised regression setting. For simplicity, we assume the model weights are initialized to zero tensors:

$$\boldsymbol{w}(0) = \mathbf{0}. \tag{10}$$

We use a regression loss to measure a grid-based model $g_{\boldsymbol{w}}$ parameterized by $\boldsymbol{w}$:

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(\boldsymbol{Y}_i - g_{\boldsymbol{w}}(\boldsymbol{X}_i)\right)^2, \tag{11}$$

and we assume $0 \le \boldsymbol{Y}_i \le 1, 0 \le g_{\boldsymbol{w}} \le 1$. Model parameters evolve following gradient descent (GD), where the r-th weight $\boldsymbol{w}_r$ can be written as:

$$\boldsymbol{w}_r(t+1) - \boldsymbol{w}_r(t) = -\eta_l \frac{\partial \mathcal{L}(\boldsymbol{w}(t))}{\partial \boldsymbol{w}_r}. \tag{12}$$

The vector form of this update rule is:

$$\text{vec}(\boldsymbol{w}(t+1)) = \text{vec}(\boldsymbol{w}(t)) - \eta_l \boldsymbol{Z}(t)(\boldsymbol{O}(t) - \boldsymbol{Y}_i), \tag{13}$$

where $\boldsymbol{Z}(t)$ is the gradient matrix at timestep t:

$$\boldsymbol{Z}(t) = \left(\frac{\partial g\left(\boldsymbol{X}_i, \boldsymbol{w}(t)\right)}{\partial \boldsymbol{w}_1}, \frac{\partial g\left(\boldsymbol{X}_i, \boldsymbol{w}(t)\right)}{\partial \boldsymbol{w}_2}, ..., \frac{\partial g\left(\boldsymbol{X}_i, \boldsymbol{w}(t)\right)}{\partial \boldsymbol{w}_r}\right)^T. \tag{14}$$

The continuous form of model dynamics can be described via gradient flow, which is an ODE [38]:

$$\begin{aligned}\frac{d\boldsymbol{w}(t)}{dt} &= -\nabla\mathcal{L}(\boldsymbol{w}(t)) \\ &= -\sum_{i=1}^{n}\left(g\left(\boldsymbol{X}_i, \boldsymbol{w}(t)\right) - \boldsymbol{Y}_i\right)\frac{\partial g\left(\boldsymbol{X}_i, \boldsymbol{w}(t)\right)}{\partial \boldsymbol{w}}.\end{aligned} \tag{15}$$

7.2. Grid-based models and their derivatives

A grid model can be represented via a weighted average of features on the grid nodes, as shown in its definition:

$$g(\boldsymbol{X}_i, \boldsymbol{w}) \triangleq \sum_{i \in U(\boldsymbol{X}_i)} \varphi\left(\boldsymbol{X}_i, \Theta_i\right)\boldsymbol{w}_i. \tag{16}$$

For ease of mathematical analysis, we generalize the summation to all parameters instead of the surrounding index $U(\boldsymbol{X})$. Such a procedure can be achieved by setting the kernel φ to zero at non-local indexes:

$$g(\boldsymbol{X}_i, \boldsymbol{w}) = \sum_{r=1}^{m} \varphi\left(\boldsymbol{X}_i, \Theta_r\right)\boldsymbol{w}_r. \tag{17}$$

We further define the vector form of the nodal function as:

$$\varphi(\boldsymbol{X}_i) = (\varphi(\boldsymbol{X}_i, \Theta_1), \varphi(\boldsymbol{X}_i, \Theta_2), ..., \varphi(\boldsymbol{X}_i, \Theta_m))^T. \tag{18}$$

Grid-based models require that the kernel function φ is only determined by the node index and the input data, and the kernel function is not changed during a concerned period of time. This constraint holds for many state-of-the-art grid-based models [18, 31, 54]. Although in some cases, such as adaptive learning [9], the kernel function is optimized along with the feature vectors, the assumption is still a reasonable one by assuming that the kernel function is updated much slower than the grid features $\boldsymbol{w}$. Therefore, during a short enough time span, the kernel function can be regarded as a static one.

The summarization of the kernel function is one (due to the normalization procedure Equation (8)):

$$\sum_{r=1}^{m} \varphi(\boldsymbol{X}_i, \Theta_r) = 1, \forall \boldsymbol{X}_i. \tag{19}$$

7.3. Proof of Theorem 1

In Theorem 1 of our paper, we show that the training dynamics of grid-based models are associated with the GTK:

Proof. The model parameters evolve according to the following differential equation:

$$\frac{dg\left(\boldsymbol{X}_i, \boldsymbol{w}(t)\right)}{dt} = \frac{d\boldsymbol{w}(t)}{dt} * \frac{\partial g\left(\boldsymbol{X}_i, \boldsymbol{w}(t)\right)}{\partial \boldsymbol{w}}. \tag{20}$$

Considering Equation (15) and the definition of GTK, we have:

$$\frac{dg\left(\boldsymbol{X}_i, \boldsymbol{w}(t)\right)}{dt} = -\sum_{j=1}^{n}\left(g\left(\boldsymbol{X}_j, \boldsymbol{w}(t)\right) - \boldsymbol{Y}_j\right)[\boldsymbol{G}_g(t)]_{i,j}. \tag{21}$$

We can write it in a compact form, while considering the fact that $\boldsymbol{Y}$ is not changed during training:

$$\frac{d\boldsymbol{O}(t)}{dt} = -\boldsymbol{G}_g(t) \cdot (\boldsymbol{O}(t) - \boldsymbol{Y}). \tag{22}$$

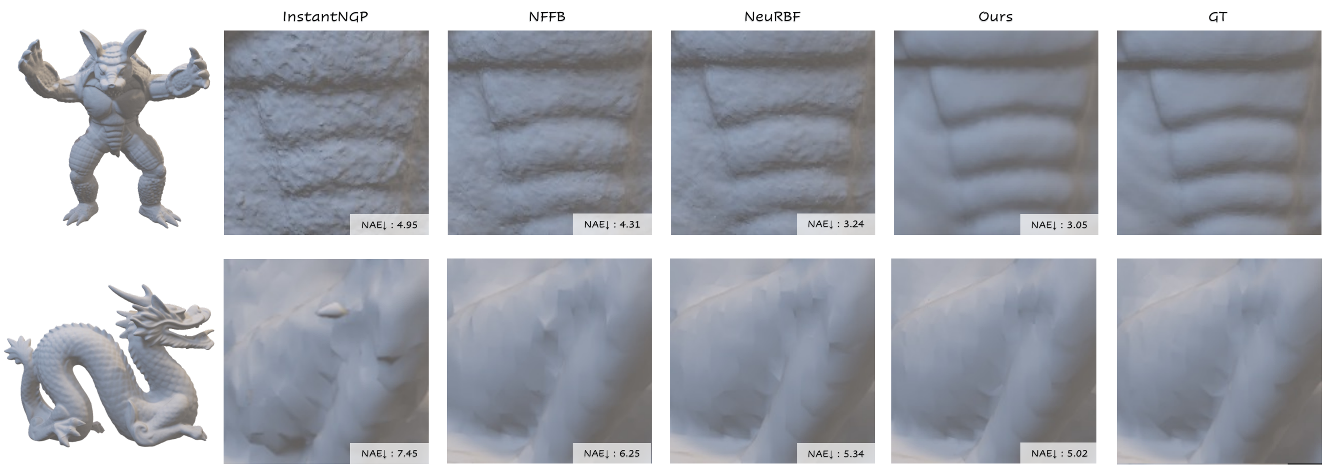

Figure 6. 3D signed distance field (SDF) reconstruction results comparing to InstantNGP [31], NFFB [54], NeuRBF [9]. We show the reconstructed geometry of our approach at the leftmost column. We show the normal angular errors (NAE) in this figure.

Variable	Definition
f	a function
g	a grid-based model
$\boldsymbol{X}_i$	an input data
$\boldsymbol{Y}_i$	a label
n	the size of the dataset
m	the number of parameters in the model
d	the dimension of the output feature
$\boldsymbol{A}$	a matrix
$\boldsymbol{A}_{ij}$	the (i, j)-th entry of $\boldsymbol{A}$
$\|\cdot\|_2$	the Euclidean norm of a vector
$\|\cdot\|_F$	the Frobenius norm of a matrix
$\lambda_{\min}(\boldsymbol{A})$	the minimum eigenvalue of a symmetric matrix $\boldsymbol{A}$
$\mathrm{vec}(\mathbf{A})$	the vectorization of a matrix $\boldsymbol{A}$
$\boldsymbol{I}$ or $\boldsymbol{I}_n$	the identity matrix with shape $n \times n$
$\boldsymbol{w}(t)$	the weight matrix of shape $r \times d$ at timestep t
$\boldsymbol{w}_r(t)$	the r-th weight of the model at timestep t
$S = \{(\boldsymbol{X}_i, \boldsymbol{Y}_i)\}_{i=1}^n$	input-label samples
$\boldsymbol{O}(t) = g_{\boldsymbol{w}}(\boldsymbol{X}_i) = g(\boldsymbol{X}_i, \boldsymbol{w})$	a model with weights $\boldsymbol{w}$ and inputs $\boldsymbol{X}_i$
$\boldsymbol{Z}(t)$	the gradient matrix at timestep t, see Equation (14)
$\boldsymbol{G}_g(t)$	the GTK matrix of a grid-based model g at timestep t
$\mathcal{L}$	the loss
η_l	the learning rate
γ	the Fourier feature mapping
φ	the nodal function
$\mathcal{F}$	a function class
ε	random variables from $\{-1, 1\}$
B	an upper bound of weight change
$B(\boldsymbol{w})$	defined in Equation (32b)
k_o	a coefficient used in Equation (35)
$\boldsymbol{T}$	defined in Equation (39)

Table 6. Definitions of notations in this paper.

The discrete version of Equation (22) with a learning rate η_l can be written as:

$$\boldsymbol{O}(t+1) - \boldsymbol{O}(t) = \eta_l \boldsymbol{G}_g(t)(\boldsymbol{O}(t) - \boldsymbol{Y}). \tag{23}$$

□

Discussions. Another equivalent definition of GTK given the gradient matrix $\boldsymbol{Z}$ is:

$$\boldsymbol{G}_g(t) = \boldsymbol{Z}(t)^T \boldsymbol{Z}(t). \tag{24}$$

This theorem shows that GTK connects the error term $\boldsymbol{O}(t) - \boldsymbol{Y}$ to the changing rate of the output. Therefore, this theorem can be used to analyze the training behaviors of grid-based models. We further show in Theorem 2 that GTK keeps constant during training, and therefore, standard kernel regression methods [38] can be applied to analyze behaviors of grid-based models.

7.4. Proof of Theorem 2

Proof. According to the previous analysis, the kernel function φ remains constant during training. Therefore, according to Equation (14) and Equation (17), the gradient matrix can be written as:

$$\boldsymbol{Z}(t) = \varphi(\boldsymbol{X}_i). \tag{25}$$

Therefore, the gradient matrix remains constant during training:

$$\boldsymbol{Z}(t) = \boldsymbol{Z}(0) = \boldsymbol{Z}. \tag{26}$$

According to Equation (24), we can conclude that the GTK is not changed across training. Therefore, we have:

$$\boldsymbol{G}_g(t) = \boldsymbol{G}_g(0), \tag{27}$$

□

Discussions. This theorem shows that the GTK is unchanged during training for grid-based models. Therefore, GTK is a powerful tool for understanding these grid-based models' training and generalization properties. We can call these grid-based models as *quasi-linear* models because although the model is not linear regarding the input data $\boldsymbol{X}_i$, the model is linear regarding the weights. Different from NTK [17], which is constant only when the network width is infinite, the property of GTK is not asymptotic, which means that grid-based models might be better understood than conventional neural networks (MLPs).

7.5. Proof of Theorem 3

Proof. The major technique uses the empirical Rademacher complexity to bound the population loss according to the following theorem from [3]. We first recap the definition of population loss and empirical loss:

$$\mathcal{L}_{\mathcal{D}}(t) = \mathbb{E}_{(\boldsymbol{X}_i,\boldsymbol{Y}_i)\sim\mathcal{D}}\left[\mathcal{L}\left(f\left(\boldsymbol{X}_i, \boldsymbol{w}(t)\right), \boldsymbol{Y}_i\right)\right], \quad (28)$$

$$\mathcal{L}_S(t) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}\left(f\left(\boldsymbol{X}_i, \boldsymbol{w}(t)\right), \boldsymbol{Y}_i\right). \quad (29)$$

Then, we introduce a useful theorem from [3].

Theorem 4. *(from* ***Theorem B.1*** *of [3]) Given a set of n samples S, the empirical Rademacher complexity of a function class $\mathcal{F}$ (mapping from $\mathbb{R}^d$ to $\mathbb{R}$) is defined as:*

$$\mathcal{R}_S(\mathcal{F}) = \frac{1}{n}\mathbb{E}_{\boldsymbol{\varepsilon}\in\{\pm1\}^n}\left[\sup_{f\in\mathcal{F}}\sum_{i=1}^{n}\varepsilon_i f\left(\boldsymbol{X}_i\right)\right], \quad (30)$$

where ε contains i.i.d random variables drawn from a uniform Rademacher distribution in $\{-1, 1\}$. Given a bounded loss function $\mathcal{L}(\cdot,\cdot)$, which is 1-Lipschitz in the first argument. Then with probability at least $1-\delta_p$ over sample S of size n:

$$\sup_{f\in\mathcal{F}}\{\mathcal{L}_{\mathcal{D}}(f) - \mathcal{L}_S(f)\} \le 2\mathcal{R}_S(\mathcal{F}) + 3c\sqrt{\frac{\log(2/\delta_p)}{2n}}. \quad (31)$$

Given a bound $B > 0$ (we will calculate B in our case later), we consider a bounded function of grid-based models:

$$\mathcal{F}_B^{\boldsymbol{w}(0)} = \{g_{\boldsymbol{w}} : B(\boldsymbol{w})\}, \quad (32a)$$

$$B(\boldsymbol{w}) \triangleq \|\boldsymbol{w} - \boldsymbol{w}(0)\|_F \le B. \quad (32b)$$

We calculate the empirical Rademacher complexity as follows:

$$\begin{aligned}\mathcal{R}_S(\mathcal{F}_B^{\boldsymbol{w}(0)}) &= \frac{1}{n}\mathbb{E}_{\boldsymbol{\varepsilon}\in\{\pm1\}^n}\left[\sup_{f\in\mathcal{F}_B^{\boldsymbol{w}(0)}}\sum_{i=1}^{n}\boldsymbol{\varepsilon}_i g\left(\boldsymbol{X}_i\right)\right] \\ &= \frac{1}{n}\mathbb{E}_{\boldsymbol{\varepsilon}\in\{\pm1\}^n}\left[\sup_{B(\boldsymbol{w})}\sum_{i=1}^{n}\boldsymbol{\varepsilon}_i\sum_{r=1}^{m}\varphi(\boldsymbol{X}_i,\Theta_r)\boldsymbol{w}_r\right],\end{aligned} \quad (33)$$

where $\varphi(\boldsymbol{X}_i, \Theta_i)$ is the kernel function, and here we levearage Equation (17).

Considering Equation (26), we can write the above equation as:

$$\begin{aligned}\mathcal{R}_S(\mathcal{F}_B^{\boldsymbol{w}(0)}) &= \frac{1}{n}\mathbb{E}_{\boldsymbol{\varepsilon}\in\{\pm1\}^n}\left[\sup_{B(\boldsymbol{w})}\operatorname{vec}(\boldsymbol{w})^T\boldsymbol{Z}\boldsymbol{\varepsilon}\right] \\ &= \frac{1}{n}\mathbb{E}_{\boldsymbol{\varepsilon}\in\{\pm1\}^n}\left[\sup_{B(\boldsymbol{w})}\operatorname{vec}(\boldsymbol{w})^T\boldsymbol{Z}(0)\boldsymbol{\varepsilon}\right] \\ &= \frac{1}{n}\mathbb{E}_{\boldsymbol{\varepsilon}\in\{\pm1\}^n}\left[\sup_{B(\boldsymbol{w})}\operatorname{vec}(\boldsymbol{w}-\boldsymbol{w}(0))^T\boldsymbol{Z}(0)\boldsymbol{\varepsilon}\right] \\ &\le \frac{1}{n}\mathbb{E}_{\boldsymbol{\varepsilon}\in\{\pm1\}^n}\left[B\cdot\|\boldsymbol{Z}(0)\boldsymbol{\varepsilon}\|_2\right] \\ &\le \frac{B}{n}\sqrt{\mathop{\mathbb{E}}_{\boldsymbol{\varepsilon}\sim\{\pm1\}^n}\left[\|\boldsymbol{Z}(0)\boldsymbol{\varepsilon}\|_2^2\right]} \\ &= \frac{B}{n}\|\boldsymbol{Z}(0)\|_F.\end{aligned} \quad (34)$$

We first bound $\|\boldsymbol{Z}(0)\|_F$:

$$\begin{aligned}\|\boldsymbol{Z}(0)\|_F^2 &= \sum_{r=1}^{m}\sum_{i=1}^{n}\varphi^2(\boldsymbol{X}_i,\Theta_r) \\ &= \sum_{i=1}^{n}\sum_{r=1}^{m}\varphi^2(\boldsymbol{X}_i,\Theta_r) \\ &\le k_o n.\end{aligned} \quad (35)$$

The design and weights of the kernel function affect the constant k_o. Since it is a common practice in NTK theories [3, 17] that we scale the output (and therefore the gradient) of the network by a constant, we set $k_o = \frac{1}{2}$ to make the resulting generalization bound consistent with that in the NTK theory [3]. The exact value of k_o does not affect the conclusions of our analysis, and it's safe to set $k_o = \frac{1}{2}$. We now have the following:

$$\mathcal{R}_S(\mathcal{F}_B^{\boldsymbol{w}(0)}) \le \frac{1}{\sqrt{2n}}B. \quad (36)$$

Then we prove Equation (32b) holds and calculates the upper bound B in the equation. We start from Equation (23) and apply Equation (27):

$$\boldsymbol{O}(k+1) - \boldsymbol{O}(k) = \eta_l\boldsymbol{G}_g(\boldsymbol{O}(k) - \boldsymbol{Y}). \quad (37)$$

Recursively applying the above equation, we can derive the following:

$$\begin{aligned} \boldsymbol{O}(k)-\boldsymbol{Y} &= -(\boldsymbol{I}-\eta_l \boldsymbol{G}_g)^k(\boldsymbol{O}(0)-\boldsymbol{Y}) \\ &= -(\boldsymbol{I}-\eta_l \boldsymbol{G}_g)^k \boldsymbol{Y}. \end{aligned} \tag{38}$$

Here, we use the assumption that the model weights are all set to zero stated in Equation (10), and we use the definition of the model predictions in Equation (17). We introduce a polynomial of $\boldsymbol{G}_g$ as:

$$\boldsymbol{T} \triangleq \sum_{i=0}^{k-1} \eta_l (\boldsymbol{I}-\eta_l \boldsymbol{G}_g)^i \tag{39}$$

Then we plug the above result including Equation (26) into Equation (13):

$$\begin{aligned} \|\boldsymbol{w}(k)-\boldsymbol{w}(0)\|_F &= \sqrt{\|\operatorname{vec}(\boldsymbol{w}(k))-\operatorname{vec}(\boldsymbol{w}(0))\|_2^2} \\ &= \sqrt{\left\|\sum_{i=0}^{k-1} \eta_l \boldsymbol{Z}(\boldsymbol{I}-\eta_l \boldsymbol{G}_g)^i \boldsymbol{Y}\right\|_2^2} \\ &= \sqrt{\|\boldsymbol{ZTY}\|_2^2} \\ &= \sqrt{\boldsymbol{Y}^T \boldsymbol{T}^T \boldsymbol{Z}^T \boldsymbol{ZTY}} \\ &= \sqrt{\boldsymbol{Y}^T \boldsymbol{T} \boldsymbol{Z}^T \boldsymbol{ZTY}} \\ &= \sqrt{\boldsymbol{Y}^T \boldsymbol{T} \boldsymbol{G}_g \boldsymbol{TY}}, \end{aligned} \tag{40}$$

where we use Equation (24) and we consider the fact that $\boldsymbol{T}$ is a symmetric matrix. Decompose the matrix of $\boldsymbol{G}_g$ as follows:

$$\boldsymbol{G}_g = \sum_{i=1}^{n} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top. \tag{41}$$

Since $\boldsymbol{T}$ is a polynomial of $\boldsymbol{G}_g$, its eigenvectors are the same as $\boldsymbol{G}_g$, and we have:

$$\begin{aligned} \boldsymbol{T} &= \sum_{i=1}^{n} \eta_l \sum_{j=0}^{k-1} (1-\eta_l \lambda_i)^j \boldsymbol{v}_i \boldsymbol{v}_i^\top \\ &= \sum_{i=1}^{n} \frac{1-(1-\eta_l \lambda_i)^k}{\lambda_i} \boldsymbol{v}_i \boldsymbol{v}_i^\top. \end{aligned} \tag{42}$$

Therefore, we have:

$$\begin{aligned} \boldsymbol{T} \boldsymbol{G}_g \boldsymbol{T} &= \sum_{i=1}^{n} \left(\frac{1-(1-\eta_l \lambda_i)^k}{\lambda_i}\right)^2 \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top \\ &\preceq \sum_{i=1}^{n} \frac{1}{\lambda_i} \boldsymbol{v}_i \boldsymbol{v}_i^\top \\ &= (\boldsymbol{G}_g)^{-1}. \end{aligned} \tag{43}$$

Plug this into Equation (40), we have:

$$\|\boldsymbol{w}(k)-\boldsymbol{w}(0)\|_F \leq \sqrt{\boldsymbol{Y}^\top \boldsymbol{G}_g^{-1} \boldsymbol{Y}}. \tag{44}$$

Here we have proved Equation (32b).

Set $B = \boldsymbol{Y}^\top \boldsymbol{G}_g^{-1} \boldsymbol{Y}$ and plug into Equation (36), we have:

$$\mathcal{R}_S(\mathcal{F}_B^{\boldsymbol{w}(0)}) \leq \sqrt{\frac{\boldsymbol{Y}^\top \boldsymbol{G}_g^{-1} \boldsymbol{Y}}{2n}}. \tag{45}$$

Then we are ready to apply Theorem 4:

$$\begin{aligned} LHS &= \sup_{f \in \mathcal{F}_B^{\boldsymbol{w}(0)}} \{\mathcal{L}_{\mathcal{D}}(f) - \mathcal{L}_S(f)\} \\ &\leq \sqrt{\frac{2\boldsymbol{Y}^\top \boldsymbol{G}_g^{-1} \boldsymbol{Y}}{n}} + 3\sqrt{\frac{\log(\frac{2}{\delta_p})}{2n}} \\ &= \sqrt{\frac{2\boldsymbol{Y}^\top \boldsymbol{G}_g^{-1} \boldsymbol{Y}}{n}} + O\left(\sqrt{\frac{\log \frac{2}{\delta_p}}{n}}\right). \end{aligned} \tag{46}$$

Bound $\mathcal{L}_S(f)$ as follows (considering Equation (11)):

$$\begin{aligned} \mathcal{L}_S(f) &= \frac{1}{n} \sum_{i=1}^{n} [\mathcal{L}(\boldsymbol{O}_i(k), \boldsymbol{Y}_i) - \mathcal{L}(\boldsymbol{Y}_i, \boldsymbol{Y}_i)] \\ &\leq \frac{1}{n} \sum_{i=1}^{n} |\boldsymbol{O}_i(k) - \boldsymbol{Y}_i| \\ &\leq \frac{1}{\sqrt{n}} \|\boldsymbol{O}(k) - \boldsymbol{Y}\|_2 \\ &= \sqrt{\frac{2\mathcal{L}(\boldsymbol{w}(k))}{n}} \\ &\leq \frac{1}{\sqrt{n}}, \end{aligned} \tag{47}$$

where we use the fact that the loss $\mathcal{L}(\boldsymbol{w}(k)) \leq \frac{1}{2}$. Therefore, we have proved Theorem 3. □

Discussions. The key insight behind this proof procedure is that the generalization gap is strongly associated with weight change during training. Therefore, if we can narrow down the required weight change across training (e.g., adding more inductive bias or setting proper initialization), we will have a model that generalizes better.

Another insight is that the generalization gap is associated with the dominating term $\Delta = \boldsymbol{Y}^\top \boldsymbol{G}_g^{-1} \boldsymbol{Y}$, which is a quadratic form of $\boldsymbol{G}_g^{-1}$. We may use this knowledge to motivate designs of future grid-based models better.

Also, labels $\boldsymbol{Y}$ will affect the generalization ability of grid-based models. This could explain why pose initialization accuracy and point cloud initializations greatly matter to radiance field reconstruction [18, 47, 59].

7.6. NeRF experimental details

In unbounded scenes, one must warp the scene into normalized device coordinates (NDC [4, 29]) before feeding the coordinates into neural networks. Following Mip-NeRF 360 [5] and DVGOv2 [44], we use a two-layer parameterization to model near and far objects separately. Formally, for a sampled point $\boldsymbol{p}$ in the ray $\boldsymbol{r}$ where its real-world coordinate $\boldsymbol{X}_w^{\boldsymbol{p}}$ is transferred to the normalized one:

$$\boldsymbol{X}_n^{\boldsymbol{p}} = \begin{cases} \boldsymbol{X}_w^{\boldsymbol{p}}, & \|\boldsymbol{X}_w^{\boldsymbol{p}}\|_\infty \leq 1, \\ \left(1 + b - \frac{b}{\|\boldsymbol{X}_w^{\boldsymbol{p}}\|_\infty}\right) \frac{\boldsymbol{X}_w^{\boldsymbol{p}}}{\|\boldsymbol{X}_w^{\boldsymbol{p}}\|_\infty}, & \|\boldsymbol{X}_w^{\boldsymbol{p}}\|_\infty > 1, \end{cases} \tag{48}$$

where b is a hyperparameter of the background length. We set $b = 0.2$ in all experiments.

The Fourier feature length l is set to five in all the experiments. The size of the voxels in grid-based models for NeRF experiments is set to 320^3 ($N_x = N_y = N_z = 320$). The post-activation bias is set to $\eta_b = 1 \times e^{-3}$. We use $\eta_{\mathcal{F}} = 0.5$ to balance two losses. We use the Adam optimizer [19] with a batch size of 8192 rays to optimize the representation for $40k$ iterations. We use a constant learning rate $\eta_l = 1 \times e^{-3}$ without learning rate decay. Our implementations are based on Pytorch. The speed test is conducted on a single NVIDIA 3090Ti GPU card and averaged numbers across three runs are reported to avoid random noises. The other setup of the speed test follows previous works [9, 44]. More details are in the supplementary.

8. Additional NeRF dataset details

We produce additional NeRF dataset details on unbounded datasets here. **Tanks&Temples [20].** We show experimental results on four large-scale scenes provided by [20]. All the scenes are hand-held 360-degree captures, and camera poses are estimated by COLMAP [41]. We use the same dataset split as DVGOv2 [44].

Mip-NeRF-360 [5]. A dataset of seven scenes was published. Each scene contains a challenging central object in the background with rich details. Camera poses are derived via COLMAP [41]. Comparison experiments follow the dataset split of previous work [5, 44].

San Francisco Mission Bay (SFMB) [47]. This is a street scene dataset released by Block-NeRF [47]. The images are captured in San Francisco's Mission Bay District. Twelve cameras on a vehicle record images from different angles. Collected images are divided into `train` and `test` splits. We generate a virtual driving camera sequence, including rotating and forwarding poses, in the `render` split. We do not compare to the full version of Block-NeRF [47], which is not open-sourced. However, we re-implement their block division technique for baselines and ours: we train separate models for different blocks, and renderings are generated via the block composition [47]. The number, sizes, and positions of blocks are the same for all compared methods. Please refer to the supplementary for more details. On SFMB [47], we use one block to measure the number of parameters and the training time.

9. 3D SDF reconstruction visualizations

We provide visualizations of SDF estimation in Figure 6. From this visualization, we can find that our results can recover more fine-grained details compared to other methods. Moreover, our results often produce more smooth surfaces compared to other baselines.